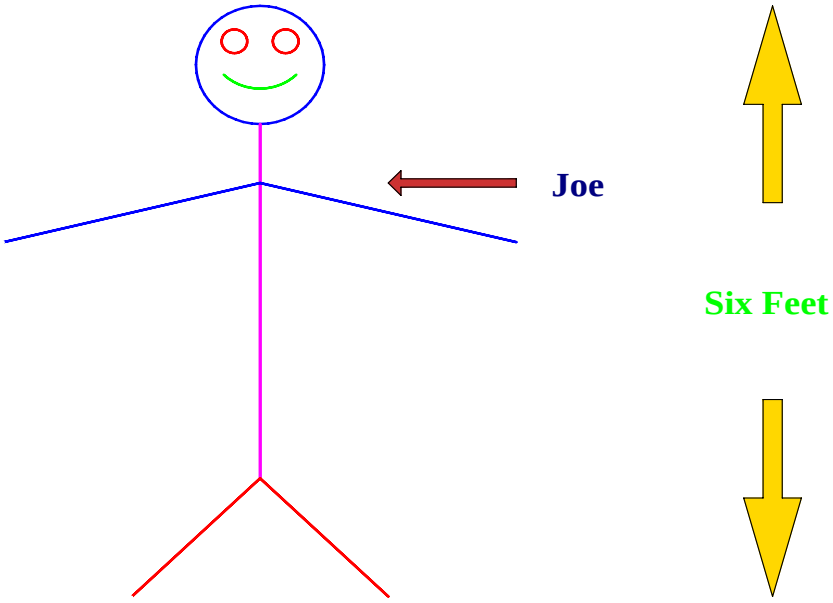
A WHS manual for authors (Draft)

Paul Eakin, Carl Eberhart,Ken Kubota
Department of Mathematics, University of Kentucky



June 7, 2004

Contents

1 Philosophy of this text.

'Problem solving' has been a buzzphrase in mathematics education for around 15 years (Don Coleman say's much longer). In 1989, the National Council of Teachers of Mathematics (NCTM) published a set of national standards for teachers to go by when teaching mathematics to students from Kindergarten to 12th grade. The standards emphasized that **'problem solving'** aspect of mathematics, and this idea has caught on.

Most universities and colleges now have problem solving courses as part of the curriculum for aspiring teachers, and there have been several very good books written to use in these courses. So why should we write another book on problem solving? Well, as the title suggests, our approach to problem solving is to emphasize the teacher role in problem solving. We will concentrate on the activities a teacher engages in when making up problems to teach the student to solve problems, with special emphasis on the preparation of homework sets for posting on the Web Homework System, hosted on mathclass.com, a University of **Kentucky Mathematical Sciences server** .

The text is made available in multiple formats. The most familiar form is the portable document format or **pdf file** . A pdf file can be read in an Acrobat reader, which you can download free from ***http://www.adobe.com/products/acrobat/readstep2.html***. This file can be read on a computer screen, printed out and read like a book, or printed out in relevant pieces. It is paginated, has a table of contents at the front showing where topics are located by page number, text divided into chapters and sections, and an index at the back for looking up terms and other keywords and phrases, again by page number.

The text is also made available as an **html** (hypertext markup language) document. You are familiar with this from the internet. Most web pages you look at when you surf the internet for information are written in html. The text in this format is not paginated, but it does have a table of contents and index, which are **active** in the sense that the entries are hyperlinked to their references. So if you run across a word or phrase in the text which you don't know or can't figure out from the context, you can go to the index and look it up. If you find it in the index, you just click the link, and you are taken to a place in the text where the term is defined either explicitly or by its context. One can read a well made html document more efficiently than a hardcopy version of the same text. If you want to make notes, you can print out relevant portion of the text and make notes on that.

The third format the text is available in is as a **book of Maple worksheets** . This is the **source document** for the text, that is, any changes to the text are made in the Maple worksheet and then exported to pdf and html. Perhaps this will be the most useful of all the formats as you become familar with it. Maple is a mathematical word processor and high powered graphing calculator. There are other programs, such as Mathematica and Matlab, but we have chosen Maple. You must have ready access to Maple to make much use of this text.

The book of worksheets format has a table of contents/index worksheet which acts a **hub** for your reading; that is, you open this worksheet first, then click on the links in either the table of contents or the index to go to where you want to read and work. While there you can execute cells or make notes right in the worksheet. When you are finished, you can click on the back arrow at the top of the worksheet to go back to the hub. If you have made changes, you are given the opportunity to save them.

2 Ways to make up homework

Making up homework problems is an activity that every math teacher does. Often, it is simply a matter of choosing some problems from a textbook, or a list of problems already made up. But it should include some problems that you have had a hand in constructing, possibly alone or working with a group of teachers. When you make up quizzes and tests for your students, you are engaging in problem creation. It is work, and takes effort. But it is worthwhile and interesting work, especially if you have some interesting ways to make up problems.

2.1 Some ways to solve problems

2.1.1 Polya's four steps to solving a problem

George Polya (1887-1985), a Hungarian mathematician, wrote "How to solve it." for high school students in 1957. Here is his four step method.

Understand the problem: Read the problem over carefully and ask yourself: Do I know the meaning of all the words? What is being asked for? What is given in the problem? Is the given information sufficient (for the solution to be unique)? Is there some inconsistent or superfluous information which is given? By way of checking your understanding, try restating the problem in a different way.

Design a plan for solving the problem: In essence, decide how you are going to work on the problem. This involves making some choices about what strategies to use. Some possible strategies are:

Draw a picture or diagram – making a picture which relates the information given to what is asked for can often lead to a solution.

Make a list – this is a strategy which is especially useful in problems where you need to count the members of a set.

Solve smaller versions of the problem and look for a pattern – almost any problem can be made simpler in some way. By working out simpler versions, you can often see patterns which help solve the original problem.

Decompose the problem – Many problems can be broken into a series of smaller problems. This strategy can turn a problem which on first glance seems intractable into something more doable.

Use variables and write an equation – the method of algebra. Very useful in a lot of problems. See the section below for more details and examples.

Carry out the plan: Spend a reasonable amount of time trying to solve the problem using your plan. If you are not successful, go back to step 2. If you run out of strategies, go back to step 1. If you still don't have any luck, talk the problem over with a classmate.

Look back: After you have a proposed solution, check your solution out. Is it reasonable? Is it unique? Can you see an easier way to solve the problem? Can you generalize the problem?

2.1.2 A method for setting up and solving problems with algebra

A huge number of problems that you or I make up can be solved by setting up and solving a system of equations. Here is a method for setting up and solving such a problem.

Name. Identify the answers (unknowns) needed and give them names: x, y, z etc are good names for unknowns. Identify the given quantities (parameters) and give them names. **Notes:** a, b, d are good names for parameters, but they can be called anything. It is good practice to give names to the given parameters. When you solve the equations you can assign the given values.

Set up equations. Write down all the equations you can think of relating the unknowns and the parameters. **Notes:** You may have to read the problem several times, and use what you know to write down the equations. Generally speaking, if you have at least as many equations as you do unknowns then you can stop.

Solve the equations. Find the mathematical solutions to the system of equations you have written down for the unknowns in terms of the parameters.

Solve the problem. Inspect the solutions you have found and discard the ones which don't answer the original problem.

Check. Verify that the solutions that are left indeed are the answer needed.

2.1.3 Examples

Problem: Given that the area of a rectangle is 40 square feet, and its height is 10 feet, find its width.

Solution: Call its width x. That is the unknown. Call the area A and the length b. These are the parameters.

We know the area of a rectangle is its height times its width, that is, $A = b\,x$. Since we only have one unknown, we can stop and solve the equation for x in terms of A and b: $x = \frac{A}{b}$ feet. For the original problem. x = 40/10 = 4 feet.

Problem: Given that 1 hp pump can fill a tank in 30 minutes and a 1.5 hp pump can fill it in 20 minutes, how long would it take to fill the tank when both pumps are working?

Solution: Let t be the time in minutes it takes for both of them to fill the tank together. So in 1 minute, working together they fill 1/t of the tank. The first motor fills 1/30 of the tank in 1 minute, and the second motor fills 1/20 of the tank. So together

1/t = 1/30 + 1/20 of the tank. Solve for t to get t = 600/50 = 12 minutes. **Note:** If the first motor fills in a minutes and the second in b minutes, then the equation is 1/t = 1/a + 1/b, so t = 1/(1/a+1/b) = a*b/(a+b).

2.2 Some ways to make up problems

Basically, what we are doing here adding a fifth step to Polya's method: **Pose the problem** .

Listed below are some ways that you can use to provide inspriration, motivation, or focus while you are trying to think up problems.

2.2.1 Take a problem you can already solve

This is a way people do research in mathematics. Start with an interesting problem you can solve and try to think up related problems. There are some standard ways to do this. We describe some of them briefly here, and in more detail later.

1. You can **insert a parameter** into the problem. That is, replace one of the given numbers in the problem with a letter, which we regard as the name of an unspecified number. Then see if

you can solve the 'parameterized' problem. Of course, the solution will be stated in terms of the parameter, so it may be an algebraic expression.

Example. Jack has 5 apples and Jill has 3 pears. How many pieces of fruit do they have together? Answer: 5 + 3 or 8 pieces of fruit.

Parameterized problem: Jack has a apples and Jill has 3 pears. How many pieces of fruit do they have together? Answer: a + 3 pieces of fruit.

Solving a parameterized version of a problem exposes the method of solution and leads naturally to other investigations.

2. If you have parameterized a problem, you can investigate the **domains** of the parameters, that is you can ask 'for what values of the parameters does the answer make sense?'. In the above example, we can see that the parameter a should be a non-negative integer.

3. We can pose some of a problem's **inverse problems.** For example, an inverse problem to the fruit problem above is: If Jack has 6 apples and Jill has pears so that together they have 15 pieces of fruit, then how many pears does Jill have?

4. What are some other interesting questions you can ask with the same givens as in the problem?

Virtually any problem will be stated in some context or setting, and so without fail, there will be other questions you can ask in that context.

Example. A famous problem from plane geometry is to show that if three points in the plane are noncollinear, then they are on a circle. What are some other questions you might ask in this setting?

Question. Must any 4 noncollinear but coplanar points lie on a circle?

Solution: This one is easy. The first three points determine a circle. Now the 4th point is probably not going to lie on that circle, and in any case we could pick another 4th point that does not lie on it. So, the answer is, no, 4 random points need not lie on a circle.

Question. Choose 3 noncollinear points at random in the plane. What is the probability that they lie in a semicircle? QED

Solution: This one is not so easy. Call the points A_1, A_2, and A_3 . Call C the circle thru the points. Call C_i the semicircle of C going counterclockwise from A_i for i = 1,2, and 3. Now all three points lie in a semicircle if and only if they lie in exactly one of the C_i's. Since the points were chosen at random, the probability that A_2 and A_3 both lie on C_1 is $\frac{1}{2}\ \frac{1}{2} = \frac{1}{4}$. Since there are 3 mutually exclusive possibilities, we get a probability of $\frac{1}{4} + \frac{1}{4} + \frac{1}{4} = \frac{3}{4}$ that 3 randomly chosen points in the plane lie in a semicircle. QED

If the problem is one you haven't had luck in solving, then by asking and solving related questions, sometimes you can sneak up on the solution to the original problem.

2.2.2 Take a setting, picture or theme

The setting for a problem is there before the problem was posed, so you don't have to start with a problem, you can take an interesting setting and look for problems in that setting.

Quadrille paper.

There is no end to the interesting problems you can pose using a sheet of quadrille paper. For example,

Problem. Make a 2 by 4 rectangular grid, and insert the digits 1 thru 8 in the cells so that no two adjacent cells contain concurrent numbers. **Note:** Two cells that touch on an edge or at a corner are adjacent.

Pyramids

Let B be a region in the xy-plane and let v be a point above the plane. The pyramid P with base B and vertex v is the union of the set of segments connecting v with a point in B. For each edge E of B, the union of the segments connecting v with a point e of E is called the side of P based on E. The edges of P consist of the segments which are edges of one or more sides of B. The vertices of P are defined similarly. There are a ton of questions you could ask about pyramids. For example.

Problem. How many edges will a pyramid P have? What is it's volume? How would you define the interior of a pyramid?

Inscribed figures

If P and Q are convex polygons, and each vertex of P lies on a side of Q, then P is said to be inscribed in Q. Give some examples of pairs of polygons P and Q with P inscribed in Q.

2.2.3 Take a page from a magazine

This is an experiment I have tried. Rip pages at random out of an old magazine and distribute them to your class, one to each student. Give them 15 minutes to come up with two interesting problems motivated by what they have read in their page. You may be pleasantly surprised by what you get, and in any case it is a good exercise for the students.

Have a set of 'standards'

We can make up interesting homework problems which exercise and test a mastery of any set of standards.

2.3 Standards based homework

2.3.1 Everybody has standards

Standards are in. In 1989, the National Council of Teachers of Mathematics (NCTM) issued its first set of national standards for grades K-12. These were such a big hit that they were greatly enlarged and updated in 2000. The NCTM web page at ***http://www.nctm.org/*** has all kinds of materials related to their standards in K-5, Middle, and High school (which is called 9-11 for some reason). The province of Ontario has an extremely detailed set of standards that their students are to be taught to and measured against.

http://www.edu.gov.on.ca/eng/document/curricul/curr97ma/curr97m.html . And, closer to home, we have the Kentucky Dept. of Education core content assessment standards: look for Core Content in this link

http://www.kde.state.ky.us/KDE/Instructional+Resources/default.htm

One of the main goals of this section is to discuss how to develop WHS homework which can be used by teachers to measure (and document that they do so) their students against standards of various kinds.

Course Standards

In our teaching, we speak of 'course standards', by which we mean a list of things we want our students to know, or be able to do, or be able to use at the end of the course. These standards are usually listed in the course syllabus which we give to our students at the beginning of the course.

Course standards can be organized along several lines. Probably the simplest is to organize them temporally according to the order in which they are first addressed in the course. Then the standards can be assigned ordinal numbers to distinguish them from one another.

As an example of a set of course standards, we could cite the ones that are used in Ma 123 Elementary Calculus at the University of Kentucky. There are 40 standards arranged in logical (and thus chronological) order. These are broken into 4 groups for examination purposes. So the student knows in advance which standards will be tested at each exam.

Here are the first 5 standards written in Maple strings for use with WHS homework.

```
> ma123[1]:="1.  Student has an understanding of the relationship
  between the geometry of  lines in the plane and analytic expressions
  (i.e.  equations) describing them.    In particular, student can
> calculate  equations of  lines from point or point-slope, and
  information about  intersection,  parallel and orthogonal
  relationships.  Conversely they must be able to deduce the
> corresponding geometric information from the analytic.":
  ma123[2]:="2.  Student knows the quadratic formula and is able to
  calculate the point(s) of  intersection of two quadratics, a quadratic
> and a line,  line and circle, etc.":
  ma123[3]:="3.  Student is able to  correctly and precisely
  define/describe the derivative of a function at a point in both
> geometric and analytic terms.    They should understand that the
  analytic form represents the  limiting value of the slopes of secant
  lines.":
> ma123[4]:="4.  Student can calculate the derivative of a simple
  function (linear , quadratic, f(x) = a/(x-b) )  at a given point  as a
  limit (i.e.  using the  analytic definition)":
> ma123[5]:="5.  Student knows the standard forms of the equation for a
  circle with a given radius about a given point and should be able to
  find the tangent line to a circle at a point.":
```

2.3.2 A start on a set of standards for a pre-calculus course

It is an interesting and important exercise to make up course standards for the courses we teach.

The student will be able to:

1. use the associative, commutative, and distributive properties of addition and mulitplication to decide if two expressions for numbers obtained by combining other numbers (possibly unspecified) using those operations represent the same number;

2. state the definitions of subtraction and division, and say what the basic properties of these operations are;

3. do item 1 with the operations of subtraction and division thrown in;

4. .....

2.3.3 Project: A sample of what is to come

Let's make a start on a set of standards for a course we teach (Algebra I?) Break up into teams of 3 or 4. Choose a name for your team. One member of the team will put in the team answers

to the question below, either into a Maple worksheet or a Word document. This member will post the homework referred to in exercise 7 below, and toggle downloading.

1. List 5 standards against which you want to measure your students at the end of the course.

2. Now arrange these standards in order of importance. (This may or may not have been in the order you listed them.)

3. Now arrange these standards in a logical order. (standard B is listed after standard A if in order to satisfy B, the student needs to satisfy A)

4. Make up 3 problems which you would give to a student to measure them against the standards. One of the problems should have a diagram which helps to understand the problem.

5. For each problem, which of your standards does it measure?

6. Parameterize each problem with at least one parameter

7. Compose a WHS homework with title 'Standards Based Homework' of multiplicity 4 using your 3 problems, and post it.

8. Each member of the team will run a check of the homework, looking for mathematical, typographical or grammatical errors in the statement of the problem, and mathematical, or syntactical errors in the answers to the problems..

3 Introduction to homework preparation for WHS

3.1 WHS tags: A brief description with examples

The source. Login to WHS, go to Teacher Resources, and click on Authoring Tools. At the bottom of the page, click on Author Documents and print it out. This gives a succinct description of all the WHS tags, including the answer format tags. Below are several examples of problems which have been composed by hand using the tags. This can be done in any text editing program such as word 2000, edit, etc.

Homework Assignments from Maple Worksheets

Input files for WHS are most conveniently made using the computer algebra system **Maple** (current version 9.5). This means that the author can use the computational power of Maple to check results and draw plots for the assignment as well as the high level worksheet interface to format mathematical formulas. One intersperses these supporting computations with the input to be read by WHS. The Maple .mws file is then written out as HTML and .gif image files; these are made into a .zip file which is uploaded to WHS and converted into the .xml file which is actually read by WHS.

In order for the worksheet to be usable by WHS, it must have the format which we are about to describe. Note that the current implementation of WHS should accept any worksheet input in the specified format. It does, in fact, accept certain no longer supported formats in the interest of backwards compatibility.

The lines of text in the spreadsheet can be divided into two classes:

Control lines are instructions to WHS. They specify such things as the beginning of a question, the location of an input widget such as a text box, and the beginning of a part of the worksheet that should be ignored by WHS. Control lines have a control entity, strings of a particular form which always contains an underscore character. The precise format of these entities

are described below. But, it is recommended that worksheet authors should simply avoid use of the underscore character entirely except in control lines. Other lines are simply referred to as normal text lines**.**

Input Ignored by WHS

Each control line together with all the normal text lines immediately following it and appearing before the next control line make up a **section** of the input file. The normal text lines appearing before the first control line are also said to form a section.

The following sections are ignored by WHS:

All sections prior to the first control line containing H_, T_ or Q_ are ignored.

All sections with a control line containing SKIP_ and all immediately following sections with control lines not containing T_ or Q_ are ignored by WHS.

3.1.1 The header Tag and section

If the first section not ignored by WHS has a control line containing **H_[x]** where x is a positive integer or H_, then the text of the section is the global header. It is output at the beginning of the Homework assignment. Typically, it is used for a title. If the [x] is specified, then x is the **multiplicity** , which defaults to the value 1. If the multiplicity is greater than 1, then out of each group of x questions, only one randomly selected one is included in the displayed assignment. Each time the assignment is displayed, different selections of problems are obtained.

In more detail, when an assignment is requested, one can specify a numerical version number. There are several possibilities:

If the version is -1, one gets the Common version. If the multiplicity is m, then this consists of questions whose numbers are 1 mod m.

If one requests the Personal version, then one gets the version whose version number is equal to the account id of the user.

One can specify a Specific version by giving the version number.

One can specify a Specific version 0 or simply leave the version number field blank. This causes a random version to be selected.

Versions other than the Common version are made up by pseudo random selection of one problem from the last m - 1 of the problems in each group of m contiguous problems.

The parameters between the square brackets are semicolon separated. The first is the multiplicity as defined above. The second argument will not be addressed here; simply leave it blank. By putting a third parameter called **base** , you can cause the randomization to start after question number base, and for the Question XXX not to be displayed. This was a hack for KEMTP. Experimental feature: if you set this parameter to -1 and if the multiplicity is greater than 1, then the problems selected for a version will appear in the same order as in the Maple or Word document; if you need to have the original functionality of base, then replace it with -1 - base, e.g. base = 1 gives a new value base of -2 if you want no scrambling.

By putting a fourth parameter called **header** , one can vary the logo image at the top of the page. The value of the parameter is a filename; the file itself must be located in the html subdirectory of the main wqs directory of the server.

Finally, by putting a fifth parameter called **hwFlags** , one can affect how the homework xml is generated. If the value of hwFlags is 1, then video links on the client machine will be to files in the root directory. If you do not want this, then use 0 or no value for this parameter.

Header examples:

1. A homework with a 56 question database and a header H_[5;;6;0;1] will have 16 questions in each retrieved homework. The first 6 will be unnumbered and will be the first 6 questions in the database. The remaining 10 questions will be numbered and will be the first question in each group of 5 questions in the database counting from the 7th question in the database, if the version is -1. For other version numbers, the questions are chosen from the last 4 in each group of 5 counting from the 7th. Any videos that are associated with the homework are accessible from the top of any drive they are stored on.

2. If the header in the above setting is changed to H_[5;;-7;0;1], the only thing that changes is that the problems occur in the same order they are listed in the database.

3.1.2 The Leader Tag and Question tag

The remaining sections not ignored by WHS can be grouped into questions. Each question consists of :

An optional leader section. The control line contains **T_** . The normal text is used as a lead-in to the question. In extreme cases, it can be an entire lecture of material before the first question.

A mandatory question section whose control line contains QM_ or QN_

A series of answer sections. There should be at least one in a QM_ question. The control line is identified by the string Ax_ where x is one of the upper case letters C, F, H, L, R, and S. The letter specifies particular answer formats. In addition, for QN_ questions, one can use X.

Question Control Line Formats

Questions whose answers are not to be graded by WHS are **QN_ questions** ; those whose answers are to be graded by WHS are

QM_ questions . Students are given the opportunity to submit comments about QM_ questions only if they submit a proposed answer; these comments are sent via e-mail to their instructors.

If the character immediately following the underscore is an upper case N, then the correct answer is not displayed when the student submits an incorrect answer to the question. Otherwise, the correct answer is displayed when an incorrect answer is submitted. The correct answer is always displayed when a correct answer is submitted.

Correct answer specification occurs immediately after the underscore character (or the letter N following the underscore character. It should occur for both QM_ and QN_ questions, even though the answers are ignored in the case of QN_ questions. The specification is enclosed in square brackets and consists of a sequence of at least two quantities. The quantities are semicolon separated. The first quantity is the error bound; its use will be described below. The remaining quantities are the correct answers for each of the input fields in the answer section. The format of these will also be described below.

Note that checking of answers is done with an .asp script interpreted by a VBasic interpreter. This has the unfortunate consequence that expressions are evaluated using vbasic's non-standard operator precedence in which unary minus is of higher precedence than exponentiation. For example, -3^2 is interpreted as 9 and not -9. It is recommended that one reduce the likelihood of misinterpreted results either parenthesizing expressions or using numerical answers whenever possible.

The correct answer specification of QM_ questions can be followed by an additional square

bracket enclosed semicolon separated list of non-empty description strings. These strings should be case insensitive and contain neither square bracket characters nor single or double quote characters or semicolons. The intention is that these strings will be the names of standard skills being tested by the question. The list of the strings together with the question numbers where they are listed is stored in the homework descriptor. The homework results pages show numbers of problems correctly answered for each description string.

3.1.3 The header answer tag

The various answer types correspond to different ways to input or to check answers. In order for a question to be marked correct, all the input answers must be correct.

Header Answers: These have **AH_** in the control line. A single parameter with value 0, 1, 2, or 3 can be specified by adding [x] after the AH_. The normal text lines are output and, if the parameter is specified and has value 1 or 3, then it is followed by a line break, i.e.
. If the value is 2 or 3, then it is preceded by a line break. There is no (correct) answer for a header answer section.

Header answers can also be used to put in a video link. In this case the first parameter is as before, the second parameter is the name of the video file, and the third parameter is a label. The body of the header will be displayed followed on the right with a pull down menu and a button with the label specified as the third parameter. The pulldown menu allows the user to choose a disk drive or more typically a cd rom drive; in this case the video file is assumed to be located in a subdirectory of the root directory; the name of this subdirectory should be the homework identifier. The default value of the drive letter is So, a typical example would be

AH_[0;myvideo.wmv;Show Video] For a video about this this or a similar problem:

simply a blank; this causes the file to be retrieved from the WHS video server. **Note:** if the Homework header has a fifth argument of 1, then the video is assumed to be in the top directory of the drive letter chosen (so, if the video bill.wmv is stored on a CD which is inserted into the L drive, then WHS will look in L://video.wmv for the video.

3.1.4 Constant Answers:

These have **AC_[x]** in their control line where x is positive integer indicating the length in characters of the input text box. The input text box is followed by the normal text lines of the section. The student should input a single number expressed in 'calculator syntax'. This means an expression with explicit asterisks for multiplication and can include pi for 3.14159..., e for 2.718..., and elementary function calls such as ln(...), exp(...), sin(...), etc. The correct answer should have the same format. The submitted answer is said to be correct if its value differs in absolute value from the value of the correct answer by no more than the error bound.

Example:
QM_[0;4]
What is 1 + 3?
AC_[5]

SKIP_

3.1.5 Word Answers:

These have the same format as Constant Answers except that you use **AW_[x]** instead of AC_[x]. The answer is a sharp character separated list of strings. The submitted answer is said to be correct if at least one of these strings is the same string as the answer. Both answer and submitted string are trimmed of leading and trailing whitespace. The comparison is case sensitive.

Example.

QM_[0;triangle#Triangle#TRIANGLE]

If A, B, and C are noncollinear points, then the union of the three line segments AB, BC, and CA is called a

AW_[8]

.

SKIP_

3.1.6 Table Answers:

Constant answers can also be input as entries in tabular arrays. To do so, use **AT_** [....] control lines where the semicolon separated parameters are: The size S of the textboxes, in characters. The number R of rows in the table The number C of columns in the table The actual entries in the table – there should be R times C of these entries. Any entry which is empty is replaced by a textbox of length S. The correct answers are specified as several entries in the list of correct answers in the QM_[...] control line for the question. The answers occur in row major order.

Example:

QM_[0;12;15]

If f(x) = x + 3, then complete the table of function values.

AT_[4;3;2;x;f(x);9;;12;]

SKIP_

3.1.7 Function Answers:

These have **AF_[x;t;n;a;b]** in their control line. The parameters are semicolon separated. The integer parameter x is the size in characters of the input text box. The t parameter is the variable of the function used in both the correct answer and in the input answer. When the answer is checked, n random values of this variable are chosen from the half closed interval [a,b); the functional values of the correct and submitted answers are calculated, and the absolute values of the difference of these values must all be no larger than the error bound in order for the answer to be considered correct.

Example:

QM_[0;x+12]

AH_[0]

If f(x) is a linear function and f(9)=12, and f(12)=15, then f(x) =

AF_[10;x;2;0;1]

SKIP_

3.1.8 Extended Function Answers:

These are like function answers but allow for multiple variables. These answers have **AE_[x;m;x1;...;xm;n;lo1;hi** in their control line. The parameters are semicolon separated. The integer parameter x is the size in characters of the input text box. The parameter m is the number of variables, and the variable names expected in the submitted answer are x1, x2, ..., xm. When the answer is checked, n random m-tuples will be generated where the jth coordinate lies in the range of [loj, hij) and is used for the variaable xj. The functional values of the correct and submitted answers are calculated, and the absolute values of the difference of these values must all be no larger than the error bound in order for the answer to be considered correct.

Example:

QM_[0;x+2*y]

AH_[0]
If f is a linear function of two variables, with f(0,0)=1 and f(1,0)=2 then f(x,y) =
AE_[10;2;x;y;2;0;1;0;1]

SKIP_

3.1.9 Integral Answers:

These have exactly the same format as Function answers except that they use **AI_** in place of AF_. When integral answers are checked, the functions are checked for equality up to a constant summand. The test is done with n - 1 randomly selected points in (a,b).

Example:

QM_[0;xˆ2+2*x]

AH_[0]
Give an antiderivative for 2*x + 2. Use calculator notation.
Answer =
AI_[10;x;2;0;1]

SKIP_

3.1.10 Selection Answers:

These have **AS_** [val1;val2;...;valk] in their control line. The input is a pull down menu with val1, val2, ..., valk as the menu choices plus the possibility of a blank answer. The possible answers are semicolon separated strings. The blank answer is originally displayed; an answer is selected if the display is any of the other values. The correct answer must be one of the strings val1, val2, ..., valk. The normal text lines are displayed after the pull down menu. The strings val1, val2, ..., valk should not contain either semicolons or sharp characters. Example:

QM_[0;8]

AH_[0]
If f(x) = x + 3, then the value of f at x = 5 is
AS_[8;2;6;7]

SKIP_

3.1.11 Horizontal Line of Radio Button Answers:

These have **AL_** [val1;val2;...;valk] in their control line. The possible answers are the semicolon separated strings val1, val2, ..., valk. The correct answer must be one of the strings val1, val2, ..., valk. The radio buttons are displayed in a centered single row table with the corresponding answer string to the right of the radio button. The strings val1, val2, etc. should not contain either semicolons or sharp characters.

Example:
QM_[0;13]

If f(x) = 2*x + 3, then the value of f at x = 5 is
AL_[13;8;10;11]

SKIP_

3.1.12 Horizontal Line of Checkbox Answers:

these have **AB_** [val1;val2;...;valk] in their control line. The interface generates a one line centered table of checkboxes labeled by val1, val2, ..., valk. The correct answer string should have k semicolon separated answers which are either vali or the empty string depending on whether or not the i-th checkbox should be checked in order for the result to be counted right. Because checkboxes are two-state, one cannot distinguish between the answer not being given and not having any of the boxes checked. So, the author should always word the question in such a way that the student should have at least one box checked in order for the question to be answered – e.g. one of the boxes could have the value 'none' and the others could have numbers for various possibilities which might be true.

Example:
QM_[0;;2;;;10/5]

If f(x) = x + 3, then the value of f at x = 5 is
AB_[8;2;6;7;10/5]

SKIP_

3.1.13 Vertical Row of Radio Button Answers:

This is used when the value of the button needs a longer description. Each radio button is described by its own answer section, whose control line is of the form **AR_[x]** where x is a small integer used to identify the radio button group. A group consists of contiguous AR sections with the same group identifier – they act in unison, in that only one of the group may be selected at once. The entire group of radio buttons is indented as a <dir> group. After each radio button, the section's normal text lines are inserted. The correct answer is an integer corresponding to the number of the radio button in its group (counting from 1). When the submitted answer is displayed on the feedback page, it is specified in the same way. Because some authors prefer to list the correct answer first in the list, the first answer is re-inserted in a random location in the group. **Example:**

QM_[0;2]
Which is the correct definition of parallel lines in space?
AR_[1]

Two lines whose intersection is empty.
AR_[1]
Two coplanar lines whose intersection is empty.
AR_[1]
Neither of the others

SKIP_

3.1.14 Multiline Textual Answers:

These are used only with QN_ type questions. The **AX_[x]** where x is a small positive integer causes a textarea element to appear; it has x rows and 40 columns. Typically, you will want to proceed it with a line break, which can be accomplished with an AH_[3] section before it. Since they only occur in QN_ questions, there is no grading of AX_ answers. **Example:**

QN_[0;to be hand-graded]
Discuss the universe and give 3 examples.
AX_[3]

SKIP_

QN questions can be hand-graded. To do so, the instructor goes to Access Records for the class and then clicks on Assignment submission. For the particular assignment, one can click on Display and then click for each student to display submission of the student. There is a Right/Wrong box for each problem to be hand graded.

3.2 Creating a homework using maple.

How do we create a homework? The main tool is the word zipit from the MCtools package. It has a help page: access it by typeing

```
> mctools(zipit);
```

Now the steps to creating the homework are:

1 If necessary, Get the most recent MCtools section from
https://www.msc.uky.edu/carl/communicating_math/MCtools_page.htm
and copy the section into the top of your homework. Or, if you are starting fresh, just use the worksheet that MCtools comes in as your homework worksheet. It has a Kentucky standards section, a tagit examples section, and a start on a homework that you can modify. Note: If you start with a fresh worksheet, you should merge the styles in the MCtools worksheet you downloaded into your worksheet. Click on the Format menu at the top of the worksheet, then click on Merge existing ..., then browse for the downloaded MCtools worksheet (save it somework if you haven't), select it, click on open, then click on done.

2. Open the MCtools section and execute it, then close the section (This is something you do each time you start a Maple session.

3. Make a title for the homework. This involves putting some text in the title section.

4. Create a folder to save the homework in and save it there. You can do this from Maple from the File Menu.

a) Click Save As, then browse for a place to make your homework folder. c://temp will do fine. b) Click the Create new folder button (just to the right of the yellow up arrow); a New Folder

appears. Rename it with an appropriate name. I usually give the folder the name of the homework, which I keep short and simple, like hw1

c) Now select the folder you just named and go into it by double-clicking your right mouse d) Now type in the file name line the name of the homework (so if you named the folder hw1 then name the homework hw1) and click on the save button. Maple will save the worksheet in the directory hw1 with the name hw1.mws.

5. Create the correct zipit line for this homework. Modify the input line starting with zipit at the top of the worksheet so that the first argument is the homework worksheet name and the second argument is the correct path to your homework folder. For example, if your homework worksheet is called hw1.mws and the path is C://temp/hw1, then your zipit line would look like this **zipit("hw1","C://temp/hw1");**

Don't execute this just yet (wait until you have created the problems and exported to html).

6. Make the homework problems. Put problems in their own sections so that you can eliminate them from the homework easily by just closing off the section. All work pertaining to the problem can be put in this section. Put in just one or two problems to get started, and post the homework just to see everything is working. Before you get the homework like you want it, you will go through the editing cycle (**Save Export Zipit Install and Preview**) several times.

7. Save the worksheet again and export it to html. To save it again, just click on File, then click on Save. The changes you have made will be saved. To export to html, click on File, then click on Export As, then click on HTML. Accept all suggestions Maple makes.

8. Now execute the zipit line. Put your cursor in the input cell and press enter. A zipfile named hw1whs.zip (if your homework was called hw1) is created and placed just above the homework folder hw1. **Important:** Note that the zipfile's name hw1whs is not the same as the worksheet's name hw1. The reason is that zipit creates the html file hw1whs.html in the folder which is the file to be used by WHS to create the homework. Unless you tell it otherwise, zipit will always create a zipfile whose name is the name of your worksheet

Note: If you get an error message saying you don't have the zip program on your system, you can get the (free) info-zip from the MCtoolss Page at

or from the Info-zip Home page. Put it in C://Winnt or C://Windows. There is an info-zip unzip program, but in XP you can extract the files in a zipfile with the window explorer, and in windows 2000 you can use 7-zip or winzip to unzip a zipfile.

9. You are ready to install the new homework zipfile on WHS

3.3 Using the export to html with Mathml option in Maple

With the MCtools of June 2004 or later, you can choose to export your files to **html with Mathml** . WHS can handle this format now. Its principle advantage is that formatted mathematics is exported in a scalable form. Its principle drawback is that current browsers do not display it properly unless a plugin is installed first. IE6 requires Mathplayer (free from http://www.dessci.com/en/products/mathplaye So homeworks installed this way should have a notice in the homework header telling the students that they may need to install mathplayer before they can view the homework properly.

3.4 Installing a homework zipfile

1. Log into your WHS account, using Internet Explorer.

2. Highlight the **Teacher Resources** menu item and click on the **Authoring Tools** Button.

3 Go to the Homework Installation table and press the **Browse** button. Use the window that opens up to locate the homework zipfile hw1whs.zip on your computer (Remember, it is just above the directory hw1whs) The icon for the zipfile has a little zipper on it to distinguish it as a zipfile.

4. Highlight the zipfile and click it to make it appear in the Open line in the browse window.

5. Press the **Open Button** in the browse window. The browse window disappears, and the path to the zipfile appears in the upload line of the bottom table.

6. Press the **Install and test homework** button, and wait. After a few seconds, the zipfile is uploaded, installed, and the homework is retrieved for you to preview.

3.5 The edit cycle

Once you have a homework installed, you will find that you need to make some changes. You might want to correct an error in the homework, or add a problem, or delete a problem.

If you are working on the computer from which you made the last installation of the homework, all you have to do is open the source worksheet (unless it is already open, because you just installed the homework), make the changes, then go through the **edit cycle** : **S**ave, **E**xport, **Z**ipit, **I**nstall The first 3 steps take place in the Maple worksheet, and the last 2 take place in Internet Explorer. **Keep both applications open as you edit.** That way, you can go more easily through the cycle. Occasionally, you will have to re-authenticate your login to mathclass, if you spend more than a few minutes making changes in the source worksheet.

If you are working on a different computer (but one that has Maple and Internet Explorer), then you can login to mathclass, go to **Teacher Resources**, then click on **Authoring Tools,** then find your homework in the **Homework Testing Table** and click on **Download.** If you get the message saying you don't have permission to download, go to the **Homework Installation Table** and find your homework on the line with **Toggle Downloading,** which you should then click. Now go back and download your homework zipfile and open it up with Winzip or 7zip and click on the source worksheet in there. Maple will open the worksheet. Create a folder to put the worksheet in and go through the edit cycle. **Note:** you will need to modify the zipit line to reflect where your worksheet is on the local machine.

3.6 Homework Housekeeping

Here are some algorithms for common tasks that WHS authors are confronted with. If you discover something that is missing or wrong, please report it to the authors.

3.6.1 What to do if you are having trouble gettting a homework to install

Sometimes, when you try to install a homework, you cannot get it to install at all. What to do?

Go through the steps to install it again. If you still get the error message, try this:

1 Close Maple

2 From the Start Button, go to My Computer and locate the homework zipfile. Delete it. Then go down into the homework directory and double click to go down into it. Then select the html files in the directory and delete them. (don't delete the source worksheet.) Also delete the **images** folder you see in the directory.

3. Now start Maple up again and open the source worksheet.

4. Open the MCtools section, execute it, and close it.

5. export the worksheet to html

6. execute the zipit line.

7. Log into WHS and try to install the homework. If you still cannot get it to install, email the zipfile to me and I'll try to install it.

3.6.2 How to modify a homework .

If you already have the directory containing the source worksheet on your computer,

1. Open up the source worksheet for the homework.

2. Open the MCtools section at the top, execute it, and close it.

3. Make the modifications.

4 Save the worksheet.

5 Export the worksheet to html

6. Execute the zipit line.

7. Upload the homework zipfile to WHS (Note: Don't exit Maple here.)

8. Install and preview the homework.

9. If there is some additional change to make, go to the source worksheet which you left open and proceed from step 3 above.

This is the editing cycle, which you will perform several times before the homework is just like you want it.

3.6.3 How to download and unzip a homework zipfile from WHS.

You can download homework zipfiles that you have previously installed on WHS. Also, you can download a homework that some other author has installed, provided that author gives permission.

1 First open your browser (Internet Explorer) and log into WHS (if you haven't already done so).

2 Download *the homework zipfile:*

Highlight the **Teacher resources** Menu item and click the **Authoring Tools** button.

In the top table, select a homework list containing the homework you want and press the **Set Menus** button.

In the second row of the top table, select the homework you want (suppose it is called homeworkwhs) to download and press the download button.

A window pops up. (or you are told the author hasn't given permission for downloading. In that case, you will need to contact the author and ask him or her for permission to download. If you are the author, then you can go to the Homework Installation table at the bottom of Authoring Tools and toggle the **Download** button for that homework.) Choose to save the file, which is called homeworkwhs.zip. Save it in a convenient place, say c://temp

3 Now Unzip the homework file:

In the lab, you are using Windows XP, so you can click **My Computer** from the Start Button and find the zipfile you saved. Then rightclick the zipfile and choose the option **Extract All.** The zipfile will be unzipped into the folder homeworkwhs c://temp/homeworkwhs. If you are at home, and are working with Windows 2000, you can use 7-zip to unzip the zipfile.

Nota Bene: Usually, all you want when you download a homework zipfile is the Maple worksheet that it contains. It is the source, and with it you can regenerate the html export. So, often rather than saving the homework zipfile as described above, you can open it up (with 7-zip if you are using Windows 2000) and just click on the Maple worksheet in there to open it in Maple. Then you can save the worksheet

3.6.4 How to delete a homework that you have installed.

If you have a homework that you want to remove, just do this:

1 From the Homework Installation table at the bottom of the Authoring Tools page, select the homework you want to remove.

2. Click the **remove this homework** button.

3. You will be asked to verify that you wish to remove the homework. After checking that you are removing the correct homework, click ok.

3.6.5 How to create a homework list and add homeworks to it.

A homework list contains a sequence of homeworks. Every class (see below on how to create a class) must have a homework list.

1. Click **Authoring Tools**, and go to the middle of the **Homework List Management** table.

2. Type in a name for the List and then click the **Create Homework List** button. This creates an empty homework list.

3. Now at the bottom of the table click the toggle **Publication flag** button. This will allow you and others to use your homework list in a class.

4. You can add your homeworks or homeworks from other authors to the list.

3.6.6 How to create a class in WHS

A class in WHS is the vehicle by which you make your homework list available to your students.

1. Login to mathclass **2**. Go to **Teacher Resources** -> **Manage Classes** : A page will appear with **Class management** at the top.

3 In the **Create a Class** table, choose a name for your class:

4. Select a school. If your school is not in the list, you can put the class in any school

5. Don't bother with the homepage part. You can reconfigure the class and install a homepage later.

6. Select your homework list (or any homework list will do for the moment.).

7. Ignore the box to put a version. That is a relic from past days.

8. Now click the **Create Class** button. You have created your class. Now students can request registration in your class.

3.6.7 How to approve a students request for registration in your class

1. Login to mathclass

2. Go to **Teacher Resources** -> **Access Records**: A page will appear with 2 columns of buttons. Click the button in the 2nd row 2nd column that says **Registration and Passwords**.

3. You will see a button called **Approve for Registration**. Highlight the students name and click the button. The student will appear on class roll after you refresh the browser.

Note: You can remove a student from your roll at any time. You can also change a students password at any time, unless the student has teacher standing in WHS.

3.6.8 How to check student records

1. Login to mathclass

2. Go to **Teacher Resources** -> **Access Records**: A page will appear with 2 columns of buttons.

3 To check comments click the button **Student Comments.** This collects comments which students make in the feed back sheet. You also receive email copies, to ensure that you get the comment.

4. To check performance on homeworks, click the button **Assignment Submissions** button (to get collated class results, including comma separated files of individual responses on each homework) or the button **Homework Standards Download** (to get a comma separated file of all the standards measurements on each student.), or finally and most often the button **Summaries by student** (to get a summary of each students homework results on each homework and calculate selected averages.)

4. To check student uploads (not frequently used yet), click the **Student uploads** button.

3.6.9 How to use the Class Downloads button

If you want to provide a document, worksheet, etc to the students in your class, you can use the Class Downloads Button. Go to **Teacher Resources** -> **Access Records**: The button in the fourth row second column is labelled Class Downloads. Click on that: a browse window opens. Write in a short description of the file for you students to identify it by. Browse for the file you want to upload to the class and click open with it highlighted. The window disappears and the filename appears above the upload file button. Click on that, and the file is uploaded to the class. You can verify this by going back to the File Uploads Page and refreshing the browser. If you make a mistake and upload the wrong file, you can delete the file with the delete button.

Your students (the ones you have approved for registration) can download the file by going to the bottom of the Announcements page. They will see a Class downloads button. Clicking that will take them to a page listing the files you have uploaded. Note: If you are a student in your own class, you can use this same place to upload the files.

This provides a secure method of providing files for your students. It is more convenient and less public than putting a link on your webpage.

3.6.10 How to install and maintain a webpage for a class.

Each class comes with a place to put a webpage. It provides a place you can put your syllabus and other information relavant to the class. The students and anyone else with the URL can access it without logging into mathclass, so you can put announcements there as well as on the announcement section of Access Records.

To start:

1. Use Word 2000 to create a web page. Start with something simple.

2. Create a folder, say c://temp/webpage and save your webpage in this folder. Name it default.doc

3. Also save default.doc as type webpage in the same folder. This will create a file default.htm and a subfolder of webpage, default_files, which contains any images you have put into the webpage.

4. Make the zipfile. If you are in Windows XP, compress the folder from **Start** –> **My Computer** by locating the folder, rightclicking its icon and choosing **Send to Compressed file.** This will make a zipfile with the same name as the folder, which is ok. The name of a webpage zipfile is not important, just the name of the webpage, which should be default.htm. If you are in Windows 2000, use 7-zip or Winzip to zip the folder. Alternatively, you can open a command window, cd to c://temp and issue the command zip -r webpage webpage. This assumes you have zip in your path. If you don't, you can get it from ***www.info-zip.org***

To install the webpage in your class:

1. Login to mathclass

2. Go to **Teacher Resources** -> **Manage Classes** : A page will appear with **Class Management** at the top

3. Choose the class you want to Configure: A page will appear with **Configure Classes** at the top.

4. Scroll down to the **Upload webpage zip archive** portion and click **Browse**

5. A Browse window opens up. Search for the zipfile webpage.zip in C:\temp

6. Click **Open** . The browse window will close and the name webpage.zip will appear in the Upload line.

7. Then click **Upload Zipfile** You will get a message saying the file installed and the webpage can be found at the url **www://mathclass.org/Homepages/xxxx** where xxxx is the identifier for your class.

8. Copy that website url into **Homepage** line in the **Configure Class table** at the top. Then when you check announcements in the class, you can go to the class webpage from there. Of course, you can also go directly to the class web page.

To maintain the webpage in your class: If you want to add some links or assignments to the class webpage, you need to modify it and zip it up again.

1. You can download your webpage zipfile and go through 7 steps at the top.

2. Or if you have access to the files you used to make the webpage, just go through steps 2-7 at the top.

After you have made a new webpage zipfile, go through the 8 step procedure to install the webpage in your class.

3.6.11 How to find out your Magic number

If you want to have hidden sections in your homework that are stored in the homework zipfile, then you will need to supply zipit with your accId, which we are calling your magic number. Here's how to get yours. Login to WHS, then bring up the html souce in your browser. On Internet Explorer 6, the source is accessed via the View Menu. You'll see the word source about halfway down. Click that and you will get a page of gobbletygook. About halfway down the page, you will see the line var accId = ###

where ### is some number. That is your **Magic number** . When you have a homework that has a hidden section that you have pasted into a WHS problem or in the Header sec-

tion, or one that you have created using addsectionhint, make sure that you include the option Magic=### (where ### is your magic number) in the zipit line. So for example, if you have a homework c://temp/stuff with hidden sections and you want to zip it up, then you would use zipit("stuff","c://temp/stuff",Magic=###);

3.6.12 How to construct WHS homeworks with Word 2000

WHS homeworks can be constructed with Word2000. If you are used to using tagit in MCtools to format the problems, you will need to review the question and answer format tags because these have to be inserted by hand in a Word homework. Also, you will need to zip the files by hand. Note: It is important to save the file as a filtered web page. This creates an htm file which WHS can process into an xml file.

The steps in creating and installing a homework using Word2000 (in WindowsXP, or Windows2000) are: **1** Open a new word document. **2**. Type in one or two of the problems, beginning with a header tag. **3**. From the 'save as' menu, create a directory with a suitable name (say wordsample) and save the homework as a filtered web page in that directory, giving the file the same name as you gave the directory. Later if you want to make changes, you can open this with Word and use it as a source document. **4**. Now, if you are in XP (as you are in CB 313 or Mathskeller), go to My Computer from the Start Menu, then browse for the directory wordsample that you have saved wordsample in and right-click it. A window will open up. Choose Send to Compressed Zip folder. A zipfile wordsample.zip will appear (it will look like the wordsample directory, except it will have a zipper on it.) If you are in Windows 2000 or Windows 98, you can use Winzip or 7-Zip. These are programs which you can install that you can use to zip and unzip folders. 7-zip is free to install on your home computer. It can be downloaded from http://www.7-zip.org/download.html Winzip has an evaluation version which you can download at http://www.winzip.com/ddchomea.htm **5**. Log into WHS and install the wordsample.zip zipfile as usual. **6** Now go back to Word2000 and insert the rest of the problems in your homework, then resave (as a filtered web page), rezip and reinstall, until you have it like you want it.

4 Maple Editing

The conventional introduction to a mathematics problem solving language such as Maple is a demonstration of its ability to execute most of the common algebraic and graphing operations one encounters in algra, calculus, linear algebra, etc. Here, however, we begin by thinking of Maple as a word processor which has the ability to format mathematics. The basic "point and click" editing tools on the toolbar will, for the most part, be familiar to users of editing.

In this section will introduce most of the basic editing tools you will need to get starting using a Maple worksheet as a **source homework document.** The Help button at the top of the worksheet gives access to a good deal more. For example, select **Help: Glossary** to get the Glossary help page. At the top, you will see the Help graph. Choose Basic Features : Document Processing to get more information.

4.1 Entering Commands

A new or blank worksheet will have a single execution line (also called a "group", "input cell", or "paragraph") with the Maple "command line" prompt ">" in the leftmost position - as in the following.

```
> text entered in a command line will usually be red.  However Maple
  cannot parse random text and when one finally hits the "ENTER" key  on
  this line the result will be
```

```
Error, missing operator or `;`
```

Terminating the above line with a semicolon will evoke the same response since the line is not a valid Maple expression or command. Only valid Maple commands or instructions are entered into **execution cells** at the ">" prompt and they must be terminated by a **semicolon** ";" or **colon** ":" For instance "(2*3 -1)/100" is a valid Maple expression so if we enter this in a command line the result is (almost) more expected.

```
> (2*3 -1)/100
```

```
Warning, premature end of input
```

Maple uses a semicolon to inform the parser that it has come to the end of a complete expression and execute to that point.

```
> (2*3 -1)/100;
```

$$\frac{1}{20}$$

As one might infer even from this one example, if you know how to use a calculator then you already know a significant amount since the basic "calculator syntax" for arithmetic is contained in Maple. The principal difference is that multiplication is Maple is never implicit. That is if you mean "2 times x" you must write 2*x rather than 2x.

When you complete a command in Maple you terminate it with a semicolon ";" if you want to see the result or a colon ":" if you don't. The "enter" key then tells Maple to preform the instructions. If the command is in the proper syntax, terminated by a semicolon Maple will return the results of the calculation and open a new command line. If there is a problem it will return an error message.

Here are some examples:

```
> 1+1;
```

$$2$$

```
> 1+1:
> 2*3;
```

$$6$$

```
> 2*x;
```

$$2\,x$$

```
> 2x;
Error, missing operator or `;`
> (1+x)*2;
```

$$2 + 2\,x$$

```
> (1+x)*(x+y);
```

$$(1 + x)\,(x + y)$$

```
> (1-x^n)/(1-x);
```

$$\frac{1 - x^n}{1 - x}$$

Exercise: In the input cell just below, compute the 10 factorial (the product of the first 10 positive integers).

Shift Enter . Sometimes, you want to start a new line in an input cell. You can't just press the enter key, because that will execute the input in the cell. However, you can hold the shift key down while your press the enter key. This will move the cursor to the next line in the cell. Note: the difference between a hard return (Enter key) and a **soft return** (Shift Enter) can also be seen by making the invisible characters visible. From the View menu, select Show invisible characters.

4.2 Manipulation of Input Cells and Text Cells

A line that does not begin with a command prompt ">" or is not a continuation of one that does is a **text line .** Text entered into a text line is simply recorded as if one were typing in a (primitive) word processor. Both text and input lines wrap ito the next line f you keep typing. A "return" in a text line opens a new line immediately below the current one and in the same cell. Group boundaries are defined by elongated brackets "[" at the left of a group of the text and/or commands. These can be toggled on or off in the View Menu or by pressing F9.

You can convert an input cell to a text cell by: (1) pressing Control K , or (2) selecting Insert->Text, or (3) by pressing the "T" button on the main menu. An "ENTER" at this point will not result in an error message.

In constructing a document one usually wants to keep text and commands in separate cells which we will refer to as "command" (or "execution") cells (or groups) and "text" cells (or groups).

The simplest way to start a text cell is to create an imput cell and immediately covert to text mode. This will cause the command prompt ">" to disappear.

One creates an input cell by: (1) pressing Control J, or (2) selecting Insert->Selection Group-

>Below the cursor or (3) pressing the ">" button on the main menu. Each of these will result in a new execution group below the cell in which the cursor currently resides and regardless of whether that cell is a text or execution cell. To create an input cell above the cell containing the cursor, press Control K.

The following indicate what is probably the quickest way to open a new text cell immediately below the cell containg the cursor: simply press the ">" symbol on the toolbar and then immediately press the "T" symbol. Here the cursor is in a text cell but the result would be the same were it in an input cell.

The Insert menu also provides commands for inserting command cells before and after the cursor. These can also be accomplished with the keystroke combinations **CTRL K** and CTRL J, respectively.

Exercise: Create an input cell just above this text cell. Then use the Edit menu to discover how to delete the cell.

Undoing an action . Fortunately, you can undo most actions you do in a worksheet. Select Edit then Undo. In fact, you can back up several steps, a very useful feature sometimes.

4.3 Maple as a Mathematical Word Processor

As a word processor the proper description of Maple's editing capabilities is primitive, but getting better. Table, tabs, and macros are only a few of the amenities it lacks. But it does have: line wrap, a good selection of fonts, the usual menu buttons for "**bold**", *italics*, "underline", and "left", "right" and "center" justification, and the capacity to export in multiple formats such as XML, HTML, LaTeX, and RTF. You can also select text, copy it (use control C), or cut it (control X), and paste it (control V) just like you can with Word 2000.

The main thing Maple has that other word processors lack is its ability to create and conveniently include both formatted mathematical text and high quality, mathematically-described images. Since our interest is in creating mathematical exercises and presenting them on the web this is for us a major advantage.

Exercise: Use the format menu convert the first 4 words of this sentence to bold underline text.

4.4 Placing Formatted Mathematics in a Maple Text Cell

There are at least two ways to get formatted mathematics into a Maple text cell. The simplest is simply to enter the text in Maple syntax such as: "The area under the graph of f(x) = x^2 between -3 and 4 is Int(t^2,t=0..4) ". Then, go back and select each Maple expression and convert it to formatted text with the **Format : convert to standard math** at the top of the Worksheet. (**Note:** you can get this menu more easily by 'right-clicking' the selected expression.)

In the example this would produce:

"The area under the graph of f(x) = x^2 between -3 and 4 is $\int_0^4 t^2\,dt$ ".

The reader will probably have noted the mathematical error: the limits of integration were supposed to be "-3..4". This can readily be rectified by selecting the incorrect "0" with the mouse

and observing that a "0" appears in the long narrow edit window which appears on the second line of the main menu and is part of the Context Bar

Change the "0" in the context bar window to a "-3" and hit enter or click back in the worksheet and the change will appear in the formatted symbol as $\int_{-3}^{4} t^2\,dt$.

The other way to place formatted symbols is to enter the expression in correct Maple syntax on an execution line and let Maple display it as Maple output. The output can then be copied and pasted in to the text cell. For instance we might have " ... between -3 and 4 is " , then go to an execution cell and enter the desired symbol in Maple syntax

```
> Int(t^2,t=0..4);
```

$$\int_{0}^{4} t^2\,dt$$

Now the symbol " $\int_0^4 t^2\,dt$ " can be copied and pasted after the 'is' to obtain " ... between -3 and 4 is $\int_0^4 t^2\,dt$ " . The reader will note that we made that same error again. This is to illustrate the corresponding approach to editing symbols which is based on the fact that *a formatted symbol can be copied and pasted onto a command line.* Thus if we copy " $\int_0^4 t^2\,dt$ " and paste in into a blank command line in some scratch space we get

```
> Int(t^2,t = 0 ..  4)
```

Now if we make the desired change , add a semicolon (";") and hit ENTER we get

```
> Int(t^2,t = -3 ..  4);
```

$$\int_{-3}^{4} t^2\,dt$$

Now copy and paste the correct symbol over the one for which it is to be substituted to get " ... between -3 and 4 is $\int_{-3}^{4} t^2\,dt$ " and then delete the scratch material.

Exercise: Format the expression 1 + x + 2*x^2 + 3*x^3. Also, format the square root of that expression.

4.5 Handling formatting problems that arise

There will be occasions when the formatted expression is not as good as you would like. Sometimes there are things you can do to improve the formatting, but you may have to compromise your standards.

Example: The unwanted 1 .

1/3*x/y converts to $\frac{1\,x}{3\,y}$ and we don't want the first 1 in the numerator. How to get rid of it.

Solution: Rewrite the expression 1/3*x/y as x/(3*y) and format $\frac{x}{3\,y}$

Another solution: Format each fraction separately and remove the * to get $\frac{1}{3}\,\frac{x}{y}$. This doesn't get rid of the 1, but does give a reasonable looking expression.

Exercise: Format 1/x*y/(1+x*y) in an eyepleasing manner.

Example: How to format a series of equalities, such as 1/2 = 2/4 = 3/6. If you select both equalities and convert, you get a ?. That means that the selected expression is not a **well-formed math expression** (according to Maple).

Solution: Format the first equation and the third fraction separately. $\frac{1}{2} = \frac{2}{4} = \frac{3}{6}$

Another solution: Format each fraction separately $\frac{1}{2} = \frac{2}{4} = \frac{3}{6}$. Both equality signs are alike.

Example: How to format a sum of 'n' terms, such as the sum of the reciprocals of 1 thru n. You could use Product, but you would like to **create ellipses** to use:

Solution: For ellipses, enclose 3 periods between back quotes as in `...`. Thus Product(1/i,i=1..n) =1 + 1/2 + 1/3 + `...` + 1/n converts to $\prod_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$.

There are many of these tricks you will as the need arises.

4.6 Some available conversions:

Subscripts: **A[p]** converts to
A_p

Greek Letters: **Delta** converts to Δ, **delta** converts to δ

An exception to this "upper case" "lower case" convention is **PI** converts to Π.

Pi and PI are constants standing for the ratio of the circumference of a circle to its diameter so both **Pi** and **pi** convert to the symbol π

Summation: **Sum(2^i, i=0..n)** converts to
$\sum_{i=0}^{n} 2^i$

Sum(2^i, i=0..infinity) converts to $\sum_{i=0}^{\infty} 2^i$

Product: **Product(1/(1-x^j),j = 0 .. 20)** converts to $\prod_{j=0}^{20} \frac{1}{1-x^j}$

Union: A union B converts to A **union** B (this and the other set operations are available in Maple 8 or 9)

Exercise: Format a set equation which expresses the the distributivity of of union over intersection.

4.7 Transferring Maple Output Into Text Cells

Most users prefer to format elementary mathematics symbols "in line" by using the Format->Convert To ->Standard Math sequence as described earlier. It is possible to have the Maple do the conversion "in advance" by entering the data on a command line in some scratch space and then copying and pasting the resulting, formatted output.

For example, suppose we want to expand (a + b)^2. We can type this into an input cell, press enter, and then copy the output into a text cell

```
> (a + b)^2;
```

$$(a+b)^2$$

$(a + b)^2$

like so.

Exercise: Complete and format the equation (x + 2)^4 =

so that the right hand side is the correct polynomial in x. Use the maple word expand to carry out the multiplication in the input cell just below, then copy and paste the output into the right hand side of the equation.

```
> expand((x+2)^4);
```

The **premature evaluation** problem. This occurs with using Maple output to produce your formatted mathematics. The Maple parser will try to simplify and evaluate expressions which you type in and so will obstruct your intention to generate an expression for the student to simplify. For example, even a simple sum of fractions will be added by the maple parser.

```
> 1/3 + 1/2 ;
```

$$\frac{5}{6}$$

In order to stop this behaviour, you can change the 3 and the 2 to symbols by putting back quotes around them.

```
> 1/‘3‘ + 1/‘2‘;
```

$$\frac{1}{3}+\frac{1}{2}$$

4.8 The importance of sectioning

One of the most important features that Maple worksheets have that most word processors don't have is the ability to section off material. Sectioning is very useful in the construction of sets of problems. For one thing it provides a convenient way to ecapsulate the calculations and other "scaffolding" in case we need them again. It also permits us to develop long sets of problems in a single worksheet and select only a subset for posting by hiding the rest in collapsed sections and serves as a vehicle for providing hints and supplemental material with posted problems.

The document you are reading was structured using Maple sections. If it is the source worksheet of the document, you can manipulate these sections: All sections can be opened or close from the View menu. Choose Open all Sections or Close all sections. You can make section ranges visible or invisible by toggling the **Show Section Ranges** item in the View Menu. An individual section can be closed by clicking on the 'boxed minus' icon at the top of the section. That section collapses to a 'boxed plus' icon, which you can reopen any time by just clicking. You can also select a collapsed section (without opening it), and then copy and paste it anywhere in the worksheet or into another worksheet. This is an extremely useful feature and one we use a lot.

4.9 Using Maple to make quizzes and exams

One of the uses a teacher puts Maple to is prepare short quizzes for their classes. You would type in a couple of questions, format any expressions that are used, and print it off. Before you do, you can insert page breaks (Insert Menu) and Show page numbers (Format Menu), if you want.

Exercise: Finish making up the two question algebra quiz below and format the expressions.

Algebra I - Quiz 3 Name_____

Instructions: This is a 5-minute quiz. Do all of your work on this paper.

1. (a+b)^2 - (a - b)^2 =
2.

Placing the cursor in an expanded section and pressing the "unsection" (the leftmost of the section/unsection pair) key will restore the paragraph to its unsectioned appearance.

5 An Introduction to the Maple Language

This section contains an introduction to some of the Maple vocabulary used for solving problems. It does not cover everything, just some of the basics. Read it through quickly, to get an overview of the language. Then you can come back and read with more understanding later.

5.1 Maple Arithmetic

By far the largest portion portion of the formatted mathematics symbols that one creates are simple arithmetic, algebraic, and trigonometric expressions of the type one would use when working with a scientific calculator. The following is a brief introduction to the basic syntax, commonly called **calculator syntax** in which one can formulate simple expressions which can be formatted or evaluated by Maple.

5.1.1 Real Numbers:

First, there is arithmetic: addition, subtraction, multiplication, division and exponentiation. These can be combined, just as on a calculator. The **order of precedence** is the the usual one: exponentiation first, then multiplication and division, then addition and subtraction. So saying (that is, typing in and pressing the enter key)

```
> 2-3+4/5*6^7;
```

$$\frac{1119739}{5}$$

is the same as saying

```
> (2-3)+(4/5)*(6^7);
```

$$\frac{1119739}{5}$$

You will notice that Maple works with fractions whenever possible, changing to decimal numbers only on demand. So entering

```
> 1/3 + 1/2;
```

$$\frac{5}{6}$$

However if any of the numbers in an expresion is in decimal form then the answer is returned as a decimal

```
> 1/3.  + 1/2;
```

$$0.8333333333$$

Another way to get decimals is to use the maple word **evalf** to convert a result to decimal form.

```
> evalf(1/2+1/3);
```

$$0.8333333333$$

The name for π, the area of the circle of radius 1, in Maple is **Pi** . So to calculate the area of a circle of radius 3,

```
> Pi*3^2;
```

$$9\,\pi$$

Maple leaves π in its symbolic form unless instructed to convert to floating point. Thus

```
> 3.0*Pi;
```

$$3.0\,\pi$$

If we want a floating point expression for an expression with π in it then we have to use the evalf command to force a floating point approximation for π to be used. For example, if we wanted the area of our circle to 50 digits of accuracy, we could say

```
> evalf(Pi*3^2,50);
```

$$28.274333882308139146163790449515525957774524594376$$

Without the second argumernt evalf returns the default number of digits. This number can be set by the user by setting the value of the parameter **Digits**

```
> Digits :=30;
```

$$Digits := 30$$

```
> evalf(9*Pi);
```

$$28.274333882308139146163790449 5$$

```
> sqrt(2) + sqrt(3) = sqrt(2.)  + sqrt(13.);
```

$$\sqrt{2} + \sqrt{3} = 5.019764837837084341920909991 68$$

One can change Digits to a smaller value when constructing problem answers. For example:

```
> Digits := 4;
```

$$Digits := 4$$

```
> sqrt(5.0); sqrt(6.0); sqrt(7.0); sqrt(8.);  Digits:= 10:
```

$$2.236$$
$$2.449$$
$$2.646$$
$$2.828$$

Alternatively, you can use evalf with a second argument.

```
> evalf(sqrt(5),6);
```

$$2.23607$$

For instance consider the following two problem statements:

Statement 1: Which of the following is the best estimate for $\sqrt{7}$?
A. 2.236, B. 2.450, C. 2.646, D. 2.828
Statement 2:
Problem: Which of the following is the best estimate for $\sqrt{7}$?
A. 2.236067978 , B. 2.449489743 , C. 2.645751311 , D. 2.828427125

Most people would agree that Statement 1 is a better elementary exercise that Statement 2.
Exercise: What is the 1000th digit in the decimal expansion of Pi?

5.1.2 Complex Numbers

Maple does arithmetic with **complex numbers** too. I is a Maple constant standing for $\sqrt{-1}$. So entering

```
> Joe:=(3+2*I)*(2-I)/(3-I);
```

$$Joe := \frac{23}{10} + \frac{11}{10}\,I$$

Complex numbers first come up when you solve quadratic equations.

```
> slns:= solve(3*x^2+2*x +1 ,x);
```

$$slns := -\frac{1}{3} + \frac{1}{3}\,I\,\sqrt{2},\ -\frac{1}{3} - \frac{1}{3}\,I\,\sqrt{2}$$

The evalf command works for complex numbers too.

```
> evalf(slns);
```

$$-0.3333333333 + 0.4714045206\,I,\ -0.3333333333 - 0.4714045206\,I$$

The evalc command is useful for converting a complex number to the form a + b*I.

```
> 2^I = evalc(2^I);
```

$$2^I = \cos(\ln(2)) + \sin(\ln(2))\,I$$

Exercise: Find all the solutions of $x^3 + 2\,x + 3 = 0$

5.1.3 Integers and Rational Numbers

Unlike a calculator Maple maintains **infinite precision** with integers and rational fractions. For instance if you ask it to calculate big powers of an integer or rational fraction it will retain every digit

```
> sam:=2^50;
```

$$sam := 1125899906842624$$

```
> joe:=3^50;
```

$$joe := 717897987691852588770249$$

```
> sally:=sam/joe;
```

$$sally := \frac{1125899906842624}{717897987691852588770249}$$

We often need to get a floating point estimate of such a quantity to have an idea of what size it is:

```
> evalf(sally);
```

$$0.1568328545\,10^{-8}$$

Note that the previous line didn't change sally - it just gave us a floating point approximaiton to sally.

The commands **numer** and **denom** will pick off the numerator and denominator of a fraction for us. This is often very useful:

```
> topofsally:=numer(sally);
```

$$topofsally := 1125899906842624$$

```
> bottomofsally:=denom(sally);
```

$$bottomofsally := 71789798769185258877024 9$$

There are extremely powerful functions for working with integers. The **ifactor** command factors an integer

```
> ifactor(topofsally);
```

$$(2)^{50}$$

Question: What is the prime factorization of your social security number? Your phone number?

The **irem** and **iquo** commands give the numerator and denominator when dividing one integer by another. If we divide 127680901 by 789783 for instance

```
> remainder:=irem(127680901,789783);
```

$$remainder := 525838$$

```
> quotient:=iquo(127680901,789783);
```

$$quotient := 161$$

We can check by:

```
> quotient*789783+remainder;
```

$$127680901$$

Exercise: If today is Monday what day of the week will it be in 6789756443568 days?
Hint: What is the remainder when you divide 6789756443568 by 7?

5.1.4 Basic Function and Calculus Expressions

Maple has "builtin" implementations of all of the basic functions encountered in an elementary precalculus or calculus course. All functions that appears on a scientific calculator constitute a small subset of those implemented in Maple. These can be combined with the common operations on functions (addition, multiplication, composition, etc.) with expected results in most respects. The principal caveat is that functions and operations return **expressions** rather than new functions. Thus there is a function sin which behaves as expected

```
> sin(Pi);
```

$$0$$

```
> sin(Pi/2);
```

$$1$$

And we can write expressions like

```
> f:=2*sin(x);
```

$$f := 2\sin(x)$$

However f is a symbol, not a function. If we ask for f(0) Maple will simply substitute for the symbol f. However it has no ability to assign a value to the resulting expression. There are simple ways to create the expected function which we will introduce as the need occurs.

```
> f(0);
```

$$2\sin(x)(0)$$

Limits, the derivative, antiderivatives, and definite integrals are available with an intuitive syntax. There are both "Inert" and active forms of the latter. The inert forms are capitalized. They return only a formatted representation of the expression. the active form actually evaluates (or at least attempts to evaluate) the expression. Here are some examples:

Inert form

```
> Limit(sin(x)/x,x=0);
```

$$\lim_{x \to 0} \frac{\sin(x)}{x}$$

Active form

```
> limit(sin(x)/x,x=0);
```

$$1$$

Inert form

```
> Int(sin(x),x=0..Pi);
```

$$\int_0^{\pi} \sin(x)\, dx$$

Active form

```
> int(sin(x),x=0..Pi);
```

$$2$$

Inert form

```
> Diff(x^5,x);
```

$$\frac{d}{dx}\left(x^5\right)$$

Active form

```
> diff(x^5,x);
```

$$5\, x^4$$

In-line formatting treats both active and inert forms of these operations the same. Thus lines (1) and (2) below will both become " Show that $\int \sin(x)\, dx = -\cos(x)$." when the mathematics expressions are formatted

Exercise format each of the expressions below.

1. Show that Int(sin(x),x) = -cos(x).
2. Show that int(sin(x),x) = -cos(x).

5.1.5 Simple Matrices

The simplest way to form a **matrix** is by listing its rows. For instance if we format the following

What is the determinant of A = matrix([[1,2],[5,6]]) ?

we get

What is the determinant of $A = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}$?

An entry in a Maple matrix can be any valid Maple expression. Thus

matrix([[sin(x^2), int(sin(x^2,x)],[cos(x^2),int(cos(x ^2),x)]]) formats as

$$\begin{bmatrix} \sin(x^2) & \text{int}(\sin(x^2,\, x)) \\ \cos(x^2) & \int \cos(x^2)\, dx \end{bmatrix}$$

If we recall that we can convert text to expressions by enclosing it in back quotes (" ‘ ") then we can create simple tables

matrix([[‘x‘,‘square of x‘,‘cube of x‘],[1,1,1],[2,4,8],[3,9,27],[4,‘___‘,‘___‘]]) formats to

x	*square of* x	*cube of* x
1	1	1
2	4	8
3	9	27
4	__	__

Exercise: Use a matrix to create a table of function values for the the polynomial function f(x) = 3*x^2 + 1

5.2 Expressions, Names, Statements, and Assignments

Quantities to be computed like 1/2+1/3 are called **expressions** . A **name** is a string of characters which can be used to store the result of a computation. A statement in Maple is a string of names and expressions terminated with a semicolon, or a colon if you don't want to see the output, which when entered will produce some action. The **assignment** statement is one of the most common statements. It is of the form name := expression; For example, the assignment

```
> area := Pi*3^2;
```

$$area := 9\,\pi$$

stores $9\,\pi$ in a location marked by the name area. A more useful assignment for the area of a circle is

```
> area := Pi*r^2;
```

$$area := \pi\, r^2$$

In this case, the expression $\pi\, r^2$ is stored in area and with this assignment, the area of a circle of any given radius can be computed using the Maple word **subs** . So to calculate the area when r is 3, we enter

```
> subs(r=3,area);
```

$$9\,\pi$$

Here, it is convenient to think of the assignment as defining area as a **function** of the radius r.

5.3 Functions

A function is a rule f (possibly very complicated) for assigning to each argument x in a given set, a unique value f(x) in a set. In calculus the arguments and values of a function are always real numbers, but the notion of function is much more flexible than that.

Functions can be defined in several useful ways in Maple.

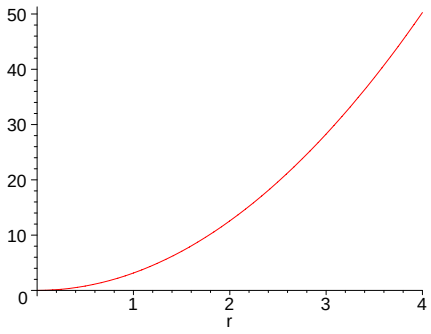
As an expression: The assignment

```
> area := Pi*r^2;
```

$$area := \pi\, r^2$$

defines the area of a circle as a function of it's radius. The area function defined as an expression is evaluated with subs. Since this function assigns real numbers to real numbers, its values can be plotted on a graph with the Maple word plot. So the statement

```
> plot(area,r=0..4);
```



will produce in a separate plot window, the graph of the area function over the interval from $r = 0$ to $r = 4$.

With the arrow operator the assignment: If you have a simple function, you can often use the arrow operator. For example,\

```
> area := r -> Pi*r^2;
```

$$area := r \rightarrow \pi\, r^2$$

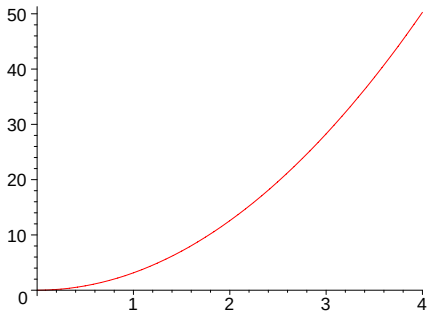
defines the area function also. Now to find the area of a circle of radius 3, we simply enter the statement

```
> area(3);
```

$$9\,\pi$$

To plot this function over the domain r = 0..4 , say

```
> plot(area,0..4);
```



Note that the variable r is omitted here.

Use unapply . The ugly little word **unapply** transforms expressions of one or more variables into fuctions defined by an arrow operator. For example, if we had a polynomial defined by the assignment

```
> pol := x^2 + 4*x -1;
```

$$pol := x^2 + 4\,x - 1$$

then the assignment

```
> pol := unapply(pol,x);
```

$$pol := x \to x^2 + 4\,x - 1$$

turns pol into a function defined by an arrow operator.

As a procedure: The Maple word **proc** can be used to define functions. For example,

```
> area := proc(r) Pi*r^2 end;
```

$$area := \mathbf{proc}(r)\,\pi * r^2\ \mathbf{end\ proc}$$

defines the area function too. It is evaluated and plotted as in the arrow operator definition. One advantage of this way of defining a function is that the domain can be specified. For example, the domain of the area function for a circle is all positive real numbers. This can be inserted into the procedure, with the Maple word **ERROR** . The message must be enclosed in backquotes ‘ or doublequotes ”, which is on the key with the **tilde** .

```
> area := proc(r)
> if r <= 0 then ERROR("radius must be positive")  else
> Pi*r^2 fi end;
```

$$area := \mathbf{proc}(r)\ \mathbf{if}\ r \le 0\ \mathbf{then}\ \mathrm{ERROR}(\text{“radius must be positive”})\ \mathbf{else}\ \pi * r^2\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

```
> area(3);
```

$$9\,\pi$$

```
> area(-3);
```

```
Error, (in area) radius must be positive
```

Note the **if..then..fi** control statement here. You can learn more about the word if by typing **?if** in an input cell and entering it.

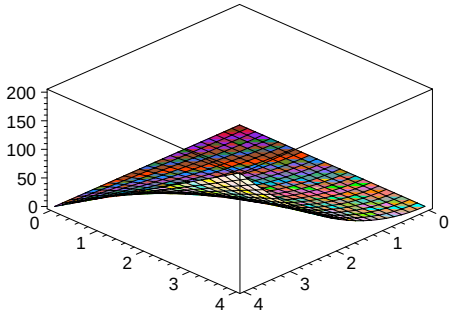
Functions of two variables can be defined and plotted just as easily in Maple as functions of one variable. For example, the volume V of a cylinder of height h and radius r is defined by

```
> V := (r,h) -> Pi*r^2*h;
```

$$V := (r,\,h) \to \pi\,r^2\,h$$

To see what the graph of V looks like, use **plot3d** .

```
> plot3d(V,0..4,0..4,axes=boxed);
```



Which way of defining a function is the preferred way? That really depends on the situation. The expression method works well for functions which have only one rule of evaluation, but eventually you cannot avoid using an -> or proc definition. You will find yourself using arrow or proc definitions more and more as time goes by.

Piecewise defined functions:

Many functions can only be described by stating various rules for various parts of the domain. The Maple word **piecewise** will help with defining such functions. Here is an example to show usage.

```
> f :=unapply(piecewise(x <= -1,x^3+8, x <= 2,7+ 2*x, x <= 4, 11 -
  cos(x),3*x),x);
```

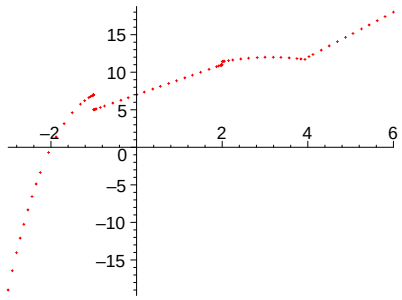
$$f := x \rightarrow \text{piecewise}(x \leq -1,\ x^3 + 8,\ x \leq 2,\ 7 + 2\,x,\ x \leq 4,\ 11 - \cos(x),\ 3\,x)$$

```
> f(2);
```

$$11$$

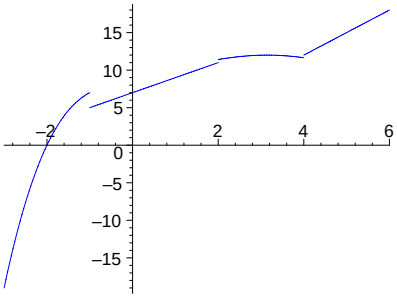
When plotting piecewise defined functions, sometimes style = point is better.

```
> plot(f, -3..6,style= point);
```



You can also plot discontinuous functions by setting discont = true.

```
> plot(f, -3..6,discont=true,color=blue);
```



Composition of functions

One of the important skills needed for calculus and other mathematics is the skill to see a function as built up from simpler functions by the 4 arithmetic operations plus **composition** . In Maple these operations are named by +, *, -, /, and @. So for example the function f(x) = $\frac{x^2+1}{\sqrt{x^3-1}}$ can be seen as the quotient of g (x) = x^2+1 by h(x) = $\sqrt{x^3-1}$.

```
> g := x-> x^2 +1;  h := x->sqrt(x^3-1);
```

$$g := x \to x^2 + 1$$

$$h := x \to \sqrt{x^3 - 1}$$

```
> f := g/h;
```

$$f := \frac{g}{h}$$

```
> f(x);
```

$$\frac{x^2 + 1}{\sqrt{x^3 - 1}}$$

This analysis of a function into simpler functions is not unique and can often be extended further. For example, the denominator function h is seen to be the composition of the square root function $\sqrt{x}$ with the cubic polynomial function k(x) = $x^3 - 1$.

```
> k := x-> x^3 -1;
```

$$k := x \to x^3 - 1$$

```
> f := g/(sqrt @ k) ;
```

$$f := \frac{g}{\mathrm{sqrt}@k}$$

```
> f(x);
```

$$\frac{x^2 + 1}{\sqrt{x^3 - 1}}$$

Problem: Define three functions g,h, and k so that the function f defined by f(x) = $(\sin(x) + x)\,(x^2+1)^4$ is $g\,(h@k)$

5.4 Built in Maple functions and Operations with Functions

All of the standard **scientific functions** are built into Maple. For example, sqrt is the square root function, abs is the absolute value function, the trig and inverse trig functions are sin , arcsin

, cos , etc., the natural logarithm and exponential functions are ln and exp . For a complete list of built in functions, type

```
> ?inifcns;
```

New functions can be obtained from old functions by use of the arithmetic operations of addition, subtraction, multiplication, and division together with the operation of composition, which is denoted by @ . Thus the function defined by the assignment

```
> y := sin(cos(x^2+3));
```

$$y := \sin(\cos(x^2 + 3))$$

and evaluated at x=3 by

```
> subs(x=3.,y);
```

$$\sin(\cos(12.))$$

could also be defined by the assignment

```
> y := sin@cos@(x->x^2+3);
```

$$y := \sin@\cos@(x \to x^2 + 3)$$

and evaluated at x=3 by

```
> y(3.);
```

$$0.7472099577$$

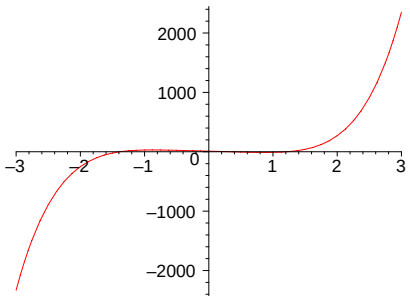
5.5 Using Maple as a fancy graphing calculator.

It is convenient to think of Maple as a fancy graphing calculator for many purposes. For example, suppose you want to find the real solutions of the equation $x^5 - 30\,x - 2 = 0$ in the interval $-3..3$. Then we can just plot the right hand side of the equation and look for where the graph crosses the x-axis.

```
> f := x -> 10*x^5 - 30*x +10 ;
```

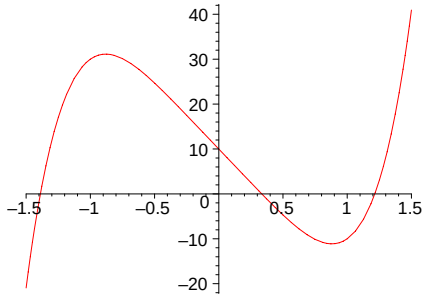
$$f := x \to 10\,x^5 - 30\,x + 10$$

```
> plot(f,-3..3);
```



By inspection, the graph crosses near 0. We can look closer.

```
> plot(f,-1.5..1.5);
```



We see that the graph crosses 3 times, the largest solution being between 1 and 1.5. If we wanted the largest solution more accurately, we could use **fsolve** . Note the syntax. There are three arguments, the equation to solve, the variable to solve for, and the interval in which to search for a solution.

```
> fsolve(f(x)=0,x,1..1.5);
```

$$1.214648043$$

5.6 Data types, Expression Sequences, Lists, Sets, Arrays, Tables

5.6.1 Data types

Maple expressions are classified into various **data types** . For example, arithmetic expressions are classified by whether they are sums **type '+'** , products **type '*'** , etc.

The Maple word **whattype** will tell what type a particular expression is.

```
> whattype(1/2);
```

$$\mathit{fraction}$$

```
> whattype(a + b);
```

$$+$$

```
> whattype(x^2 + x = 2*x - 1);
```

$$=$$

```
> whattype(a,b,3);
```

$$\mathit{exprseq}$$

5.6.2 Expression Sequence.

An **exprseq** , or **expression sequence** , is any sequence of expressions separated by commas. For example,

```
> viola := 1,2, w*r+m, a=b+c, 1/2, (x+y)/z,'hello';
```

$$\mathit{viola} := 1,\ 2,\ w\,r + m,\ a = b + c,\ \frac{1}{2},\ \frac{x + (\sin@\cos@(x \to x^2 + 3))}{z},\ \mathit{hello}$$

is an assignment to viola of an expression sequence of 7 expressions. To refer to the sixth expression in this sequence, use the expression viola[6];

```
> viola[6];
```

$$\frac{x + (\sin@\cos@(x \to x^2 + 3))}{z}$$

5.6.3 Lists.

A **list** is an expression sequence enclosed by square brackets. So

```
> explist:= [viola];
```

$$explist := [1,\ 2,\ w\,r + m,\ a = b + c,\ \frac{1}{2},\ \frac{x + (\sin@\cos@(x \to x^2 + 3))}{z},\ hello]$$

makes a list whose terms are those in viola . As with expression sequences, we can refer to particular terms of a list by appending to its name the number of the term enclosed in square brackets. Thus to get the fifth term of explist , type the expression

```
> explist[3];
```

$$w\,r + m$$

You can also reference the fifth term in this list by by using the Maple word **op** .

```
> op(3,explist);
```

$$w\,r + m$$

In general, op(n,explist); returns the nth term in the list explist .

To count how many terms are in a list, use the word **nops** . So for example,

```
> nops(explist);
```

$$7$$

tells us that there are 7 terms in the list explist . nops comes in handy when you don't want to (or aren't able to) count the terms in a list by hand.

You can't directly use the word nops to count the number of terms in an expression sequence. But you can put square brackets around the expression sequence and count the terms in the resulting list. This device is used again and again.

```
> nops(3,4,a);
```

```
Error, wrong number (or type) of parameters in function nops
```

```
> nops([3,4,a]);
```

$$3$$

A point in the plane is a list of two numbers. Points can be added and subtracted and multiplied by a number.

```
> p := [1,2]; q := [-3,1];
```

$$p := [1,\ 2]$$

$$q := [-3,\ 1]$$

```
> w := 3*p + 2*q - p;
```

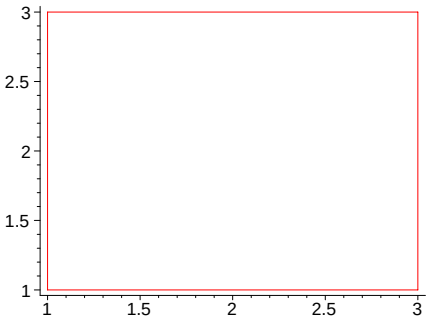
$$w := [-4,\ 6]$$

One important use of lists is to make lists of points to plot. For example, to draw a picture of the square with vertices (1,1), (3,1), (3,3), (1,3), make a list and then plot it.

```
> ab := [[1,1],[3,1],[3,3],[1,3],[1,1]];
```

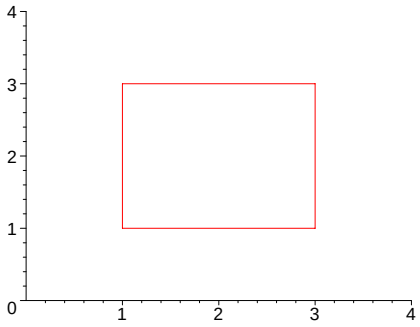
$$ab := [[1,\ 1],\ [3,\ 1],\ [3,\ 3],\ [1,\ 3],\ [1,\ 1]]$$

```
> plot(ab);
```



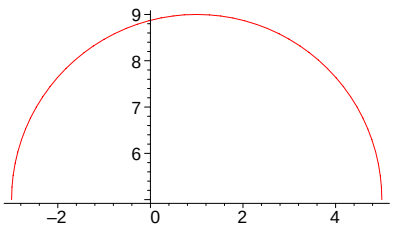
Notice in the graph that the origin is not included in the field of view. We can specify that by using the view= option.

```
> plot(ab,view=[0..4,0..4]);
```



Another use of lists is with **parametric plots** . If you have a curve in the plane described parametrically with $x = \mathrm{f}(t)$, $y = \mathrm{g}(t)$, as the parameter t runs from a to b, then you can draw it by making up a 3 term list to give to plot. Say you wanted to draw the upper half of the circle of radius 4 centered at (1,5). Then the list consists of the expressions for the x and y coordinates followed by an equation giving the range of the parameter.

```
> plot([1+4*cos(t),5+4*sin(t),t=0..Pi],scaling=constrained);
```



If you had to draw several pieces of circles, you might define a function to simplify things. You can call the function whatever you want, say circ.

```
> circ := (h,k,r,f,l) -> [h+r*cos(t),k+r*sin(t),t=f..l];
```

$$circ := (h,\ k,\ r,\ f,\ l) \rightarrow [h + r\cos(t),\ k + r\sin(t),\ t = f..l]$$

```
> ab[1];
```

$$[1,\ 1]$$

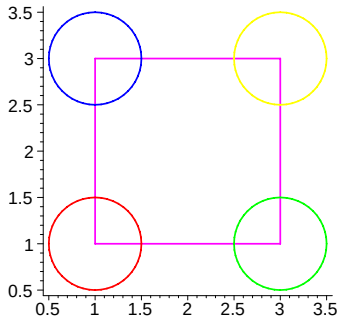
So if we wanted circles of radius 1/2 centered at the corners of the square ab we can construct the sequence of lists. Here is where the Maple word **seq** comes in handy.

```
> circs := seq(circ(op(ab[i]), 1/2,0,2*Pi),i=1..nops(ab));
```

$$circs := [1 + \frac{1}{2}\cos(t),\ 1 + \frac{1}{2}\sin(t),\ t = 0..2\,\pi],\ [3 + \frac{1}{2}\cos(t),\ 1 + \frac{1}{2}\sin(t),\ t = 0..2\,\pi],$$
$$[3 + \frac{1}{2}\cos(t),\ 3 + \frac{1}{2}\sin(t),\ t = 0..2\,\pi],\ [1 + \frac{1}{2}\cos(t),\ 3 + \frac{1}{2}\sin(t),\ t = 0..2\,\pi],$$
$$[1 + \frac{1}{2}\cos(t),\ 1 + \frac{1}{2}\sin(t),\ t = 0..2\,\pi]$$

In order to plot these circles, you need to enclose them in curly brackets to make a set of the sequence before you give them to plot . See below for a discussion of sets.

```
> plot({circs,ab},scaling=constrained,thickness=4);
```



Sometime you might want to split a list of points to plot into a list of x-coordinates and another list of ycoordinates. The Maple word **seq** is very handy for this and many other operations. So to split off from ab the odd and even terms–

```
> xdat := [ seq(ab[i][1],i=1..nops(ab) )];
```

$$xdat := [1,\ 3,\ 3,\ 1,\ 1]$$

```
> ydat := [seq(ab[i][2],i=1..nops(ab) )];
```

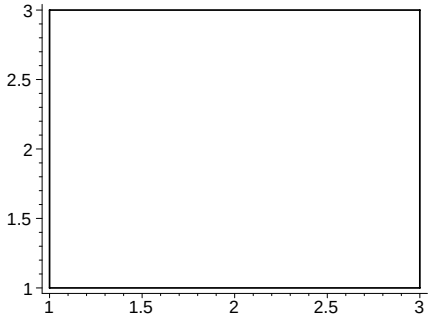
$$ydat := [1,\ 1,\ 3,\ 3,\ 1]$$

What about the converse problem? Building up a list of points to plot from two lists can also be done. The first thing you might think of works.

```
> newab:= [seq([xdat[i],ydat[i]],i=1..nops(xdat))];
```

$$newab := [[1,\ 1],\ [3,\ 1],\ [3,\ 3],\ [1,\ 3],\ [1,\ 1]]$$

```
> plot(newab,color=black,thickness=4);
```



5.6.4 Sets

A **set** is an expression sequence enclosed by curly brackets. This is much different from a list. For one thing, the order in which you specify the members of a set may not be the order in which they are stored. Also each member of the set is only stored once, no matter how many times you list it.

```
> Aset := {y+x+1,1,2,1,4,`bill`,x+y+1,`bill`};
```

$$Aset := \{1,\ 2,\ 4,\ bill,\ (\sin@\cos@(x \to x^2+3)) + x + 1\}$$

The set operations of **union** , **intersection** , and **minus** are at your beck and call.

```
> Anotherset := Aset union {4,3,a,7} ;
```

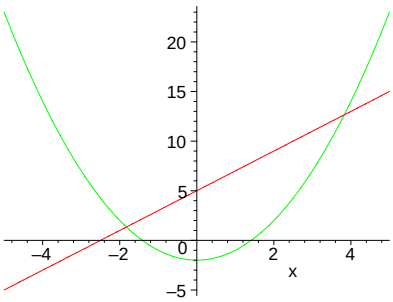
$$Anotherset := \{1,\ 2,\ 3,\ 4,\ 7,\ a,\ bill,\ (\sin@\cos@(x \to x^2+3)) + x + 1\}$$

```
> Anotherset minus Aset, Anotherset intersect Aset;
```

$$\{3,\ 7,\ a\},\ \{1,\ 2,\ 4,\ bill,\ (\sin@\cos@(x \to x^2+3)) + x + 1\}$$

Sets are important when plotting more than one function at at time, to plot the quadratic function $x^2 - 2$ and the linear function $2\,x + 5$ on the same axes,

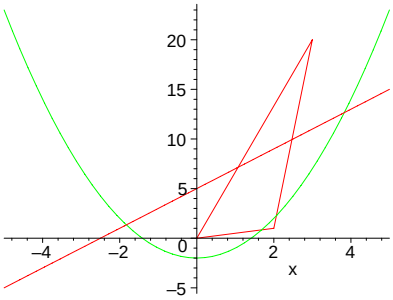
```
> plot({x^2-2,2*x+5},x=-5..5);
```



plots the parabola $y = x^2 - 2$ and the line $y = 2\,x + 5$ over the domain $x = -5..5$ on the same graph. If you have a very complicated drawing to make, you can use **plots[display]** from the

plots package. Just give names to the plots you want to display and then display the list of plots you have named.

```
> pl1 := plot({x^2-2,2*x+5},x=-5..5):
> pl2 := plot([[2,1],[3,20],[0,0],[2,1]]):
> plots[display]([pl1,pl2]);
```



5.6.5 Tables and Arrays

A **table** is a special kind of data structure which is very flexible. The packages of special vocabularies are really tables whose indices of the package are the names of the procedures and whose entries are the bodies of the procedures. We do not make much use of tables in this handbook, except for arrays. Note: The new packages in Maple are all **modules** . See below.

An **array** is a special kind of table whose indices are numerical. Somet useful arrays are matrices (2 dimensional arrays) and vectors (1 dimensional arrays).

Matrix operations are made using Maple word **evalm** together with the symbol for **matrix multiplication** &* .

```
> a := array(1..2,1..2);
```

$$a := \mathrm{array}(1..2,\ 1..2,\ [])$$

creates a 2 by 2 matrix, whose entries are accessed as a[1,1] etc.

So to rotate the square $ab := [[1,\ 1],\ [3,\ 1],\ [3,\ 3],\ [1,\ 3],\ [1,\ 1]]$ through an angle of 31 degrees counter clockwise about the origin and display it, we could proceed as follows.

```
> rot := array([[cos,-sin],[sin,cos]]);
```

$$rot := \begin{bmatrix} \cos & -\sin \\ \sin & \cos \end{bmatrix}$$

```
> ang := evalf(Pi/180*31);
```

$$ang := 0.5410520681$$

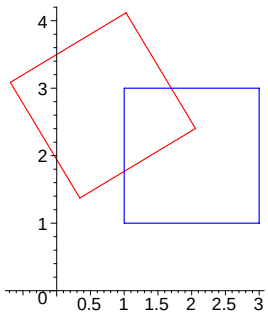
```
> ab := [[1,1],[3,1],[3,3],[1,3],[1,1]];
```

$$ab := [[1,\ 1],\ [3,\ 1],\ [3,\ 3],\ [1,\ 3],\ [1,\ 1]]$$

```
> rotab := [seq(convert( evalm(rot(ang)&*ab[i]),list) ,i=1..nops(ab)
  )];
```

$$rotab := [[0.3421292258,\ 1.372205376],\ [2.056463827,\ 2.402281526],\ [1.026387677,\ 4.116616127],\ [-0.6879469243,\ 3.086539977],\ [0.3421292258,\ 1.372205376]]$$

```
> plot({  [[0,0]],ab,rotab} ,thickness=3,color=[red,blue]
,scaling=constrained );
```



5.7 Maple control statements

There are two especially important types of control statements . One is the **repetition loop** , and the other is the **conditional execution statement** . The **repetition loop** is **for ... from ... by ... to ... while ... do ... od** and the **conditional execution statement** is **if ... then ... elif ... else ... fi**

5.7.1 Repitition loops

These statements can be used interactively or in a procedure.

Example: Add up the first 10000 numbers.

```
> s := 0:    for i from 1 to 10000 do  s := s+i od:  s;
```

$$50005000$$

Note: The Maple word **sum** could be used here instead.

```
> i := 'i':    sum(i,i=1..10000);
```

$$50005000$$

```
> i;
```

$$i$$

Note: This could also be computed using **convert** and **seq** .

```
> convert([seq(i,i=1..10000)],`+`);
```

$$50005000$$

Example: Add up the odd numbers between 1 and 973.

```
> s := 0:    for i from 1 by 2 to 973 do s := s + i od:  s;
```

$$237169$$

Example: Add up the prime numbers between 1 and 10000

```
> s := 0:    for i from 1 to 10000 do if isprime(i) then s := s+i fi od:
  s;
```

$$5736396$$

Example: Add up the first 1000 primes.

```
> s := 0:  j := 0:    for i from 1 by 2 while j <1000 do if isprime(i)
  then
  s := s+i; j:= j+1 fi od:    s;
```

$$3690838$$

Example: Compute the cubes of the first five positive integers and store them in a list. Then do it again, storing them in an array.

Solution with lists:

```
> locube := NULL:  # start with the empty exprseq
  for i from 1 to  5 do
  locube :=  locube ,i^3  od:
  locube := [locube];  # make locube a list.;
```

$$locube := [1, 8, 27, 64, 125]$$

```
> locube[3]:=15;
```

$$locube_3 := 15$$

```
> locube;
```

$$[1, 8, 15, 64, 125]$$

Note the way the list is built up from an empty exprseq **NULL** . Each time through the loop, one more term is added onto the end of the sequence. At the end, square brackets are put around the sequence, making it a list. With arrays, one can be more direct.

Solution with arrays:

```
> aocube := array(1..5):    # initialize the array.
> for i from 1 to  5 do aocube[i]:= i^3 od;
```

$$aocube_1 := 1$$
$$aocube_2 := 8$$
$$aocube_3 := 27$$
$$aocube_4 := 64$$
$$aocube_5 := 125$$

```
> op(aocube);  # to see the array
```

$$[1, 8, 27, 64, 125]$$

Now the array aocube has the numbers stored in it. To refer to the third element of aocube , we would enter aocube[3] just as if it were a list, rather than an array.

```
> aocube[3]:=0;
```

$$aocube_3 := 0$$

```
> print(aocube);
```

$$[1, 8, 0, 64, 125]$$

```
> locube := [locube[1],locube[2],0,locube[4],locube[5]];
```

$$locube := [1, 8, 0, 64, 125]$$

5.7.2 Conditional Execution: if .. then .. elif .. else .. fi;

There are lots of times when you need to consider cases, and they can all be handled with the **if .. then .. elif .. else .. fi;** statement. For example, many functions are defined piecewise. The absolute value function abs is such a function.

Problem: Define your own version of the absolute value function.

A solution:

```
> myabs := proc(x) if x > 0 then x else -x fi end;
```

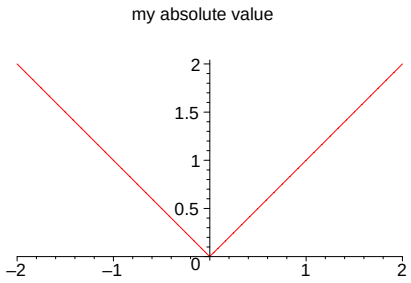
$$myabs := \mathbf{proc}(x)\,\mathbf{if}\,0 < x\,\mathbf{then}\,x\,\mathbf{else}\,-x\,\mathbf{end\ if end\ proc}$$

We test our definition and plot it.

```
> myabs(-23);
```

$$23$$

```
> plot(myabs,-2..2,scaling=constrained,title=`my absolute value`);
```



If there are more than two cases, you will need one or more elifs in your statement. For example, let f(x) = abs(x) if x is negative else if x is between 0 and 1 then let f(x) be x*x, else if x is greater than one, let f(x) = x*x*x.

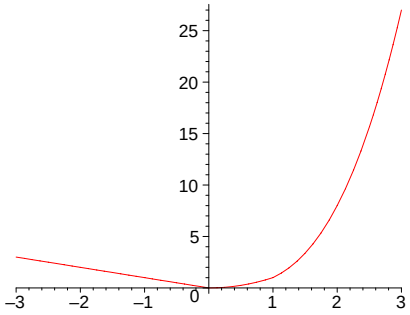
```
> f:= proc(x)
  if x < 0 then -x elif x <= 1 then x^2 else x^3 fi end;
```

$$f := \mathbf{proc}(x)\,\mathbf{if}\,x < 0\,\mathbf{then}\,-x\,\mathbf{elif}\,x \le 1\,\mathbf{then}\,x^2\,\mathbf{else}\,x^3\,\mathbf{end\ if end\ proc}$$

```
> f(-2);
```

$$2$$

```
> plot(f,-3..3);
```



5.8 A Brief Vocabulary of Maple Words

Here are some Maple words useful in mathematical problem solving, together with examples of their usage. For more information on these words and others, look at the helpsheets and use the help browser.

```
> y := (x+3)/tan(x^2-1);  # use 'colon-equal' to make assignments.
```

$$y := \frac{x+3}{\tan(x^2-1)}$$

```
> factor(x*2 + y*x);  #factors polynomial expressions
```

$$\frac{x\,(2\tan((x-1)\,(x+1))+x+3)}{\tan((x-1)\,(x+1))}$$

```
> collect(x*2 + y*x,x);  # collects like powers of x.
```

$$\frac{x^2}{\tan(x^2-1)} + (2+\frac{3}{\tan(x^2-1)})\,x$$

```
> diff(cos(x),x);  # calculates the derivative
```

$$-\sin(x)$$

```
> D(cos);  # the differential operator
```

$$-\sin$$

```
> y := denom((a+b)/(e+f));  # assigns e+f to y.
```

$$y := e+f$$

```
> y := 'y';  #  makes y a variable again.
```

$$y := y$$

```
> evalc((2+3*I)^3);  # performs complex arithmetic
```

$$-46+9\,I$$

```
> evalf(1/2^9);  #evaluates 1/2^9 to a decimal number
```

$$0.001953125000$$

```
> expand((x+b)^7);  # expands the product
```

$$x^7+7\,x^6\,b+21\,x^5\,b^2+35\,x^4\,b^3+35\,x^3\,b^4+21\,x^2\,b^5+7\,x\,b^6+b^7$$

```
> p := x^2+5*x+6;  # assigns the quadratic to p.
```

$$p := x^2+5\,x+6$$

```
> factor(p);  # factors the polynomial
```

$$(x+3)(x+2)$$

```
> fsolve(x^5-3*x=1,x,0..2);  # solve eqn for x in 0..2
```

$$1.388791984$$

```
> int(x*exp(x),x);   # returns an antiderivative.
```

$$x\,e^x - e^x$$

```
> Int(x*exp(x),x=0..1);  # A passive integral.
```

$$\int_0^1 x\,e^x\,dx$$

```
> map(x->x^2,[1,3,2,5]); # returns a list of squares.
```

$$[1, 9, 4, 25]$$

```
> nops([3,4,x,1]);  # returns the number of terms  in the list.
```

$$4$$

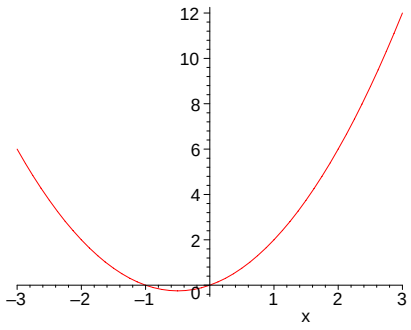
```
> numer((a+b)/c);  # gives numerator, here a+b
```

$$a+b$$

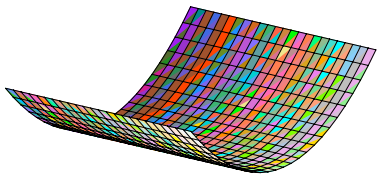
```
> op([3,4,1,x]);  # strips the brackets off the list
```

$$3,\ 4,\ 1,\ x$$

```
> plot(x^2+x, x=-3..3);  # plots x^2+x as x goes  from -3 to 3.
```




```
> plot3d(x^2+y,x=-2..2,y=0..2);  # plots a surface
```



```
> f := x -> x^2;  # defines the squaring function.
```

$$f := x \to x^2$$

```
> f(3);  # then returns 9.
```

$$9$$

```
> quo((x^4-4),(x^2-2),x);  # divides polynomials
```

$$x^2 + 2$$

```
> iquo(23,2) ;  # divides the integers
```

$$11$$

```
> rem((x^4-4*x+3),(x^2-2),x);  # gives the remainder
```

$$7 - 4\,x$$

```
> irem(23,2) ; # gives the integer remainder
```

$$1$$

```
> restart;  # very handy.  This word resets all assignments.
> eq1 := x^2 + 3*x -1 = a;  # assigns the equation
```

$$eq1 := x^2 + 3\,x - 1 = a$$

```
> rhs(eq1);  # yields the righthand side of eq1.    There is also an lhs.
```

$$a$$

```
> simplify(a/x+b/y);  # sometimes simplifies expr.
```

$$\frac{a\,y + b\,x}{x\,y}$$

```
> solve(a*x+4*y=0,x);  # solve the equation  for x.
```

$$-\frac{4\,y}{a}$$

```
> subs(x=5,x^2+x);  # substitute 5 for x where it  occurs in x^2+x.
```

$$30$$

```
> i := 'i';   # makes i a variable again
```

$$i := i$$

```
> sum((i^2,i=2..9));  # add up the 2nd thru 9th squares
```

$$284$$

5.9 Defining your own words

Maple's mathematical vocabulary is extensive, but it can never be complete. That's because we are always discovering new ways to do things. The methods can then be defined in a **Maple procedure** .

One way to develop the definition is in a series of input cells: For example, suppose we wanted to define a word to solve a quadratic equation $a\,x^2 + b\,x + c = 0$. We want the inputs to be the coefficients a,b,c of the equation, and the output to be the roots and a message describing the natrue of the roots. So given the inputs values in an input cell. Now develop the body of the

procedure in an input cell below it. Thus the nature of the roots is determined by the discriminant $b^2 - 4\,a\,c$, so we need to use a conditional statement here.

```
> a := 3 :  b:= 0:  c := 5:
> if b^2 - 4*a*c < 0 then print("negative discriminant.  no real
  roots.")
  elif b^2 - 4*a*c = 0 then print("discriminant is 0.    1 real root.");
> (-b +sqrt(b^2-4*a*c))/(2*a)
  else print("discriminant is positive, two real roots.");
  (-b +sqrt(b^2-4*a*c))/(2*a),(-b -sqrt(b^2-4*a*c))/(2*a)    fi;
```

"negative discriminant. no real roots."

After changing the values of a, b, and c in the input cell and testing the body of the procudure, we can define the procedure in an input cell. Use the copy and paste item in the edit menu to put the body into the cell below then at the top put the 'proc line' and at the bottom put the 'end line'.

```
> quadroots := proc(a,b,c)  # the proc line
  if b^2 - 4*a*c < 0 then print("negative discriminant.  no real roots.")
> elif b^2 - 4*a*c = 0 then print("discriminant is 0.    1 real root.");
  (-b +sqrt(b^2-4*a*c))/(2*a)
  else print("discriminant is positive, two real roots.");
> (-b +sqrt(b^2-4*a*c))/(2*a),(-b -sqrt(b^2-4*a*c))/(2*a)    fi;
  end;  # the end line
```

quadroots := **proc**(*a*, *b*, *c*)
if $b^2 - 4 * a * c < 0$ **then** print("negative discriminant. no real roots.")
elif $b^2 - 4 * a * c = 0$ **then**
print("discriminant is 0. 1 real root."); $1/2 * (-b + \mathrm{sqrt}(b^2 - 4 * a * c))/a$
else
print("discriminant is positive, two real roots.");
$1/2 * (-b + \mathrm{sqrt}(b^2 - 4 * a * c))/a,\ 1/2 * (-b - \mathrm{sqrt}(b^2 - 4 * a * c))/a$
end if
end proc

Now, test the procedure again

```
> quadroots(2,3,4);
```

"negative discriminant. no real roots."

```
> quadroots(2,sqrt(32),4);
```

"discriminant is 0. 1 real root."

$$-\sqrt{2}$$

```
> quadroots(2,10,4);
```

"discriminant is positive, two real roots."

$$-\frac{5}{2} + \frac{\sqrt{17}}{2},\ -\frac{5}{2} - \frac{\sqrt{17}}{2}$$

At this point, you have added a word to the Maple language. It can be used just like any of the other words. It is not a permanent addition, however. In order to use it, you would need to load a Maple worksheet containing the definition and

One way to keep all your procedures together is to put them into a " **package** ". A package

in Maple is a table or a module whose indices are names of procedures and whose entries are the bodies of those procedures. For example, in this worksheet, we have defined two procedures that we might like to keep in a package, **quadroots** and one further back **myabs** . We could call the package 'mystuff'.

```
> mystuff := table([]);
```

$$mystuff := \mathrm{table}([])$$

```
> mystuff[quadroots]:= proc(a,b,c)  # the proc line
  if b^2 - 4*a*c < 0 then print("negative discriminant.  no real roots.")
> elif b^2 - 4*a*c = 0 then print("discriminant is 0.    1 real root.");
  (-b +sqrt(b^2-4*a*c))/(2*a)
  else print("discriminant is positive, two real roots.");
> (-b +sqrt(b^2-4*a*c))/(2*a),(-b -sqrt(b^2-4*a*c))/(2*a)    fi;
  end:    # the end line
  mystuff[myabs]:= proc(x) if x > 0 then x else -x fi end:
  with(mystuff);
```

$$[myabs,\ quadroots]$$

5.9.1 modules

Most of the new packages in Maple are kept in **modules** , which you can read about using help. Modules have certain advantages over tables. When you load a package defined as a module, the procedures defined in the module which are listed in the export line can be accessed; the ones listed in the local line cannot . So if your package has procedures which are only used by other procedures in the module and would not be useful otherwise, they are not shown. Here is the package mystuff stored in a module with myabs keep private (for no particular reason other than to show it can be done).

```
> mystuff := module()
  options package;
  export quadroots;
> local myabs;
  quadroots := proc(a,b,c)
  if b^2 - 4*a*c < 0 then print("negative discriminant.  no real roots.")
> elif b^2 - 4*a*c = 0 then print("discriminant is 0.    1 real root.");
  (-b +sqrt(b^2-4*a*c))/(2*a)
> else print("discriminant is positive, two real roots.");
  (-b +sqrt(b^2-4*a*c))/(2*a),(-b -sqrt(b^2-4*a*c))/(2*a)    fi;
  end:
> myabs:= proc(x) if x > 0 then x else -x fi end:
  end module:
> with(mystuff);
```

$$[quadroots]$$

```
> myabs(4);
```

$$\mathrm{myabs}(4)$$

```
> quadroots(3,4,2);
```

"negative discriminant. no real roots."

Now in order to use the procedures in the 'mystuff' package, load a worksheet containing the package and execute the cell containing the definitions. Include the line with(mystuff) at the

bottom to complete the loading process.

5.10 Using the Packages that come with Maple

There are many packages that come with Maple, for example, **plots** and **plottools** are packages of words used to draw pictures, **student** is a package of words for calculus students, **numtheory** is a package of number theory words, **linalg** a package of linear algebra words, **geometry** a package of words for analytic geometry, and so on. To see the words in a package and at the same time make them usable, load the package using with

```
> with(plots);
```

```
Warning, the name changecoords has been redefined
```

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *cylinderplot*, *densityplot*, *display*, *display3d*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *graphplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *replot*, *rootlocus*, *semilogplot*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *sphereplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*]

Just like the regular vocabulary, these words all have help pages too. A good strategy when you have a task of some sort to do is to scan for words which might help you complete the task.

Example: Draw a blue ball and a red cone.

A solution. There might be some words in the plottools package to help here.

```
> with(plottools);
```

```
Warning, the name arrow has been redefined
```

[*arc*, *arrow*, *circle*, *cone*, *cuboid*, *curve*, *cutin*, *cutout*, *cylinder*, *disk*, *dodecahedron*, *ellipse*, *ellipticArc*, *hemisphere*, *hexahedron*, *homothety*, *hyperbola*, *icosahedron*, *line*, *octahedron*, *pieslice*, *point*, *polygon*, *project*, *rectangle*, *reflect*, *rotate*, *scale*, *semitorus*, *sphere*, *stellate*, *tetrahedron*, *torus*, *transform*, *translate*, *vrml*]

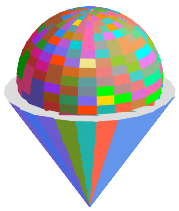
Yep. cone and sphere are the words we are looking for. What is their syntax? we can type

colors()

```
> example(cone);
```

This brings up the helpsheet for cone and goes to the examples section. copy the example and paste it into your worksheet.

```
> icecream := cone([0,0,-1],0.7,color=pink),
  sphere([0,0,0.1],0.6,color=gold):
  plots[display](icecream, scaling=constrained,
  style=patchnogrid,lightmodel=light2,orientation=[40,70]);
```



This does not solve our problem, but we can modify it to solve the problem. All we have to do is change colors.

```
> icecream := cone([0,0,-1],0.7,color=yellow),
  sphere([0,0,.1],0.6,color=blue):
  plots[display](icecream, scaling=constrained);
```

Note the use of the word **plots[display].** This is used to display several plots in the same picture. By using it as plots[display] instead of in the form display, it is not necessary to have loaded the plots package into vocabulary in order to use it. (If you say display(icecream) and have not loaded the plots package by saying with(plots) first, the output will be the input 'display(icecream)'.)

5.11 The linalg package

The linalg package of word is one of the most useful in Maple. To load the package, use **with(linalg);**

```
> with(linalg);
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
```

[*BlockDiagonal*, *GramSchmidt*, *JordanBlock*, *LUdecomp*, *QRdecomp*, *Wronskian*, *addcol*, *addrow*, *adj*, *adjoint*, *angle*, *augment*, *backsub*, *band*, *basis*, *bezout*, *blockmatrix*, *charmat*, *charpoly*, *cholesky*, *col*, *coldim*, *colspace*, *colspan*, *companion*, *concat*, *cond*, *copyinto*, *crossprod*, *curl*, *definite*, *delcols*, *delrows*, *det*, *diag*, *diverge*, *dotprod*, *eigenvals*, *eigenvalues*, *eigenvectors*, *eigenvects*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stackmatrix*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

Most of the algorithms in Ma 322 (as well as many that aren't) can be found in this list. For example, the dot product of two vectors can be computed using the word **linalg[dotprod].** If

you have loaded in the linalg package in a worksheet, you can use the short name **dotprod**.

```
> dotprod([2,1,3],[4,2,1]);
```

$$13$$

A vector can be defined in maple as a list or as a vector

```
> A := [3,2,3];
```

$$A := [3,\ 2,\ 3]$$

```
> B := vector([3,2,1]);
```

$$B := [3,\ 2,\ 1]$$

```
> dotprod(A,B);
```

$$16$$

To get the angle between two vectors, use **linalg[angle].** For example, if I want to know the angle between A and A+2*B

```
> ang:= angle(A,A+2*B) ;
```

$$ang := \arccos\left(\frac{27\sqrt{22}\sqrt{142}}{1562}\right)$$

To convert the angle to a decimal number use **evalf**

```
> evalf(ang);
```

$$0.2609862573$$

To convert the angle to degrees, we can multiple by 180/Pi

```
> evalf(180/Pi*angle(A,A+2*B));
```

$$14.95341105$$

A matrix is entered row by row. So, for example to define the 2 by 3 matrix whose top row is [3,1,5] and whose bottom row is [6,5,4]

```
> C := matrix(2,3,[3,1,5,6,5,4]);
```

$$C := \begin{bmatrix} 3 & 1 & 5 \\ 6 & 5 & 4 \end{bmatrix}$$

To get the norm of a vector, take the square root of the dotproduct of the vector with itself.

```
> sqrt(dotprod(A,A));
```

$$\sqrt{22}$$

Randmatrix is used to generate a matrix with some random entries.

```
> E:= randmatrix(2,3);
```

$$E := \begin{bmatrix} -85 & -55 & -37 \\ -35 & 97 & 50 \end{bmatrix}$$

To add two matrices of the same size use **matadd** or **&+** (together with **evalm**)

```
> matadd(C,E);
```

$$\begin{bmatrix} -82 & -54 & -32 \\ -29 & 102 & 54 \end{bmatrix}$$

```
> evalm(C &+ E);
```

$$\begin{bmatrix} -82 & -54 & -32 \\ -29 & 102 & 54 \end{bmatrix}$$

Row operations on matrices are important in linear algebra. One particular one is to add a scalar multiple of one row of a matrix to another row of that matrix. So, to add 5 times the first row of E above to the second row, use **addrow**

```
> addrow(E,1,2,5);
```

$$\begin{bmatrix} -85 & -55 & -37 \\ -460 & -178 & -135 \end{bmatrix}$$

Matrix multiplication is especially important operation also. Use **multiply** or **&*** (together with **evalm**) to do that

```
> F := matrix(3,2,[1,2,4,5,0,-3]);
```

$$F := \begin{bmatrix} 1 & 2 \\ 4 & 5 \\ 0 & -3 \end{bmatrix}$$

```
> multiply(E,F);
```

$$\begin{bmatrix} -305 & -334 \\ 353 & 265 \end{bmatrix}$$

```
> evalm(E &* F);
```

The best way to learn to use the words in linalg is to work problems in linear algebra. Over the course of a semester, you can become fairly proficient at using linalg to do (or check) the computation you need to do.

5.12 The MCtools Package

The **MCtools Package** does not come with Maple. It is a package of words written by us to facilitate the creation of homeworks for posting on WHS. It is in a constant stage of revision, with new words being added and old ones being refined. The convention is to put the package used to create a homework in a section at the top of the source homework worksheet. That way, you can be (pretty) sure that you can modify the homework if you want. Also, when changes are made to MCtools, the version date is changed, and the changes are made if at all possible so that things that worked in an older version still work in the new version.

In order to 'load' the MCtools package, you open the MCtools section, execute the input cell containing the definitions of all the words in the package, then close the section. Here is a listing of the words in the package.

```
> mctools();
```

```
MCtools, version Oct 15 2003.


Current choices are [ARRW, Axes, CARR, DL, DV, GP, GP2, GP3, Header,
MC_constants, MM, PA, PARAMS, PC, PP, PT, RANDANS, addlink,
addsectionhint, addvlink, ah_, av_, colors, fill, generator, hashang,
htmltable, lineit, maketable, mcpiecewise, mcprint, mctools, printit,
printspreadsheet, roundto, symbolize, tableit, tagit, vspace, zipit,
zipp].


"mctools(X)" gives syntax and defaults for "X"
```

There are several drawing words in the MCtools package.

MCtools Drawing words: ARRW, Axes, CARR, DL, DV, GP, GP2, GP3, MM, PA, PC, PP, PT, hashang and fill

These **drawing words** are written to help draw diagrams for problems. You can get help on the syntax of the words by using mctools. Thus to refresh your memory of how DL works, type

```
> mctools(DL);
MCtools, version Oct 15 2003.

DL(A,B,thknss=2,
styl=1,clr=blue,leftshrinkfactor=0,rightshrinkfactor=0,hashnum=3,hashl
ength=.3,hashspacing=.1,hashlocation=.5)
```

The first two arguments are points in the plane or in space and the remaining arguments are various options which you can change. We will get into drawing diagrams for problems as the need arises.

There are also several formatting words in the MCtools package.

MCtools text Formatting words: addlink, addsectionhint, addvlink, ah_, av_, **lineit** , maketable, mcpiecewise, **mcprint** , printit, printspreadsheet, roundto, symbolize, and vspace.

These are used to automate the production and improve the appearance of a WHS homework. Here again we won't cover these at this point. Brief help for each word is available via mctools.

```
> mctools(mcprint);
MCtools, version Oct 15 2003.

mcprint(strings,expressions)  converts all the inputs to a single
string and lprints it without the quotes.
```

Finally, the two Worksheet formatting words.

MCtools Worksheet formatting words: **tagit** and **zipit**

Don't like to tag homework problems by hand for WHS? tagit does that for you, leaving you to focus on the creative part of problem creation. Another advantage of using tagit is that it is a lot easier to make changes in your problem if they have been composed with tagit. And you will want to make changes.

```
> tagit("What is 2 + 3?",_AC(5));
QM_[0.05;5]
AH_[0]
What is 2 + 3?
AC_[5]
SKIP_
```

5.13 Making Movies

A **movie** (or animation) is a sequence of frames displayed one after the other. A well conceived animation can be of great use in gaining an understanding of some phenomena. They are also fun

to make. Here is an example:

Example: Have a blue ball move up out of a red cone.
A solution: We can start with the picture we drew before

```
> icecream := cone([0,0,-1],0.7,color=red),
  sphere([0,0,.1],0.6,color=blue):
  plots[display](icecream, scaling=constrained);
```

This is the first frame of our movie. What has to change in order for the ball to rise up out of the cone? the z-coordinate of the center of the ball. If we define our frame like so

```
> frame := t ->
  plots[display](cone([0,0,-1],0.7,color=red),
  sphere([0,0,t],0.6,color=blue), scaling=constrained);
> frame(2);
```

Now in order to see the ball clearly we have to change the orientation and put in some axes.

```
> frame := t ->
  plots[display](cone([0,0,-1],0.7,color=red),
  sphere([0,0,t],0.6,color=blue),
  scaling=constrained,axes=boxed,orientation=[50,70]);
> frame(2);
```

Now we can make a movie using **plots[display]** with the option **insequence=true**

```
> plots[display](seq(frame(i/2),i=1..5),insequence=true,scaling=constra
  ined);
```

To run the animation, activate the plot and press the play button.

5.14 Trouble Shooting Notes

Learning to use Maple can be an extremely frustrating experience, if you let it. There are some types of errors which occur from the beginning that can be spotted and corrected easily by a person fluent in Maple, so if you have access to such a person, use him or her.

Here are a few suggestions that may be of use when you're stuck with a worksheet that's not working like it should.

- **Important!** Loading an old Maple worksheet does not excute it. This is something you must do separately, either one cell at a time or all at once.

- **Also important!** Remember that you can rexecute any cell in the worksheet at any time. This can lead to confusion about what you did last, because it may not be what is just above.

- **Use help:** There is a help sheet with examples for every Maple word. A quick read thru will often clear up syntax problems. One very common early mistake is to leave out the parentheses around the inputs of a word. For example, typing

```
> plot x^2;
```

```
Error, missing operator or ';'
```

will get you a syntax error, because you left out the parentheses.

- **The maple prompt is '>'** . You can begin entering input after it. Make sure you are typing into an input cell, if you are expecting output.

- **End maple statements with a semicolon ';' or colon ':'** . Maple suppresses output when the command line ends with a colon. Maple does not process until it comes to a (semi)colon. If you are getting no output when you should be, try feeding in a semicolon. This often works.

```
> p := expand((a+b)^5);
```

$$p := a^5 + 5\,a^4\,b + 10\,a^3\,b^2 + 10\,a^2\,b^3 + 5\,a\,b^4 + b^5$$

```
> p;
```

$$a^5 + 5\,a^4\,b + 10\,a^3\,b^2 + 10\,a^2\,b^3 + 5\,a\,b^4 + b^5$$

- **When in doubt, put in parentheses.** For example, (x+3)/(x-3) is very different from x+3 / x-3 .

```
> (x+3)/(x-3),x+3 / x-3 ;
```

$$\frac{x+3}{x-3},\ x + \frac{3}{x} - 3$$

- **Make sure your variables are variable**. You may have assigned a value, say 3, to x in a previous problem. To make x a variable again, type x := 'x': . Use the forward quote ' key, just below the double quote " here. If you forget this, strange things might happen. One way to handle this is to keep an input cell of variables used.

```
> x := 3;
```

$$x := 3$$

```
> x := 'x';
```

$$x := x$$

- Use **restart;** By typing restart; in an input cell and pressing enter, you clear all assignments, and start with a clean slate. This fixes a lot of problems fast, but you will need to re-execute input cells.

```
> hello := yes;
```

$$hello := yes$$

```
> hello;
```

$$yes$$

```
> restart;
> hello;
```

$$hello$$

- **Are you using the correct quote symbol?** In Maple, the **forward quote '** is used to suppress evaluation. The **back quote ‘** is used to form names or symbols. The **double quote "** is used to define strings.

```
> ‘this is a long name‘ := 12;
```

$$this\ is\ a\ long\ name := 12$$

```
> ‘this is a long name‘;
```

$$12$$

```
> check := "do not worry";
```

$$check := \text{“do not worry”}$$

```
> check;
```

$$\text{“do not worry”}$$

- **Do not forget to end loops with od**, ‘if‘ statements with **fi** , and procedures with **end** . If you start a loop with do , Maple does not begin processing until it finds the end of the loop, which is signaled by the word **od** ; The same applies to the if .. then ... fi; and proc ... end; contructions. If you are getting no output when you should be, try feeding an od; , fi; , or end; This often works.

- **Unwanted output?: Is there output you need but don't want to see? Use a colon ‘:‘ instead of a semicolon to end the Maple statement which generates the output.**

- **Use printlevel := 10;** if you want to see what Maple is doing behind the scenes when you give it a command. If you want to see more, use **printlevel** := 50 or higher. Often by inspecting the output when printlevel is greater than 1 (the default), you can discover what is ailing your worksheet.

now is the time

```
> unapply(x^2-100,x);
```

$$x \rightarrow x^2 - 100$$

```
> printlevel:= 10;
```

$$printlevel := 10$$

```
> unapply(x^2-200,x);
{--> enter unapply, args = x^2-200, x
```

$$x := [x]$$
$$xtra := []$$
$$num := []$$
$$x := x$$
$$op1 := x \rightarrow x^2 - 200$$
$$x \rightarrow x^2 - 200$$

```
<-- exit unapply (now at top level) = proc (x) options operator,
arrow; x^2-200 end proc}
```

$$x \rightarrow x^2 - 200$$

```
> printlevel:=1;
```

$$printlevel := 1$$

- **Are all words you are using defined?** Have you executed the input cells containing the procedures you need? Have you loaded the packages containing the words you need? Undefined words are simply returned by Maple. Hint: When using words from packages in procedure definitions, use the full name, e.g, use plots[display] instead of display.

```
> thisword := proc(x) x^2 + 10 end;
```

$$thisword := \mathbf{proc}(x)\, x^2 + 10 \ \mathbf{end\ proc}$$

```
> thisword(3);
```

$$19$$

```
> restart;
> thisword(3);
```

$$\text{thisword}(3)$$

- **Use debug**. If you have defined a word, say ' **thisword** ' and it does not do what you want, you can often discover the error by typing **debug(something);** in an input cell and pressing the enter key. When you use the word again, its behind the scene computations are printed out for your inspection.

```
> thisword := proc(x) x^2 + 10 end;
```

$$thisword := \mathbf{proc}(x)\, x^2 + 10 \ \mathbf{end\ proc}$$

```
> debug(thisword);
```

$$thisword$$

```
> thisword(3);
```

```
{--> enter thisword, args = 3
```

19

```
<-- exit thisword (now at top level) = 19}
```

19

```
> undebug(thisword);
```

thisword

```
> thisword(3);
```

19

- **Want to see a word definition?** Say you want to see how plot works. Type **interface(verboseproc=2);** in an input cell and press enter. Then type print(plot);

```
> interface(verboseproc=2);
> print(plot):
```

6 Automating the tagging of a problem with tagit.

We want to illustrate one way to make the process of tagging a problem automatic. This way uses an MCtools word called **tagit tagit** .

We have defined it so that you type in the problem, and tell what the answer format and the answer is, and it prints the problem out with the appropriate tags inserted. This method has many advantages over hand-tagging problems. It is much easier to maintain and modify a problem which has been composed with tagit. Here are a few simple examples to get started.

Example: What is 2 + 3?
Answer: 5.

If we wanted to tag this problem with a **constant answer constant answer** format using tagit, we would type the statement of the problem "What is 4 + 3?", and follow it with _ac(7), which says to use the constant answer format and the correct answer (to be put in the qm line) is 7.

```
> tagit("What is 4 + 3?",_AC(7));
```

```
QM_[0.05;7]

AH_[0]

What is 4 + 3?

AC_[5]

SKIP_
```

To change it to a **selection box selection box** multiple choice problem, all we would need to do is change the answer format to _AS([7,1,5,2]). The correct answer is listed first amonst the alternatives.

```
> tagit("What is 4 + 3?",_AS([7,1,5,2]));
```

Note: The alternatives are scrambled in the tagged output of the problem, so the correct answer does not always appear first.

Another common answer format is the **row of radio buttons row of radio buttons** for multiple choice problems. To change to that, change _AS to _AL([7,1,5,2]). The first alternative answer you list is taken as the correct answer.

```
> tagit("What is 4 + 3?",_AL([7,1,5,2]));
```

What about **word answers word answers** ? That is handled with _AW

Example: What is a 3 letter word for a feline pet?
Answer: cat or Cat or CAT

To compose this with tagit we would type:

```
> tagit("What is a 3 letter word for a feline
  pet?",_AW("cat#Cat#CAT"));
```

6.1 Using tagit to tag a problem with a diagram.

To put a diagram into a problem tagged by tagit, you can give a name to the diagram and put its name as one of the inputs to tagit.

Here is a right triangle. We have given names to its vertices. We have used DL (draw line) from MCtools to draw the sides of the triangle. We then used display from the plots package to display all the sides together. We name the drawing TRNG.

```
> A:=[0,0]:
> B:=[2,0]:
> C:=[2,1]:
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),
  DL(A,C,thickness=3,linestyle=1,color=blue),
  DL(B,C,thickness=3,linestyle=1,color=blue):
> TRNG:=plots[display](LNS,scaling = constrained,axes=none):
> TRNG;
```

Now we can make up a problem about this triangle. We put the name of the diagram, TRNG, in front of the statement of the problem.

Problem. If the bottom leg of the right triangle is 4 and the hypotenuse is 5, then the vertical leg is

Answer: 3

```
> tagit(TRNG,"If the bottom leg of the right triangle is 4 and the
  hypotenuse is 5, then the vertical leg is",_AC(3));
```

If we wanted to put the diagram in the middle of the statement of the problem, just break the statement into two and put it in between.

```
> tagit("Suppose bottom leg of the right triangle is 4 and the
  hypotenuse is 5.",TRNG, "What is the length of the vertical
  leg?",_AC(3));
```

Task: Use tagit to tag some problems. Then install the homework.

6.2 Composing problems with tagit: answers formats and options

You can use a programming language such as Maple to ease the tagging burden and also reduce error. MCtools has a word **tagit tagit** which provides one easy way to do this. What tagit does is take certain inputs and then outputs a complete tagged WHS problem starting with the QM line and ending with a SKIP. Tagit can be written in two different forms.

The **options form options form** is a sequence of lists, each of which is a list of equations. Each list is a separate problem part and each equation is one of the options: problem=, anstype=,rightanswers=, answers=, txtboxsize=, precision=, etc. The **freestyle form freestyle form** (which is the preferred form) is a sequence of strings, expressions, and lists: each string, expression or list is treated as part of the current problem. Expressions of the form _XX(answer,other stuff) where _XX is one of _AC, _AW, _AS, _AL, _ALlabelled, _AB, _ABlabelled, _AR, _AF, _AE, _AT, _ATspread, _ATcross, or _AX signal that an answer format tag is to be inserted into the problem. Each answer format tag is explained below and some examples are given.

The freestyle form is newer than the options form, and allows a more natural formatting of problems. We will use the freestyle form exclusively here, but many old homeworks are composed using the options form, so you may need to be familiar with it. With the freestyle form, you don't use the options problem=, aftertext=, anstype=, rightanswers=, and answers=. The other options have default values which you will be able to change as needed. There are lots of examples below.

Tagit reads its arguments from left to right and acts on them in turn. Strings are printed unformatted as is in a left justified manner. These contain the text used to state the problems. Expressions, such as x^2 + 3, are not formatted or evaluated if they are inside a string. An expression which is outside a string will be evaluated and printed in a formatted evaluated form centered on a line by itself.

In the sections below, we give example of how to compose problems using the various answer formats and options available with tagit.

6.2.1 tagit syntax

If you forget how to use tagit, you can get the **syntax of tagit syntax of tagit** by just typing mctools(tagit); in an input cell and pressing return.

```
> mctools(tagit);

MCtools, version June 1 2004.


tagit(strings,expressions,lists,answer formats,option equations)

strings are mcprinted, expressions are printed,  the ops of lists are
mcprinted,

answer formats produce answer tags:   They are

_AC(answer,options) where answer is the correct answer to the
question, options are option equations, and aftertext are strings,
lists and expressions to be put into an aftertext option. Two options
which are commonly set are txtboxsize and precision. The defaults for
txtboxsize and precision are 5 and .05.

_AW(answers)  where answers is a list of possible spellings of the
word, or a string of #-separated possible spellings of the word.

_AS(answers) This is a multiple choice answer format.    producing a
scroll box with alternatives

_AS(answers) where answers is a list of alternative answers with the
correct answer first.

_TP() Use this if you want to end a problem part after an answer
description has been specified.   Any options given before the problem
terminator are applied to that part.  Any strings and expression are
put in the aftertext option to that part.

_AF(f,options)   where f is an algebraic expression with one variable,
and options are txtboxsize and precision  the defaults for txtboxsize
and precision are 15 and .05.

_AI(f,options)   just like _AF, that   WHS counts it as correct if it
differs by a constant within a specified error set by the precision
option.  It is most often used to test answers to antidifferentiation
problems.

_AE(f,options)  just like _AF, except that more than one varible is
permitted.

_AL(answers) where answers is a list of alternative answers with the
correct answer first. a row of radio buttons is produced
```

```
_AR(answers) like _AL, except that the radio buttons occur vertically

_ALlabels(answers)  just like AL execpt the alternative are labelled
and put above the row of radio buttons

_AB(answers)  like AL except that more than one answer can be right.
right answers in the list of answers are enclosed in square brackets.

_ABlabels(answers) like AB except that the alternatives are labelled
and put above the row of checkboxes.

_AT(list of lists)  creates a tables of  AC boxes.   entries which are
enclosed in square brackets are replaced with AC boxes.

_ATcross(list of lists)  creates a table of AC, AF, AI, AE, AW, and AS
boxes.  each entry is printed or mcprinted unless it is a list [X,
answers] or [answers] where X is one of C,F,I,E,W, or S and answers is
written in the form expected by AC, AF, AI, AE, AW or AS.


The option equations are

standards=""  set to a comma separated list of labels identifying a
list of standards tested (from some list of course standards or
state-mandated standards)

prextext=NULL, set to a string or list to put text before question

brks=0, for no breaks, 1 for break after, 2 for break before, 3 for
break before and after the text preceding the answer box.  To suppress
the header tag, use brks=4

precision=‘.05‘,  measures error in response, used with AC, AF, AI,
AE, AT, ATspread, ATcross

txtboxsize=8,   number of spaces in the answer box  used with AC, AF,
AI, AE, AT, ATspread, ATcross_

randomize=no, set to yes if you want the alternatives scrambled..

matsize=[1,5], sets the array the alternatives are shown in _ALlabels
and _ABlabels

Alabels=[A,B,C,D,E,F,G,H], sets the labels used in _ALlabels and
_ABlabels

Flabels="color=red",  sets the label color in _ALlabels and _ABlabels

hidden=no  set to yes to hide the answer on feedback sheet.

inline=no,  set to [yes] for one line questions;[yes,opts1] opts1 is
list of options to Lineit; [yes,opts1,opts2] opts2 is list of options
to Lineit (used in aftertext)
```

6.2.2 The numberbox format _AC : numerical answer

This is an answer format that requires the student type in a number, in calculator syntax. The structure is

_AC _AC (answer,option) where answer is the correct answer to the question, options are option equations The defaults txtboxsize and precision are 5 and .05.

Here is an example.

```
> tagit( "Bill can mow a yard in 3 hours.  Jim can mow the same yard in
  5 hours.    How many hours does it take Bill and Jim to mow the yard
  together?",_AC(1/(1/3+1/5))):

QM_[0.05;15/8]
```

```
AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together?

AC_[5]

SKIP_
```

Here is the same example with the options **txtboxsize=** **txtboxsize=** and **precision=** **precision=** changed from the default values. Note one of the options occurs before the answer format, and the other is an argument to it. Either place is ok.

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each
  other?",txtboxsize=10,_AC(1/(1/3+1/5),precision=‘.1‘));
```

```
QM_[.1;15/8]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other?

AC_[10]

SKIP_
```

Here is the same example with the reference to not interferring with each other moved after the answer box. We have put a return \n in the string after the answer format in order to get it on a separate line.

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.  How many hours does it take Bill and Jim to mow the yard
  together?",txtboxsize=11,_AC(1/(1/3+1/5)),"
  Assume they do not interfere with each other.");
```

```
QM_[0.05;15/8]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together?

AC_[11]

Assume they do not interfere with each other.

SKIP_
```

You can have as many numberboxes in a problem as you want. In order for the problem to be counted right, each part has to be answered correctly. This one will put them all inline.

```
> tagit("3 + 4 =",_AC(7),"   5 + 12 =",_AC(17),"   4 + 1 =",_AC(5));
```

```
QM_[0.05;7;17;5]

AH_[0]

3 + 4 =

AC_[5]
```

```
AH_[0]
   5 + 12 =
AC_[5]
AH_[0]
   4 + 1 =
AC_[5]
SKIP_
```

This usage puts the question one above the other.

```
> tagit("3 + 4 =",_AC(7),brks=before,"5 + 12 =",_AC(17),"4 + 1
  =",_AC(5));

QM_[0.05;7;17;5]
AH_[0]
3 + 4 =
AC_[5]
AH_[2]
5 + 12 =
AC_[5]
AH_[2]
4 + 1 =
AC_[5]
SKIP_
```

6.2.3 Use of brackets to group text and numerical values.

Numerical values which are inserted between two text strings are printed by default. If you enclose the text and numbers in square brackets the numbers will be mcprinted inline. You cannot put plot structures or answer formats inside brackets.

```
> tagit("What is ",3," + ",5,"?",_AC(8));

QM_[0.05;8]
AH_[0]
What is
                                        3
 +
                                        5
?
AC_[5]
SKIP_
> tagit(["What is ",3," + ",5,"?"],_AC(8));

QM_[0.05;8]
```

```
AH_[0]

What is 3 + 5?

AC_[5]

SKIP_
```

6.2.4 Use of Line to format the lines of a problem.

I you have a mathematical expression or plot structures in your problem statement, then you will want to try using **Line Line** to format the lines in your problem statement. The advantage is that you do not have to format any inline math by hand. You will still have to resize the pictures.

```
> mctools(Line);
```

```
MCtools, version June 1 2004.

Line(expression sequence), where expression sequence is a list of
strings, mathexpressions, pictures, and answer formats to be put on a
single line. It is your responsibility to decide whether the line is
not too long. Any options to an answer format must be put in the
format.
```

```
> tagit(Line(2+`3 =`,_AC(5,txtboxsize=7,precision=1)),"Hint:  Count up
  3, starting at 2.");
```

```
QM_[1;5]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_
```

$$2 + 3 \; =$$

```
lt_/td gt_

lt_td gt_

AC_[7]

lt_/td gt_

lt_/tr gt_lt_/table gt_

Hint: Count up 3, starting at 2.

SKIP_
```

6.2.5 The entry format _AW : word answers

This an answer format that requires the student type in a word or short phrase.

_AW _AW (answers) where answers is a string of #-separated possible spellings of the word.

```
> tagit("The name for the large african or indian mammal with a trunk
  and floppy ears is
  ",txtboxsize=24,_AW("elephant#ELEPHANT#Elephant"));
```

```
QM_[0.05;elephant#ELEPHANT#Elephant]
```

```
AH_[0]

The name for the large african or indian mammal with a trunk and
floppy ears is

AW_[24]

SKIP_
```

6.2.6 The multiple choice format _AS : selection box answers

This is a multiple choice answer format. The format is

_AS _AS (answers) where answers is a list of alternative answers with the correct answer first. If the first alternative is not correct, put brackets around the correct one

```
> evalf(1/(1/3+1/5));
```

1.875000000

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?  Select the
> most nearly correct answer.",_AS([1.875,2.125,4,"None of the
  others"]));
```

```
QM_[0.05;1.875]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other? Select the most nearly
correct answer.

AS_[2.125;None of the others;4;1.875]

SKIP_
```

By default the first alternative in the list of alternatives is assumed the right answer. You can change this by setting the rightanswers option to the correct answer, or by enclosing the right answer in square brackets. You can also fix the order of the alternatives in the list to be the one given by setting the **randomize= randomize=** option to no.

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?  Select the
> most nearly correct answer.",_AS([2.125,[1.875],4,"None of the
  others"],randomize=no));
```

```
QM_[0.05;1.875]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other? Select the most nearly
correct answer.

AS_[2.125;1.875;4;None of the others]
```

```
SKIP_
```

6.2.7 problems with several parts

The three answer format pretags _AC, _AW, and _AS serve very well for most problems. These and the others below can be mixed freely in the same question, to get a multiple part problem.

Note: In order for a multiple part problem to be answered correctly, each part must be answered correctly. Also, the format _AX cannot be used in a multiple part problem.

```
> tagit("The name for the large african or indian mammal with a trunk
  and floppy ears is ",_AW("elephant#ELEPHANT#Elephant"),brks=before,
  "What is the name that the curved front teeth that these mammals often
> have?", _AW("tusk#tusks#Tusk#Tusks",txtboxsize=8),brks=before,
  "How many feet does an elephant have?",_AC(4),brks=before,"An elephant
  has ",_AS([1,2,3,4])," tusks.");
```

```
QM_[0.05;elephant#ELEPHANT#Elephant;tusk#tusks#Tusk#Tusks;4;1]

AH_[0]

The name for the large african or indian mammal with a trunk and
floppy ears is

AW_[15]

AH_[2]

What is the name that the curved front teeth that these mammals often
have?

AW_[8]

AH_[2]

How many feet does an elephant have?

AC_[5]

AH_[2]

An elephant has

AS_[3;1;2;4]

 tusks.

SKIP_
```

6.2.8 The entry format _AF : function of 1 variable

_AF _AF is an answer format that requires the student type in an expression, in calculator syntax, with one variable. For example, on the lawnmowing problem, if we let x be the time that it takes Bill to mow the yard then the expression 1/(1/x + 1/5) represents the time it takes to mow the yard when both Bill and Jim are mowing. The WHS formatting of a question designed to elicit this expression (or an equivalent expression) from the student is shown below:

Note: the defaults for txtboxsize and precision are 15 and .05.

```
> tagit("Bill can mow a yard in x hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each
  other?",txtboxsize=20,_AF(1/(1/x+1/5)));
```

```
QM_[0.05;1/(1/x+1/5)]

AH_[0]

Bill can mow a yard in x hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other?

AF_[20;x;6;.1;1]

SKIP_
```

6.2.9 The entry format _AI : integrals (function of 1 variable to within a constant)

The **_AI _AI** answer format is just like **_AF _AF** , that WHS counts it as correct if it differs by a constant within a specified error set by the precision option. It is most often used to test answers to antidifferentiation problems.

```
> tagit(Line("Find a function whose derivative is  ",3*x^2 + 4*x +1),
  _AI(x^3+2*x^2 +x,txtboxsize=20));
```

```
QM_[0.05;x^3+2*x^2+x]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_

Find a function whose derivative is

lt_/td gt_

lt_td gt_
```

$$3\,x^2 + 4\,x + 1$$

```
lt_/td gt_

lt_/tr gt_lt_/table gt_

AI_[20;x;6;.1;1]

SKIP_
```

6.2.10 The entry format _AE : function of 1 or more variables

This is an answer format that requires the student type in an expression, in calculator syntax, with one or more variables. For example, on the lawnmowing problem, if we let x be the time that it takes Bill to mow the yard and y be the time it takes Jim to mow the yard, then the expression

$1/(1/x + 1/y)$ represents the time it takes to mow the yard when both Bill and Jim are mowing. The WHS formatting of a question designed to elicit this expression (or an equivalent expression) from the student is shown below:

Note: the defaults for txtboxsize and precision are 15 and .05. You can change this by resetting them inside the list headed by **_AE _AE** .

```
> tagit("Bill can mow a yard in x hours.  Jim can mow the same yard in y
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?    Express
> your answer in terms of x and y, using calculator syntax.",
  _AE(1/(1/x+1/y)));
```

```
QM_[0.05;1/(1/x+1/y)]

AH_[0]

Bill can mow a yard in x hours. Jim can mow the same yard in y hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other?  Express your answer
in terms of x and y, using calculator syntax.

AE_[15;2;x;y;5;.1;1;.1;1]

SKIP_
```

We can change the evaluation lattice for the function in two ways: By a list of three numbers after the answer, like so: [6,1,2] tells WHS to test the difference between students answer and the QM line answer at 6 randomly chosen values betwee 1 and 2. If the difference is always less than the precision the students answer is counted correct.

```
> tagit("Bill can mow a yard in x hours.  Jim can mow the same yard in y
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each
  other?",_AE((x*y)/(x+y),[6,1,2]),"hours");
```

```
QM_[0.05;x*y/(x+y)]

AH_[0]

Bill can mow a yard in x hours. Jim can mow the same yard in y hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other?

AE_[15;2;x;y;6;1;2;1;2]

hours

SKIP_
```

or by a list containing a string, like so: [”4;2;3;1;3”] tells WHS to evaluate at 4 randomly chosen points with the first coordinate between 2 and 3 and the second coordinate between 1 and 3.

```
> tagit("Bill can mow a yard in b hours.  Jim can mow the same yard in j
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each
  other?",_AE((b*j)/(b+j),["4;2;3;1;3"]),"hours.");
```

```
QM_[0.05;b*j/(b+j)]

AH_[0]

Bill can mow a yard in b hours. Jim can mow the same yard in j hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other?
```

```
AE_[15;2;j;b;4;2;3;1;3]

hours.

SKIP_
```

6.2.11 The multiple choice format _AL: radio button answers

This is a multiple choice answer format. Use with numerical or word answers. The format is

_AL _AL (answers) where answers is a list of alternative answers with the correct answer first. Note: If you want an answer other than the first one in the list to be the correct one, put brackets around it.

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?  Select the
> most nearly correct answer.",_AL([1.875,2.125,4,‘None of the
  others‘]));
```

```
QM_[0.05;1.875]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other? Select the most nearly
correct answer.

AL_[1.875;None of the others;2.125;4]

SKIP_
```

By default the first alternative in the list of alternatives is assumed the right answer. You can change this by putting brackets around the correct answer, or by specifiying the right answer with the option rightanswers= option in the answer format. You can also fix the order of the alternatives in the list to be the one given by setting the randomize option to no.

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?  Select the
> most nearly correct answer.",brks=3,_AL([2.125,[1.875],4,‘None of the
  others‘],randomize=no));
```

```
QM_[0.05;1.875]

AH_[3]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other? Select the most nearly
correct answer.

AL_[2.125;1.875;4;None of the others]

SKIP_
```

6.2.12 The multiple choice format _ALlabels : radio button answers with labels

This is a multiple choice answer format. Use with numerical, symbolical, or string answers. It is especially useful when the alternatives contain formatted mathematics or are very long. The a The format is

_ALlabels _ALlabels (answers) where answers is a list of alternative answers with the correct answer first. **Note:** The text output in source is pretty ugly. That is because of the html pre-tags which are used to set up a table in the posted version of the problem.

```
> A:='A':
> tagit("Which is the correct formula for the area of a trapezoid with
  base b, top t, and height h?",_ALlabels([A = h*(t+b)/2,A=h*(t+b),A=
  b*h/2,"None of the others"]));
```

```
QM_[0.05;C]

AH_[0]

Which is the correct formula for the area of a trapezoid with base b,
top t, and height h?

lt_table gt_lt_tr gt_lt_td valign="middle" gt_ lt_font color="red" gt_
lt_b gt_A. lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$A = h\,(t + b)$$

```
 lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; B.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$A = \frac{b\,h}{2}$$

```
lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; C.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$A = \frac{h\,(t + b)}{2}$$

```
lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; D.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_

None of the others

lt_/td gt_

lt_/tr gt_lt_/table gt_

AL_[A;B;C;D]

SKIP_
```

6.2.13 The multiple choice format _AR : a vertical column of radio button answers

This is a multiple choice answer format. Use with numerical or word answers. The format is

_AR _AR (answers,options if wanted,aftertext if wanted) where answers is a list of alternative answers with the correct answer first. The position of the correct answer is scrambled by WHS and cannot be turned off.

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?  Select the
> most nearly correct answer.",_AR([1.875,2.125,4,"None of the
  others"]));
```

```
QM_[0.05;2]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other? Select the most nearly
correct answer.

AR_[35135066117]

None of the others

AR_[35135066117]

1.875

AR_[35135066117]

2.125

AR_[35135066117]

4

SKIP_
```

6.2.14 The multiple choice format _AB : check box answers

This is a multiple choice answer format. Use with numerical or word answers. The format is

_AB _AB (answers) where answers is a list of alternative answers. You indicate the right answers by enclosing each one in square brackets.

The output is a row of check boxes

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?  Select the
> most nearly correct answer.",_AB([2.125,[1.875],4,"None of the
  others"]));
```

```
QM_[0.05;;1.875;;]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other? Select the most nearly
correct answer.
```

```
AB_[2.125;1.875;None of the others;4]

SKIP_
```

6.2.15 The multiple choice format _ABlabels : check box answers with labels

This is a multiple choice answer format. Use with numerical or word answers. The format is

_ABlabels _ABlabels (answers) where answers is a list of alternative answers. You indicate the right answers by enclosing each one in square brackets.

```
> tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
  together, assuming they do not interfere with each other?  Select the
> most nearly correct answer.",_ABlabels([[1.875],2.125,4,"None of the
  others"]));
```

```
QM_[0.05;;;;D]

AH_[0]

Bill can mow a yard in 3 hours. Jim can mow the same yard in 5 hours.
How many hours does it take Bill and Jim to mow the yard together,
assuming they do not interfere with each other? Select the most nearly
correct answer.

lt_table gt_lt_tr gt_lt_td valign="middle" gt_ lt_font color="red" gt_
lt_b gt_A. lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

2.125

```
 lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; B.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

4

```
lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; C.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_

None of the others

lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; D.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

1.875

```
lt_/td gt_

lt_/tr gt_lt_/table gt_

AB_[A;B;C;D]
```

```
SKIP_
```

6.2.16 The entry format _AX : handgraded answers

This format is for questions that you want to get short answers to. It is not graded by WHS, but is archived so that you can inspect it if you download the responses to the homework. Note: This format must be used alone in its own question headed by a QN line. Otherwise, the question will not be treated correctly by WHS. To grade an **_AX _AX** answer , the instructor goes to Access Records for the class and then clicks on Assignment submission. For the particular assignment, one can click on Display and then click for each student to display submission of the student. There is a Right/Wrong box for each problem to be hand graded

```
.
> tagit("What is the correct definition of a triangle?    Write your
  answer in complete sentence(s).",_AX());

QN_[0.05;Your answer will be hand graded.]

AH_[0]

What is the correct definition of a triangle?  Write your answer in
complete sentence(s).

AX_[4]


SKIP_
```

6.2.17 The answer format _AT

This format implements the table format described in the Author documentation.

_AT _AT (list of lists, (opts) where list of list is the complete table you want to display. The numerical entries you want the student to fill in are specified by enclosing them in square brackets.

```
> tagit("Let f be the function defined by ",f(x)=x^2+2,"Complete the
  table of function values.",_AT([[2,[6]],[3,[12]]]));

QM_[0.05;6;12]

AH_[0]

Let f be the function defined by
```

$$f(x) = x^2 + 2$$

```
Complete the table of function values.

AT_[5;2;2;2;;3;]

SKIP_
```

6.2.18 The answer format _ATcross

This answer format is like AT except that it allows the AW AS AF and AE formats in addition to the AC format. **_ATcross(listoflists,options)** **_ATcross(listoflists,options)** . If an entry in the listoflists is a not a list then it is displayed. If it is a list with 2 elements, and the second element is a color name, then the cellcolor is set to that and the first element is printed in the cell. If the entry is a list with only one element, the element is an answer and the answer format is AC if the element evalfs to a number, AW if the element is a string, AS if the element is a list, AF if the element is an expression with one indeterminant, and AE if the element is an expression with more than one indeterminant. Any entry not of the above type is printed in the cell with the default cellcolor. The default cellcolor can be changed to say yellow with the option cellcolor="yellow". Another option peculiar to ATcross is **tableoptions= tableoptions=** . Set this to "border=1" to make the border visible.

```
> b:='b':a := 'a':
> tagit("Select the correct reason.",_ATcross([[a*(b+c)=a*b +
  a*c,[[distributive,associative,commutative]]],[a*[b*c]=[a*b]*c,[[assoc
  iative,distributive,commutative]]]]));
```

```
QM_[0.05;distributive;associative]

AH_[0]

Select the correct reason.

lt_table border="0" gt_

lt_tr gt_

lt_td bgcolor="white" gt_
```

$$a\,(b+c) = a\,b + a\,c$$

```
lt_/td gt_

lt_td bgcolor="white"gt_

AS_[distributive;associative;commutative]

lt_/td gt_

lt_/tr gt_

lt_tr gt_

lt_td bgcolor="white" gt_
```

$$a\,[b\,c] = [a\,b]\,c$$

```
lt_/td gt_

lt_td bgcolor="white"gt_

AS_[commutative;distributive;associative]

lt_/td gt_

lt_/tr gt_

lt_/table gt_

SKIP_
```

The output from ATcross is rather ugly in source, because of all the html pretags that are inserted to put things in a table when it is posted to WHS. Here is an example of a simple language question.

```
> tagit("Fill in the table:",
  _ATcross([["English",["German",grey]],
  [["one",pink],["eine#Eine"]],
> [["two",turquoise],["zwei#Zwei"]],
  [["three#Three"],"drei"]],txtboxsize=10,cellcolor=yellow)
  );
```

Here is an example showing how to use ATcross to make a table of function values. We use the option **tableoptions= tableoptions=** to make the border of the table visible.

```
> tagit("Fill in the
  table:",tableoptions="border=\"1\"",cellcolor=blue,txtboxsize=10,
  _ATcross([[["   Rule",yellow],["times 2 ",yellow]],
> [["  Input",yellow],["  Output",yellow]],
  [3,[2*3]],
  [4,[2*4]],
> [5,[2*5]],
  [c,[2*c]],
  [[(x+y)],2*(x+y)],
  [10,[10*2]]]));
```

6.2.19 The option standards=

The **standards= standards=** option can be used to align a homework with a set of state standards, or with your own standards. Use a comma separated list for your standards. Here's an example.

```
> tagit(standards=["Can add two digit numbers","Can divide by 2"],
  Line("What is ",(‘23 + 45‘)/2,"?  Answer:",_AC(68/2)));
```

```
QM_[0.05;34][Can add two digit numbers;Can divide by 2]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_

What is

lt_/td gt_

lt_td gt_
```

$$\frac{23 + 45}{2}$$

```
lt_/td gt_

lt_td gt_

? Answer:

lt_/td gt_

lt_td gt_

AC_[5]

lt_/td gt_
```

```
lt_/tr gt_lt_/table gt_

SKIP_
```

6.2.20 The option hidden=

The **hidden= hidden=** option can be used to suppress the answer from the feedback sheet student get when they submit a homework for grading. All it does is append an N to QM_ **Note.** This option can be implemented on the homework level by setting a deadline on the homework in WHS.

Here's an example of the use of the hidden= option

```
> tagit("What is 22 + 45?",_AC(67,hidden=yes));

QM_N[0.05;67]

AH_[0]

What is 22 + 45?

AC_[5]

SKIP_
```

6.2.21 The option brks=

This option is used to insert answer headers into the problem. These control where WHS puts line breaks in the text just before an answer box. This enables one to string answer boxes inline or have them arranged vertically, etc. By default, tagit inserts an AH_[0] into the problem. By inserting the option brks=1 or **brks=after brks=after** , you can modify this to an AH_[1]. Similarly **brks= brks=** 2 or brks=before and brks=3 or brks=afterbefore modify the answer header appropriately. Any other value will suppress the answer header altogether.

```
> tagit("What is 22 + 45?",_AC(67));

QM_[0.05;67]

AH_[0]

What is 22 + 45?

AC_[5]

SKIP_
> tagit("What is 22 + 45?",_AC(67),"What is 3 - 2.",_AS([1,2,3]),
  brks=after,"What is 22 + 45?",_AC(67),"What is 3 - 2.",_AS([1,2,3]),
  brks=before,"What is 22 + 45?",_AC(67),"What is 3 - 2.",_AS([1,2,3]),
> brks=beforeafter,"What is 22 + 45?",_AC(67),"What is 3 -
  2.",_AS([1,2,3]));

QM_[0.05;67;1;67;1;67;1;67;1]

AH_[0]

What is 22 + 45?

AC_[5]
```

```
AH_[0]

What is 3 - 2.

AS_[1;2;3]

AH_[1]

What is 22 + 45?

AC_[5]

AH_[1]

What is 3 - 2.

AS_[1;2;3]

AH_[2]

What is 22 + 45?

AC_[5]

AH_[2]

What is 3 - 2.

AS_[1;3;2]

AH_[3]

What is 22 + 45?

AC_[5]

AH_[3]

What is 3 - 2.

AS_[2;1;3]

SKIP_
```

6.2.22 The option inline=

This option can be used place formatted mathematics and diagrams inline with text. When you set **inline=yes inline=yes** , the strings, expressions and plot structures preceeding the answer format are put into an html table. It is principally used to format short lines with one or two expression, or put text to the right or left of a diagram. Note: this option has been deprecated by the word Line.

```
> tagit(Line("What is " ,1/‘2‘ + 1/‘3‘,"?",_AC(1/2+1/3)));

QM_[0.05;5/6]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_

What is

lt_/td gt_

lt_td gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_
lt_td gt_
?
lt_/td gt_
lt_td gt_
AC_[5]
lt_/td gt_
lt_/tr gt_lt_/table gt_
SKIP_
```

inline=yes does not put answer boxes inline. To do that you could use ATcross or Line.

```
> tagit(_ATcross([[ "What is " ,1/‘2‘ + 1/‘3‘,"?",[1/2+1/3]]]));
```

```
QM_[0.05;5/6]
AH_[0]
lt_table border="0" gt_
lt_tr gt_
lt_td bgcolor="white" gt_
What is
lt_/td gt_
lt_td bgcolor="white" gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_
lt_td bgcolor="white" gt_
?
lt_/td gt_
lt_td bgcolor="white"gt_
AC_[15]
lt_/td gt_
lt_/tr gt_
lt_/table gt_
SKIP_
```

6.2.23 The options matsize=, Flabels=, and Alabels=

These options are all used with the answer formats ABlabels, ALlabels.

The **matsize= matsize=** option can be used to change the arrangement of the alternatives. The default is to have the alternatives labelled and listed horizontally. For many problems, there

are 5 short alternative and this fits. However if the alternatives have long statements you may want to change the geometry to a vertical listing. Set matsize=[5,1] to do this.

```
> tagit(Line("What is " ,1/‘2‘ + 1/‘3‘,"?"),matsize=[5,1],
  _ABlabels([[1/2+1/3],[1/‘2‘+1/3],2/5,1/10,3/2],randomize=no));
```

```
QM_[0.05;A;B;;;]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_

What is

lt_/td gt_

lt_td gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_

lt_td gt_

?

lt_/td gt_

lt_/tr gt_lt_/table gt_

lt_table gt_lt_tr gt_lt_td valign="middle" gt_ lt_font color="red" gt_
lt_b gt_A. lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{5}{6}$$

```
 lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_B. lt_/b gt_
amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_C. lt_/b gt_
amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{2}{5}$$

```
lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_D. lt_/b gt_
amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{1}{10}$$

```
lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_E. lt_/b gt_
amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{3}{2}$$

```
lt_/td gt_

lt_/tr gt_lt_/table gt_

AB_[A;B;C;D;E]

SKIP_
```

The **Alabels= Alabels=** option is used to change the labels provided (Capital letters) to your choice. The Flabels= option changes the label color used with _ALlabels and _ABlabels from the default (red) to your choice.

```
> tagit(Line("What is " ,1/‘2‘ + 1/‘3‘,"?"),
  matsize=[5,1],Flabels="blue",Alabels=["i","ii","iii","iv","v"],
  _ABlabels([[1/2+1/3],[1/‘2‘+1/3],2/5,1/10,3/2]));
```

```
QM_[0.05;i;;iii;;]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_

What is

lt_/td gt_

lt_td gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_

lt_td gt_

?

lt_/td gt_

lt_/tr gt_lt_/table gt_

lt_table gt_lt_tr gt_lt_td valign="middle" gt_ lt_font color="blue"
gt_ lt_b gt_i. lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{5}{6}$$

```
 lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="blue" gt_lt_b gt_ii. lt_/b gt_
amp_nbsp; lt_/font gt_lt_/td gt_
```

```
lt_td valign="middle" gt_
```

$$\frac{3}{2}$$

```
lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="blue" gt_lt_b gt_iii. lt_/b
gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="blue" gt_lt_b gt_iv. lt_/b gt_
amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{2}{5}$$

```
lt_/td gt_

lt_/tr gt_lt_tr gt_

lt_td valign="middle" gt_lt_font color="blue" gt_lt_b gt_v. lt_/b gt_
amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{1}{10}$$

```
lt_/td gt_

lt_/tr gt_lt_/table gt_

AB_[i;ii;iii;iv;v]

SKIP_
```

6.2.24 The option pretext=

The **pretext= pretext=** option is used place text in a **T T** section before the question. **Note:** For extensive material, it is best to put this in a section by hand. Also, put the pretext stuff before the first answer format in a problem.

```
> tagit(pretext=["In the next problem, get a common denominator before
  adding."], Line("What is " ,1/‘2‘ + 1/‘3‘,"?",_AC(1/2+1/3)));

T_


In the next problem, get a common denominator before adding.

SKIP_


QM_[0.05;5/6]

AH_[0]

lt_table  gt_lt_tr gt_
```

```
lt_td gt_
What is
lt_/td gt_
lt_td gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_
lt_td gt_
?
lt_/td gt_
lt_td gt_
AC_[5]
lt_/tr gt_lt_/table gt_
SKIP_
```

6.2.25 The option randomize=

Use the **randomize= randomize=** option with the multiple choice formats AL, ALlabels, AB, ABlabels, AS, and AR to turn off the shuffling of the alternative answers. When you do this, you need to use rightanswers= to override the assumption that the first answer is the correct one for AL, ALlabels, AS, and AR.

```
> tagit(Line("Is the follow equation true?  "
  ,1/‘2‘+1/3=2/5,_AS([true,false],
  randomize=no,rightanswers=false)));
```

```
QM_[0.05;false]
AH_[0]
lt_table  gt_lt_tr gt_
lt_td gt_
Is the follow equation true?
lt_/td gt_
lt_td gt_
```

$$\frac{1}{2}+\frac{1}{3}=\frac{2}{5}$$

```
lt_/td gt_
lt_td gt_
AS_[true;false]
lt_/td gt_
lt_/tr gt_lt_/table gt_
SKIP_
```

6.2.26 The options txtboxsize= and precision=

The **txtboxsize= txtboxsize=** and **precision= precision=** options are used with the entry answer formats AC, AW, AF, AE, AI, AT, and ATcross to control the size of an entry box and the acceptable error (excepting AW), respectively. The default for precision is .05. The default txtboxsize for AC and AT is 8 and for AW, AF, AE, and AI is 15.

```
> tagit(Line( "What is " ,‘1.342‘ +
  21.43,"?",_AC(1.342+21.43,txtboxsize=6, precision=‘.001‘)));
```

```
QM_[.001;22.772]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_

What is

lt_/td gt_

lt_td gt_
```

$$1.342 + 21.43$$

```
lt_/td gt_

lt_td gt_

?

lt_/td gt_

lt_td gt_

AC_[6]

lt_/td gt_

lt_/tr gt_lt_/table gt_

SKIP_
```

6.2.27 putting drawings into problems.

To put a drawing into a problem composed with tagit, give the drawing a name and then insert the name into the tagit line.

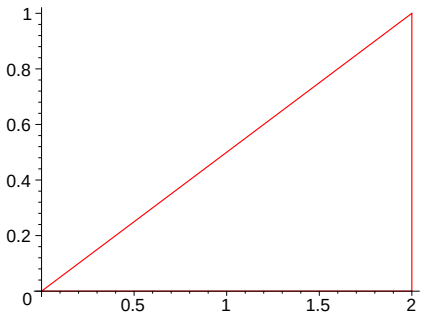
```
> triangle_drawing :=
  plots[display](plot([[0,0],[2,0],[2,1],[0,0]]),thickness=3);
```

triangle_drawing := PLOT(
CURVES([[0., 0.], [2., 0.], [2., 1.], [0., 0.]], COLOUR(*RGB*, 1.0, 0., 0.)),
THICKNESS(3), AXESLABELS(“”, “”), VIEW(*DEFAULT*, *DEFAULT*))

```
> tagit("What is the area of the displayed
  triangle?",triangle_drawing,"Area = ",_AC(1));
```

```
QM_[0.05;1]
```

```
AH_[0]

What is the area of the displayed triangle?
```




```
Area =

AC_[5]

SKIP_
```

6.2.28 More on avoiding hand-formatting of problems

Try to avoid hand-formatting of any mathematical expressions whenever possible. This is for maintainance reasons. If we need to make a change in the problem, we would have to re-format the expression. The MCtools word **Line Line** can be used to format the lines of a problem. Another method is to use ATcross. Here is an example of the same problem done with each format. We have overridden the default txtboxsize of 15 in ATcross.

```
> tagit(Line("What is ",1/`2`+1/3,"?",_AC(5/6)),
  txtboxsize=5,_ATcross([["What is ",1/`2`+1/3,"?",[5/6]]]));
```

```
QM_[0.05;5/6;5/6]

AH_[0]

lt_table  gt_lt_tr gt_

lt_td gt_

What is

lt_/td gt_

lt_td gt_
```

$$\frac{1}{2}+\frac{1}{3}$$

```
lt_/td gt_

lt_td gt_

?
```

lt_/td gt_

lt_td gt_

AC_[5]

lt_/td gt_

lt_/tr gt_lt_/table gt_

lt_table border="0" gt_

lt_tr gt_

lt_td bgcolor="white" gt_

What is

lt_/td gt_

lt_td bgcolor="white" gt_

$$\frac{1}{2}+\frac{1}{3}$$

lt_/td gt_

lt_td bgcolor="white" gt_

?

lt_/td gt_

lt_td bgcolor="white"gt_

AC_[5]

lt_/td gt_

lt_/tr gt_

lt_/table gt_

SKIP_

> ;

7 Drawing diagrams to accompany problems.

You can often think of a picture or diagram that would be useful in the statement of a problem. We can use Maple to draw these pictures, using words from plots and plottools, packages of words used to draw pictures, and also from MCtools, which is the same package that contains tagit, and should be loaded at the beginning of each session. To see the words in a package and at the same time make them usable, load the package using with

```
> with(plots); with(plottools); with(MCtools);
```

There are a lot of words here, but just like the regular vocabulary, the words in plots and plottools all have help pages too. The MCtools package does not have help pages, but the command mctools() does provide some basic online help. The main drawing words in MCtools are **DL DL** , **DV DV** , **GP GP** , **PA PA** , **PC PC** , and **PT PT** . These were designed to give easy access to some of the words in plots and plottools.

One of the words you will use a lot is **display** from the plots package. You draw different parts of the picture you are thinking of and give them names, then you display them together.

When you have a picture to draw, you can always load plots and plottools and scan for words which might help you make the drawing. You can bring up the help sheet on words that might help you and look at the examples there. There is also an extensive drawing tutorial you can consult later in this text.

7.1 Example: Floor plan

Problem: Draw a floor plan for a rectangular kitchen which is 10 by 20 and has a 2 by 4 sink along the long side.

Solution: We use **plots[polygonplot]** **plots[polygonplot]** to draw the floor. The floor is a rectangle, so we set up a coordinate system and assign coordinates to the vertices of the rectangle.

```
> fl := [[0,0],[20,0],[20,10],[0,10],[0,0]];
```

$$fl := [[0, 0], [20, 0], [20, 10], [0, 10], [0, 0]]$$

Now draw the floor with polygonplot and color it yellow. Give a name to the drawing.

```
> pl1:=plots[polygonplot](fl,color=yellow);
```

$$pl1 := \mathrm{PLOT}(\mathrm{POLYGONS}([[0., 0.], [20., 0.], [20., 10.], [0., 10.], [0., 0.]]), \mathrm{COLOUR}(RGB, 1.00000000, 1.00000000, 0.))$$

Now draw a sink and give it a name.

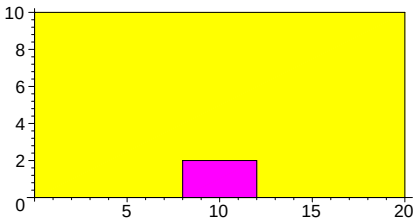
```
> sink := [[8,0],[12,0],[12,2],[8,2],[8,0]];
> pl2:= plots[polygonplot](sink,color=magenta);
```

$$sink := [[8, 0], [12, 0], [12, 2], [8, 2], [8, 0]]$$

$$pl2 := \mathrm{PLOT}(\mathrm{POLYGONS}([[8., 0.], [12., 0.], [12., 2.], [8., 2.], [8., 0.]]), \mathrm{COLOUR}(RGB, 1.00000000, 0., 1.00000000))$$

We display both of these pictures together using **display** from the plots package. We use the option **scaling = constrained** **scaling = constrained** because we want the same scale on both axes.

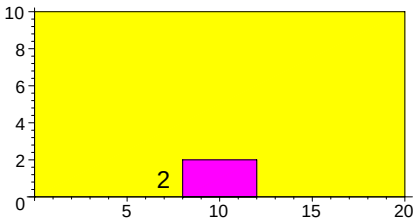
```
> plots[display](pl2,pl1,scaling=constrained);
```



Other solutions: We could have used **polygon** from the plottools package to draw the floor and sink.

We might want to label the diagram. This can be done with **plots[textplot]** **plots[textplot]** or more easily with **PT** (put text) from the MCtools package. The labels can be created created and named separately and added to the display or they can simply be put into the display as below.

```
> plots[display](PT([7,1],2),pl2,pl1,scaling=constrained);
```



After you get all the labels added you can remove the axes by adding the option axes = none.

```
> plots[display](PT([7,1],2),pl2,pl1,scaling=constrained,axes=none);
```

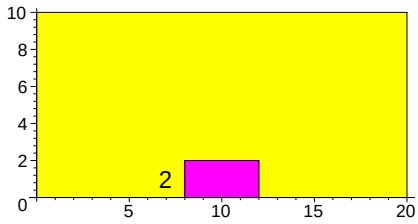
Note: The process of naming the parts to a drawing and displaying them together can be carried out several ways. A good way is to develop the drawing in a single input cell, using the shift return to add new lines to the input cell. Our drawing could have been developed like so:

```
> fl := [[0,0],[20,0],[20,10],[0,10],[0,0]]:
  pl1:=polygonplot(fl,color=yellow):
  sink := [[8,0],[12,0],[12,2],[8,2],[8,0]]:
> pl2:= plots[polygonplot](sink,color=magenta):
  plots[display](PT([7,1],2),pl2,pl1,scaling=constrained);
```

We can make up a problem which uses this diagram.

QM_[0;200-6;four]

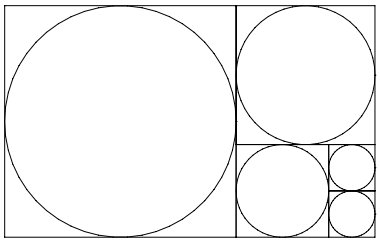
A kitchen floor which is 10 feet by 20 feet is to be tiled, except for a 2 foot by 3 foot cabinet along one of the long walls (see the diagram).



If we use tiles which are 12 inches square, how many tiles will be needed?
AC_[4]

AH_[0]
If we use tiles which are 6 inches square, we will need
AS_[two,three,four]
times as many tiles.
SKIP_

Problem. Make up and solve an area problem about a polygon such as a rectangle with some cutouts, or a trapezoid. Draw a diagram for the statement of the problem and add the problem with accompanying diagram to your WHS homework. **Problem.** Duplicate this drawing.



I will get you started: plots[display](

7.2 Another example: Draw an ice cream cone

A solution. There might be some words in the plottools package to help here.

```
> with(plottools);
```

Yes. cone and sphere are the words we are looking for. What is their syntax? we can type

```
> example(cone);
```

This brings up the helpsheet for cone and goes to the examples section. copy the example and paste it into your worksheet. Then bring up the helpsheet for sphere. (Actually that word is used in other packages in Maple, so you have to choose. Or you can say example(plottools[sphere]);)

Here is diagram we have developed from the helpsheets.

```
> icecream :=
  cone([0,0,-1],0.7,color=yellow),sphere([0,0,0.1],0.6,color=pink):
  plots[display](icecream, scaling=constrained,
  style=patch,axes=boxed);
```

This does not solve our problem. We want the sides of the cone to be tangent to the sphere.

It is easier to see in cross section. The angle drawn from the center of the sphere [0,0,z] to the point of tangency [.7,0,0] and then to the base of the cone [0,0,-1] should be a right triangle. So $(z+1)^2 = 1^2 + .7^2 + z^2 + .7^2$

. Solving for z, we get z = .7^2. So the radius of the sphere is $\sqrt{.7^2 + (.7^2)^2} = .854$ approximately

So we can modify the drawing to solve the problem.

```
> icecream := plottools[cone]([0,0,-1],.7,color=yellow),
  plottools[sphere]([0,0,.7^2],.854,color=pink):
  plots[display](icecream, scaling=constrained);
```

We can label this diagram using the MCtools word PT. Note we have also changed the rendering of the ice cream by changing the style and lightmodel.

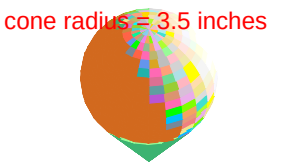
```
> plots[display](PT([0,0,1.5],‘cone radius = 3.5
  inches‘,color=red),icecream,scaling=constrained,style=patchnogrid,ligh
  tmodel=light4);
```

Note the use of the word plots[display]. This is used to display several plots in the same picture. By using it as plots[display] instead of in the form display, it is not necessary to have loaded the plots package into vocabulary in order to use it. (If you say display(icecream) and have not loaded the plots package by saying with(plots) first, the output will be the input 'display(icecream)'.)

Often a problem is easier to understand if it is accompanied by an appropriate diagram. Or perhaps the diagram may serve only to focus attention on the problem, and make it more interesting. In this case, we also are drawing the diagram to check our own solution to the problem.

Draw the diagram after a skip tag and before the qm tag, then copy and paste it into the problem

QM_[0;5*sqrt(.7^2 + (.7^2)^2)]



A cone has a a height of 5 inches, and a radius of 3.5 inches. What is the radius of the ice cream ball that just fits in the cone?

AC_[8]

SKIP_

Problem: Make a problem which has a three dimensional diagram in it. Solve the problem. Then post it to your WHS homework along with the diagram.

7.2.1 Using color in your problems

Here are the 24 named colors in Maple

```
> colors();
```

Custom color Custom color can be created with the plot option color=COLOR(RGB,r,g,b) where r, b, and g are numbers between 0 and 1. So COLOR(RGB,1,0,0) is red, COLOR(RGB,0,1,0) is green , and COLOR(RGB,0,0,1) is blue. Also COLOR(RGB,0,0,0) is black , and COLOR(RGB,1,1,1) is white. Other mixtures, produce other colors: for example, yellow is equal parts of green and blue. The word trycolor defined below allows you to experiment with different values for r, b, and g and compare the results with the named colors.

```
> trycolor:=proc(r,g,b)
  plots[display](colors(),plottools[rectangle]([-10,20],[160,40],color=
  COLOR(RGB,r,g,b)))
  end:
```

For example, Maple green is too lime. A much better green for display purposes is obtained by making it darker. So COLOR(RGB,0,1/2,0) is what some would call hunter green or forest green.

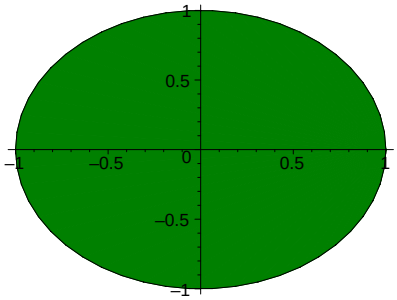
```
> trycolor(0,1/2,0);
```

After you find a color you like, you can give it a name (using the word macro) and use it like you would the 24 named colors. However, the name must be redefined each time you open up the worksheet, if you plan to use the color again.

```
> ?plot[options]
> macro(darkgreen=COLOR(RGB,0,1/2,0));
```

darkgreen

```
> plots[display](plottools[disk]([0,0],1,color=darkgreen));
```



8 Parameterized problems - problem generators

It is useful for many purposes to have several versions of the same problem with only the numbers changed. The numbers we change are called the **parameters parameters** of the problem. We can insert letters to stand in for the parameters, then use some algebra or some other mathematical methods to solve the problem in terms of the parameters. Usually, this gives us a formula for the solution as a function of the parameters. Then to solve a specific version of the problem, we just plug the specific values of the parameters into the formula.

We loosely refer to this process as **parameterizing the problem parameterizing the problem** . Sometimes, and this is something you can pass on to your students, it is as easy or even easier to solve a parameterized version of the problem as it is to solve the original problem. In any case, the technique can be used by teachers to make up lots of problems which are the same except the numbers are different, without going through the solution each time.

8.1 Example of a parameterization of a problem

Here is an old nugget, which has been used for generations to train student to set up equations to solve problems.

Original Problem. Bill is 1/2 his father's age. In 10 years, he will be 2/3's his father's age. How old is he now?

Solution. Let x be Bill's age, and y be his father's age. The first sentence says x = 1/2*y. The second sentence says x + 10 = 2/3*(y+10). So y = 2*x and x + 10 = 2/3*(2*x + 10) = 4/3*x + 20/3. Solving for x, 1/3*x = 10 - 20/3, or x = 30-20 = 10. Bill is 10 years old.

We can check the solution with Maple using **solve solve** :

```
> solve({x = 1/2*y,x + 10 = 2/3*(y+10)});
```

To parameterize this problem and make a **problem generator problem generator** , first we compose the original problem using the word **tagit tagit** from the MCtools package. We have already seen that this word relieves you of the burden of inserting tags into the problem; it also makes it easier to parameterize a problem, as we shall see. Here is the **tagitized tagitized** original problem.

```
> tagit("Bill is 1/2 his father's age.  In 10 years, he will be 2/3's
  his father's age.    How old is he now?",_AC(10)):
```

```
QM_[0.05;10]

AH_[0]

Bill is 1/2 his father's age. In 10 years, he will be 2/3's his
father's age.  How old is he now?

AC_[5]

SKIP_
```

To Parameterize the problem:

There are 4 numbers in the problem. We could parameterize all of them, but for starters, we could restrict ourselves to just parameterizing the 10. Just replace 10 by a suitable letter, say n, and resolve the problem in terms of n.

Parameterized Problem. Bill's 1/2 as old as his father. In n years, he will be 2/3's as old. How old is Bill now?

Solution. Let x be Bill's age, and y be his father's age. The first sentence says x = 1/2*y. The second sentence says x + n = 2/3*(y+n). So y = 2*x and x + n = 2/3*(2*x + n) = 4/3*x + 2/3*n. Solving for x, 1/3*x= 1/3*n, or x = n. Bill is n years old.

Now make a problem generator in the following fashion: Define a procedure, we have called it prob below, which has the parameter n as input. That is given in the first line of the procedure (the **'proc' line 'proc' line**) after the proc line comes the body of the procedure, which in this case is just a call to tagit. The procedure definition ends with an **'end' line 'end' line** .

In order to actually define the procedure, you must execute the input cell containing the definiton. This only needs to be done once in any given Maple session, unless you change the definition.

Then when we execute the procedure with specific values for the parameter, an **instance instance** or **version** of the problem is generated.

Compare the use of tagit below with its use above. In particular, note how the 10 was change to an ",n,". This is called **breaking out the parameter breaking out the parameter** . The parameter cannot be enclosed in double quotes so the string the 10 was in is broken into a sequence of two strings with the parameter in between. We enclosed the problem statement in a list so that the parameter value would be printed inline, rather than printed centered on a separate line.

```
> prob:= proc(n)  #the proc line
  tagit(["Bill is 1/2 as old as his father.  In  ",n,"  years, he will
  be  2/3's as old.    How old is Bill?"],_AC(n))
  end; # the end line
```

```
prob := proc(n)
    tagit(["Bill is 1/2 as old as his  father. In  ", n,
    "  years, he will be  2/3's as old.  How  old is Bill?"], _AC(n))
end proc
```

Now, to generate an instance of the problem, just make a call to the procedure defining the problem.

```
> prob(15);
```

```
QM_[0.05;15]

AH_[0]

Bill is 1/2 as old as his father. In  15  years, he will be  2/3's as
old.  How old is Bill?

AC_[5]

SKIP_
```

The nice thing about this is once you have the problem defined, it is easy to gererate several versions of the problem.

```
> prob(20):
```

```
QM_[0.05;20]

AH_[0]

Bill is 1/2 as old as his father. In  20  years, he will be  2/3's as
old.  How old is Bill?

AC_[5]

SKIP_
```

Problem: Replace the 1/2 in the problem by a parameter a and the 2/3 by a parameter b. Solve the resulting parameterized problem. Make a copy of the problem generator we have already, and modify it to problem generator with three paramers n, a, and b. Test your solution by posting two or three versions of the problem.

When you are putting in values for the parameters, a problem arises: What values of the parameters give a problem which 'makes sense'?

Even before you solve the problem in terms of the parameters, you can sometimes figure out this answer. For example, here we can see that a and b must be between 0 and 1, a must be less than b if n is postive, a must be = b if n is 0 and a must be greater than b if n is negative.

But even more interesting questions come up. For what values of the parameters does the problem not only make sense, but comes out nice.

Question: In the problem: 'Bill is a as old as his father. In n years, he will be b as old. How old is Bill?', for what positive integers n, and ratios a and b with $0< a < b < 1$ is Bill's age a positive integer < 120?

8.2 Another example: A carry problem

Here is an addition problem. We want the student to enter the correct carrys.

Carry _ _ _
7 8 9
+ 6 8 3
———-

- - - -

This is an example where we can put ATcross to good use. The output is suppressed because it is so busy. It looks better when posted.

```
> tagit(standards=["Can carry correctly."],"Complete the addition.    Put
  in the correct carrys.",
  _ATcross([["Carry:",[1],[1],[1],""],
> ["","",7,8,9],
  ["","+",6,8,3],
  ["",[1],[[4,2,3,1]],[7],[2]]
  ],txtboxsize=2));
```

Now we want to make a problem generator from this. First, we define and test a procedure to select an element at random from a given list or set.

```
> randelt := proc(set)
  local f;
  f := rand(1..nops(set));
  set[f()] end:
> seq(randelt({3,jack,10}),i=1..10);
```

jack, 3, *jack*, 3, *jack*, 3, *jack*, 3, *jack*, *jack*

That seems to work. Now to turn the tagit line into a problem generator, we copy it down and start parameterizing. If we replace all six digits in the two terms of the sum by parameters then we can express the digits of the answer and the digits of the carry's as shown in the diagram below.
Carry ch ct cu
a b c
+ d e f
———
ch h t u

So here is the problem generator. We make use of irem and iquo to get the digits of the answer and the carry's.

```
> carry := proc(a,b,c,d,e,f)
  local u,t,h,cu,ct,ch,randelt;
  randelt := proc(set)
> local f;
  f := rand(1..nops(set));
  set[f()] end:
> u := irem(c+f,10):  cu := iquo(c+f,10);
  t := irem(cu+b+e,10):  ct := iquo(cu+b+e,10);
  h := irem(ct+a+d,10):  ch := iquo(ct+a+d,10);
> tagit(standards="Can carry correctly.","Complete the addition.    Put in
  the correct carrys, including 0.",
  _ATcross([["Carry:",[ch],[ct],[cu],""],
> ["","",a,b,c],
  ["","+",d,e,f],
```

```
> ["",[[ch,ch+randelt([-1,1,2])]],[[h,h+randelt([-1,1,2])]],[t],[u]]
  ],txtboxsize=2)); end:
```

Now test the generator.

```
> carry(3,2,4,5,6,3);
```

8.2.1 Make the parameters random: Writing parameterless problem generators

After you have worked with a problem generator for awhile, you can use the rand function to supply values for the parameters in the generator.

We can use the randelt function to turn carry into a **parameterless problem generator** **parameterless problem generator** .

```
> carry := proc()
  local u,t,h,cu,ct,ch,randelt,a,b,c,d,e,f;
  randelt := proc(set)
> local f;
  f := rand(1..nops(set));
  set[f()] end:
> a := randelt([seq(i,i=1..9)]);
  b := randelt([seq(i,i=0..9)]);  #nonleading digits can be 0.
  c := randelt([seq(i,i=0..9)]);
> d := randelt([seq(i,i=1..9)]);
  e := randelt([seq(i,i=0..9)]);
  f := randelt([seq(i,i=0..9)]);
> u := irem(c+f,10):  cu := iquo(c+f,10);
  t := irem(cu+b+e,10):  ct := iquo(cu+b+e,10);
  h := irem(ct+a+d,10):  ch := iquo(ct+a+d,10);
> tagit(standards="Can carry correctly.","Complete the addition.    Put in
  the correct carrys, including 0.",
  _ATcross([["Carry:",[ch],[ct],[cu],""],
> ["","",a,b,c],
  ["","+",d,e,f],
  ["",[ch],[[h,h+randelt([-1,1,2])]],[t],[u]]
> ],txtboxsize=2));
  end:
```

Now we would test it by simply executing

```
> carry();
```

Writing a problem generator for adding m n-digit numbers.

This could be done using matrices and vectors to store the data. We would use the linalg package. It has a word **linalg[randmatrix]** **linalg[randmatrix]** to generate random matrices.

```
> linalg[randmatrix](3,2,entries=rand(1..9));
```

$$\begin{bmatrix} 7 & 5 \\ 5 & 1 \\ 5 & 6 \end{bmatrix}$$

Here is one solution. The word **map map** is used to check the problem, since the output

from ATcross is so extemely ugly and hard to read. We have also inserted the **standards=** **standards=** option in this problem.

```
> carryprob := proc(m,n)
  local A,sm,cr,hold,hnew,i,j;
  A := linalg[randmatrix](m,n,entries=rand(1..9));
> sm := vector(n+1);
  cr := vector(n+1);
  hold:= 0;
> cr[n+1]:=hold:
  for i from 1 to n do
  sm[n-i+2] := irem(hold+add(A[j,n-i+1],j=1..m) ,10,'hold');
> cr[n-i+1]:=hold;  od;
  if hold=0 then sm[1]:= `` else sm[1]:= hold fi:
  print(map(evalm,[cr,A,sm]));
> tagit(standards=["Can carry correctly."],"Complete the addition.    Put
  in the correct carrys.",
  _ATcross([["Carry:",seq([cr[i]],i=1..n),""],
> seq(["","",seq(A[j,i],i=1..n)],j=1..m-1),
  ["","+",seq(A[m,i],i=1..n)],
  ["",sm[1],seq([sm[i]],i=2..n+1)]
  ],txtboxsize=2));
> end:
```

To make a random problem for the student to add three 4-digit numbers, we would just execute

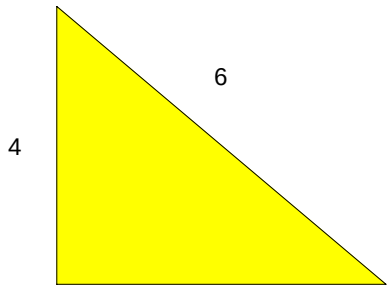
```
> carryprob(3,4);
```

Note: Maintainence problems arise when using randomly generated values for a parameter.

8.3 An example of a parameterized problem with a diagram

If you have a diagram that you want to add to a parameterized problem, just give a name to the plot and insert it into the problem=. For example:

Problem The right triangle shown (not to scale) has height 4 and hypotenuse 6. What is its area? What is its perimeter?



Solution: let x = base. then x^2 + 4^2 = 6^2, by Pythagoras. so x = sqrt(36-16) = sqrt(20).

so the area is 1/2*4*sqrt(20)= 4*sqrt(5) and the perimater is 4+6+x= 10+sqrt(20)

Note:

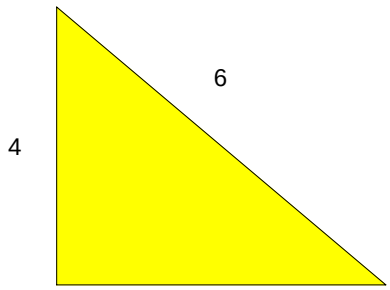
```
> pic
  :=plots[display](PT([-1,2],4),PT([4,3],6),plottools[polygon]([[0,0],[0
  ,4],[8,0]],color=yellow),axes=none);
```

$pic := \text{PLOT}(\text{TEXT}([-1., 2.], \text{“4”}, \text{COLOUR}(RGB, 0, 0, 0), \text{FONT}(HELVETICA, 14)),$
$\text{TEXT}([4., 3.], \text{“6”}, \text{COLOUR}(RGB, 0, 0, 0), \text{FONT}(HELVETICA, 14)),$
$\text{POLYGONS}([[0., 0.], [0., 4.], [8., 0.]],$
$\text{COLOUR}(RGB, 1.00000000, 1.00000000, 0.)), \text{AXESSTYLE}(NONE))$

To compose this with tagit,

```
> tagit(pic,"The right triangle shown (not to scale) has height 4 and
  hypotenuse 6.    \nIts area is ",
  _AC(4*sqrt(5)),", and its perimeter is ",_AC(4+6+sqrt(20)));
```

```
QM_[0.05;4*5^(1/2);10+2*5^(1/2)]

AH_[0]
```

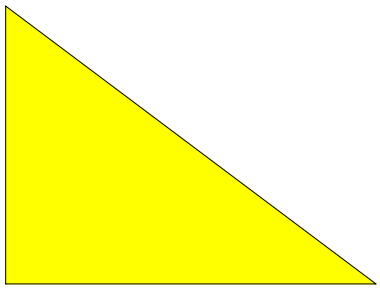



```
The right triangle shown (not to scale) has height 4 and hypotenuse 6.

Its area is

AC_[5]

AH_[0]

, and its perimeter is

AC_[5]

SKIP_
```

Note: the diagrams are always too large for the problem, and must be rescaled. The bad news is that this must be done by hand; the good news is that it only has to be done once. If you need to change something in the tagit line, then when you execute it again, the diagram assumes its new shap.

To parameterize the right triangle problem, we could parameterize both numbers

Problem The right triangle shown has height a and hypotenuse b. What is its area?



Solution: let x = base. then x^2 + a^2 = b^2, by Pythagoras. So x = sqrt(b^2-a^2). So the area is 1/2*a*sqrt(b^2-a^2)

Now to compose this for Mathclass, we can add a proc line and an end line, then break out the parameters, and replace the answers by the formulas which give the answers in terms of the parameters. Note: I have put the definition of pic right in the **problem generator problem generator** . That reduces the change of losing the picture and makes it easier to parameterize if you decide to do it. (See below)

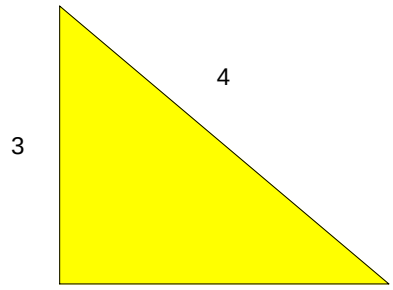
```
> prob:=proc(a,b)
  local pic;
  pic := plots[display](PT([-1,2],a),PT([4,3],b),
> plottools[polygon]([[0,0],[0,4],[8,0]],color=yellow),axes=none);
  tagit(pic,["The right triangle shown (not to scale) has height ",a,"
  and hypotenuse ",b,"."],"Its area is ",
> _AC(2*sqrt(b^2-a^2)),", and its perimeter is
  ",_AC(a+sqrt(b^2-a^2)+a/2*sqrt(b^2-a^2))) end;
```

$$
\begin{aligned}
&prob := \mathbf{proc}(a, b)\\
&\mathbf{local}\ pic;\\
&\quad pic := plots_{display}(MCtools : -PT([-1, 2], a),\ MCtools : -PT([4, 3], b),\\
&\qquad plottools_{polygon}([[0, 0], [0, 4], [8, 0]],\ color = yellow),\ axes = none);\\
&\quad MCtools : -tagit(pic, [\\
&\qquad \text{“The right triangle shown (not to scale) has height ”}, a, \text{“ and hypotenuse ”}, b, \text{“.”}],\\
&\qquad \text{“Its area is ”}, \_AC(2 * \mathrm{sqrt}(b^2 - a^2)),\ \text{“, and its perimeter is ”},\\
&\qquad \_AC(a + \mathrm{sqrt}(b^2 - a^2) + 1/2 * a * \mathrm{sqrt}(b^2 - a^2)))\\
&\mathbf{end\ proc}
\end{aligned}
$$

```
> prob(3,4);
```

```
QM_[0.05;2*7^(1/2);3+5/2*7^(1/2)]
```

```
AH_[0]
```




```
The right triangle shown (not to scale) has height 3 and hypotenuse 4.

Its area is

AC_[5]

AH_[0]

, and its perimeter is

AC_[5]

SKIP_
```

Problem: Parameterize this problem with 3 parameters: 'A right triangle with legs 3 and 4 is sitting on its hypotenuse, so that the hypotenuse is its base. What then is its height?' Solve the parameterized problem. Make a problem generator. Then insert a diagram into the problem generator.

Problem: Take one of your favorite problems and parameterize it with at least 1 parameter. Make it a problem generator from that.

8.4 Inessential parameters:

A parameter whose value does not affect the answer to the problem is called an **inessential parameter inessential parameter** . For example, if the problem is 'What is the slope of the line ax + b = y?', the answer to the problem, a, is independent of the value of b. I would call b an inessential parameter to this problem.

Here is another example of an inessential: In the problem, 'a can mow a yard in 3 hours and Jim can mow it in 5 hours. How long does it take them to mow it together?', the parameter a can take on the value 'Bill' or 'Sue' or any other name (including 'a') without changing the answer to the problem.

We can introduce inessential parameters into a problem to change the way the problem looks or is stated as a device for introducing variety in versioned homework.

For example, in the 'Bill and his father' problem above we could parameterize Bill's name. All you have to do is break out Bill from each string in which it occurs in the statement of the problem and put Bill in the inputs to the problem. It is not necessary to change Bill to a because 'Bill' can be used as a name for a parameter. Below is the old prob followed by the new prob. Note the changes.

```
> prob:= proc(n)  #the proc line
  tagit(["Bill is 1/2 as old as his father.  In  ",n,"  years, he will
  be  2/3's as old.    How old is Bill?"],_AC(n))
  end; # the end line
```

$$prob := \mathbf{proc}(n)$$
$$MCtools : -tagit([\text{“Bill is 1/2 as old as his father. In ”}, n,$$
$$\text{“ years, he will be 2/3's as old. How old is Bill?”}], \_AC(n))$$
$$\mathbf{end\ proc}$$

```
> prob:= proc(Bill,n)  #the proc line
  tagit([Bill," is 1/2 as old as his father.  In  ",n,"  years, he will
  be  2/3's as old.    How old is ",Bill,"?"],_AC(n))
  end; # the end line
```

$$prob := \mathbf{proc}(Bill, n)$$
$$MCtools : -tagit([Bill, \text{“ is 1/2 as old as his father. In ”}, n,$$
$$\text{“ years, he will be 2/3's as old. How old is ”}, Bill, \text{“?”}], \_AC(n))$$
$$\mathbf{end\ proc}$$

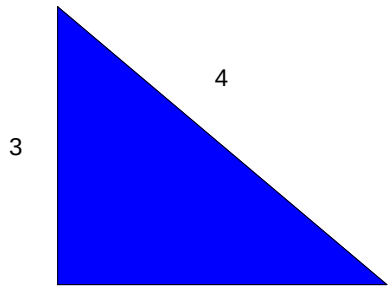
```
> prob(Sam,5);
```

```
QM_[0.05;5]

AH_[0]

Sam is 1/2 as old as his father. In  5  years, he will be  2/3's as
old.  How old is Sam?

AC_[5]

SKIP_
```

Another example: In the 'right triangle' problem above, the color of the (interior of the) triangle is yellow. We can parameterize this color by just inserting 'clr' as an input to prob. Then we must supply the color as input.

```
> prob:=proc(a,b)
  local pic;
  pic := plots[display](PT([-1,2],a),PT([4,3],b),
> plottools[polygon]([[0,0],[0,4],[8,0]],color=yellow),axes=none);
  tagit(pic,["The right triangle shown (not to scale) has height ",a,"
  and hypotenuse ",b,"."],"Its area is ",
> _AC(2*sqrt(b^2-a^2)),", and its perimeter is
  ",_AC(a+sqrt(b^2-a^2)+a/2*sqrt(b^2-a^2))) end:

> prob:=proc(a,b,clr)
  local pic;
  pic := plots[display](PT([-1,2],a),PT([4,3],b),
> plottools[polygon]([[0,0],[0,4],[8,0]],color=clr),axes=none);
  tagit(pic,["The right triangle shown (not to scale) has height ",a,"
  and hypotenuse ",b,"."],"Its area is ",
> _AC(2*sqrt(b^2-a^2)),", and its perimeter is
  ",_AC(a+sqrt(b^2-a^2)+a/2*sqrt(b^2-a^2))) end:


> prob(3,4,blue);
```

```
QM_[0.05;2*7^(1/2);3+5/2*7^(1/2)]

AH_[0]
```




```
The right triangle shown (not to scale) has height 3 and hypotenuse 4.

Its area is

AC_[5]

AH_[0]

, and its perimeter is

AC_[5]

SKIP_
```

Problem: Are there any inessential parameters you can introduce into the favorite problem you parameterized above? If so, introduce one of them into the inputs for your prob and re-post.

8.5 Parameterizing diagrams

Often the diagram that goes with a WHS **problem generator problem generator** is static, that is, it does not change when the inputs are changed. Or, as in the case of the right triangle problem, the labels on the legs were changed in the original problem generator, and then when we introduced the color of the triangle as an inessential parameter later, we can parameterize the diagram as it sits in the problem. Sometimes it is convenient to parameterize a diagram outside a problem generator. Then we can make a call to the diagram function in the problem generator.

Example: In the right triangle problem, if we wanted to draw the triangle to scale, then we have to worry about placing the labels correctly and also what the vertices of the triangle are going to be.

We could work on this leaving pic as a local variable in prob, but a more convenient way is to extract pic from prob and make a procedure out of it.

First how will we express the coordinates of the triangle in terms of a and b?

Keep one vertex at [0,0], and one of the others [0,a]. Now since b is the length of the hypotenuse, the vertex [x,0] on the x-axis satisfies $\sqrt{x^2 + a^2} = b$ Solving for x, we get $x = \sqrt{b^2 - a^2}$.

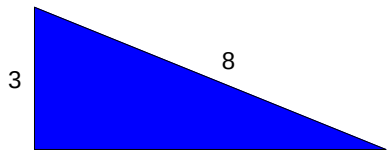
Next where will be place the labels in terms of a and b? A good place for the label of the vertical leg is at [-1,a/2], in other words, halfway up and to the left. Of course if a is very small,

then the label will be way off to the left. A better place would be [-eps,a/2], where we make eps an 'adjustment' parameter to the diagram generator.

Now, where to place the label for the hypotenuse. How about just above and to the right of the midpoint of the hypotenuse? The midpoint is [sqrt(b^2-a^)/2,a/2], so we use our adjustment parameter. [sqrt(b^2-a^2)/2+eps,a/2+eps].

The diagram generator could look like this. Note that we have added the option scaling = constrained, since we have gone to the trouble of making a scaled diagram.

```
> pic := proc(a,b,clr,eps)
  plots[display](PT([-eps,a/2],a),PT([sqrt(b^2-a^2)/2+eps,a/2+eps],b),
  plottools[polygon]([[0,0],[0,a],[sqrt(b^2-a^2),0]],color=clr),axes=non
> e,scaling=constrained);
  end:
> pic(3,8,blue,.4);
```



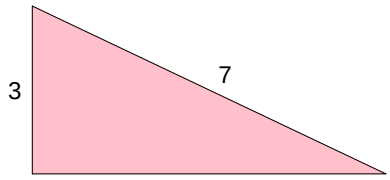
Now the problem generator has to be changed to incorporate the paramterized diagram. Note that we have modified the statement of the problem by taking out the disclaimer that the diagram is not to scale.

```
> prob:=proc(a,b,clr,eps)
  tagit(pic(a,b,clr,eps),["The right triangle shown  has height ",a,"
> and hypotenuse ",b,"."],"Its area is ",
  _AC(2*sqrt(b^2-a^2)),", and its perimeter is
  ",_AC(a+sqrt(b^2-a^2)+a/2*sqrt(b^2-a^2))) end:
```

Now test this.

```
> prob(3,7,pink,.3);
```

```
QM_[0.05;4*10^(1/2);3+5*10^(1/2)]

AH_[0]
```




```
The right triangle shown  has height 3 and hypotenuse 7.

Its area is

AC_[5]

AH_[0]

, and its perimeter is

AC_[5]

SKIP_
```

8.6 Optional Parameters: Inserting your own options into a diagram or problem generator

One of the things that happens after you learn to build generator is that the number of parameters in a procedure gets too long. It becomes difficult to remember the exact order of the parameters in the input line, and there are no default values for the parameters built in. This can be remedied by by using **option equations option equations** like the options to tagit, or plot. The parameters will have default values, and so can be left out if not needed. Also the order in which options are specified is not important.

8.6.1 Making a diagram generator with optional parameters.

We can illustrate with the diagram generator used above.

```
> pic := proc(a,b,clr,eps)
  plots[display](PT([-eps,a/2],a),PT([sqrt(b^2-a^2)/2+eps,a/2+eps],b),
  plottools[polygon]([[0,0],[0,a],[sqrt(b^2-a^2),0]],color=clr),axes=non
> e,scaling=constrained);
  end:
```

We show a way to reduce the number of parameters in the procedure to two (a and b).

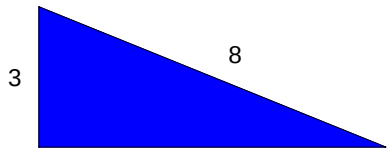
The parameters clr and eps are added to the local line. Also, two new local variables defaults and opts are added: defaults is set to a sequence of option equations, and opts is a list of the values of the optional parameters with the default values replaced by any value given by an option

equation at the call. Note the use of **subs subs** and **select select** to do this. Then the values of the parameters clr and eps are set from opts. The command select(type,[args],'=') take the arguments to pic, selects out the ones which are of type '=' (that is the equations in the input line). This returns a list (which may be empty), so op removes the enclosing square brackets and then result is joined with defaults to make a list of equations which are substituted into the list of left hand sides of the option equations. This is done from left to right, so any option equation which is input takes precedence over the default equation.

```
> pic := proc(a,b)
  local defaults,opts,clr,eps;
  defaults:= Color=yellow, Offset=.3;
> opts:= subs([ op(select(type,[args],'=')),defaults],[Color,Offset]);
  clr := opts[1]:
  eps := opts[2]:
> plots[display](PT([-eps,a/2],a),PT([sqrt(b^2-a^2)/2+eps,a/2+eps],b),
  plottools[polygon]([[0,0],[0,a],[sqrt(b^2-a^2),0]],color=clr),axes=non
  e,scaling=constrained);
  end:
```

Now, if we wanted a blue triangle with the labels offset by .5 instead of .3, we would execute

```
> pic(3,8,Color=blue, Offset=.5);
```



8.6.2 A problem generator with optional parameters

How would you modify the problem generator prob above to use this version of pic? This simplest way is to just have clr and esp be non-optional parameters to prob.

```
> prob:=proc(a,b,clr,eps)
  tagit(pic(a,b,Color=clr,Offset=eps),["The right triangle shown  has
> height ",a," and hypotenuse ",b,"."],"Its area is ",
  _AC(2*sqrt(b^2-a^2)),", and its perimeter is
  ",_AC(a+sqrt(b^2-a^2)+a/2*sqrt(b^2-a^2))) end:
```

However, if you wanted to, you could make clr and eps optional parameters in prob also, just by adding the same lines to prob that were added to pic.

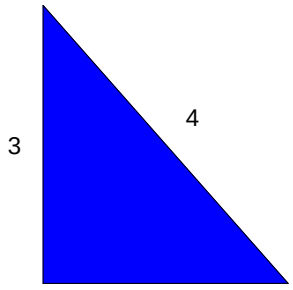
```
> prob:=proc(a,b)
  local defaults,opts,clr,eps;
  defaults:= Color=yellow, Offset=.3;
```

```
> opts:= subs([ op(select(type,[args],`=`)),defaults],[Color,Offset]);
  clr := opts[1]:
  eps := opts[2]:
> tagit(pic(a,b,Color=clr,Offset=eps),["The right triangle shown  has
  height ",a," and hypotenuse ",b,"."],"Its area is ",
> _AC(2*sqrt(b^2-a^2)),", and its perimeter is
  ",_AC(a+sqrt(b^2-a^2)+a/2*sqrt(b^2-a^2))) end:
```

Test this out

```
> prob(3,4,Color=blue);
```

```
QM_[0.05;2*7^(1/2);3+5/2*7^(1/2)]

AH_[0]
```




```
The right triangle shown  has height 3 and hypotenuse 4.

Its area is

AC_[5]

AH_[0]

, and its perimeter is

AC_[5]

SKIP_
```

8.6.3 Adding a Help= option to a generator

It may be useful in the future for both you and others who are using your generator to add a Help option. This is a simple text message explaining the syntax. It is similar to the mctools help, except it is part of the generator itself rather than a separate procedure. We illustrate by adding a help option to the diagram generator. As with each option, there is a new local variable, a new addition to the defaults line, another entry at the end of the list of options in the opts := line, and a new assignment of the local variable from that. In the help case, there is a conditional statement that is executed if help=yes. Note we use mcprint rather than print or lprint to display the help message. This is for cosmetic purposes.

```
> pic := proc(a,b)
  local defaults,opts,clr,eps,help;
  defaults:= Color=yellow, Offset=.3,Help=no;
```

```
> opts:= subs([
  op(select(type,[args],`=`)),defaults],[Color,Offset,Help]);
  clr := opts[1]:
> eps := opts[2]:
  help:=opts[3]:
> if help=yes then
  mcprint("pic(a,b) draws a right triangle (with blue interior) having
  labelled
> leg a and hypotenuse b.    The labels are offset by [-.3, 0] for a and
  [.3,.3]
  for b.
> Options:
  Color=yellow
  Offset=.3");  RETURN(NULL) fi;
> plots[display](PT([-eps,a/2],a),PT([sqrt(b^2-a^2)/2+eps,a/2+eps],b),
  plottools[polygon]([[0,0],[0,a],[sqrt(b^2-a^2),0]],color=clr),axes=non
> e,scaling=constrained);
  end:
```

8.7 Protecting yourself against bad calls to a diagram or problem generator

After you have constructed a generator, you may want to use it many times. Usually, you will discover that some ranges of values of the parameters should be avoided. You can add error traps to the generator to explain what ranges to avoid. For example, take the orginal diagram generator (without all the optional parameters).

```
> pic := proc(a,b,clr,eps)
  plots[display](PT([-eps,a/2],a),PT([sqrt(b^2-a^2)/2+eps,a/2+eps],b),
  plottools[polygon]([[0,0],[0,a],[sqrt(b^2-a^2),0]],color=clr),
> axes=none,scaling=constrained);
  end:
```

Where does this go bad? That is what values of a and b won't give a picture? When b <= a, since b is the hypotenuse of a right triangle with leg a. We could put in a gentle reminder of that in case the user (including yourself) might (inadvertently) put in values with a > b.

```
> pic := proc(a,b,clr,eps)
  if a >= b then ERROR("The first argument must be smaller than the
> second.") fi:
  plots[display](PT([-eps,a/2],a),PT([sqrt(b^2-a^2)/2+eps,a/2+eps],b),
> plottools[polygon]([[0,0],[0,a],[sqrt(b^2-a^2),0]],color=clr),
  axes=none,scaling=constrained);
  end:
```

So, if we try to execute

```
> pic(3,1,yellow,.3);
```

```
Error, (in pic) The first argument must be smaller than the second.
```

8.8 Inverse problems/domain of a parameter

Suppose you have solved a problem with parameters and have a method for obtaining the solution in terms of a, b, and c. Then we can often solve one or more of the **inverse problems inverse**

problems you get by exchanging the solution with one of the parameters. If your method is an equation with the solution on one side by itself and the other side an expression in the parameters, then it we can solve the inverse problem by solving the equation for the desired parameter.

Example

Problem: Bill has a oranges and Sally has b apples. How many pieces of fruit do they have together?

Solution: Let x be the number of fruits. Then x = a + b.

An inverse problem: Sally has a apples. Bill and Sally together have b pieces of fruit. How many pieces of fruit does Bill have?

Soution: Let x be the number of fruits Bill has. Then b = a + x. So x = b - a.

Problem State and solve an inverse problem to the Bill father age problem Bill's age is a times his father's age. In 10 years, his age will be b times his fathers age his father's age. How old is Bill? Put it in your WHS homework, using the answer format _AS with 5 alternatives. Make 2 of the alternatives 'possible wrong answers', that is answers a student might make either carelessness or by poor understanding of the rules of algebra.

Another concept that arises when you are solving a parameterized problem is that of the **domain domain** of a parameter. By this, we mean the set of values of the parameter for which there is a solution to the problem. For example, in the oranges and aples problem above, the domain of a (and b too) is all non-negative integers. Here the domains of the parameters are independent of each other. In the inverse problem described above, however, a can be any non-negative integer, but b must be an integer no less than a. So the domain of b depends on a.

Problem: In your solution to the problem 'Suppose we replace the 1/2 in the problem by a parameter a. and the 2/3 by a parameter b. Solve the resulting parameterized problem. Make a problem generator for it and post two or three versions of the problem.', are the parameters a and b independent? Describe the domains of these parameters.

8.9 Writing parameterless problem generators

After you have worked with a problem generator for awhile and know the domain of the parameters, you can use the rand function to supply values for the parameters in the generator. This makes the problem generator completely automated.

For example, a simple problem generator with random supplied values would be one which as one to add 1 digit numbers.

```
> addem := proc()
  local a,b,f;
  f := rand(1..9);
> a := f(); b:=f();
  tagit(["Find the sum:  ",a," + ",b," = "],_AC(a+b))
  end:
> addem();
```

```
QM_[0.05;11]

AH_[0]
```

```
Find the sum: 6 + 5 =
AC_[5]
SKIP_
```

A more complicated example would be to turn carry (from a previous section) into a **parameterless problem generator parameterless problem generator** .

```
> carry := proc()
  local u,t,h,cu,ct,ch,randelt,a,b,c,d,e,f;
  randelt := proc(set)
> local f;
  f := rand(1..nops(set));
  set[f()] end:
> a := randelt([seq(i,i=1..9)]);
  b := randelt([seq(i,i=0..9)]);  #nonleading digits can be 0.
  c := randelt([seq(i,i=0..9)]);
> d := randelt([seq(i,i=1..9)]);
  e := randelt([seq(i,i=0..9)]);
  f := randelt([seq(i,i=0..9)]);
> u := irem(c+f,10):  cu := iquo(c+f,10);
  t := irem(cu+b+e,10):  ct := iquo(cu+b+e,10);
  h := irem(ct+a+d,10):  ch := iquo(ct+a+d,10);
> tagit(standards="Can carry correctly.","Complete the addition.    Put in
  the correct carrys, including 0.",
  _ATcross([["Carry:",[ch],[ct],[cu],""],
> ["","",a,b,c],
  ["","+",d,e,f],
  ["",[ch],[[h,h+randelt([-1,1,2])]],[t],[u]]
> ],txtboxsize=2));
  end:
```

Now we would test it by simply executing

```
> carry();
```

Writing a problem generator for adding m n-digit numbers.

We can write problem generators where some of the parameters are author supplied, and others are randomly chosen. For example, we can generalize the carry problem to one where we specifiy how many numbers we want to add and how many digits they should have. This could be done using matrices and vectors to store the data. We would use the linalg package. It has a word **linalg[randmatrix]** **linalg[randmatrix]** to generate random matrices.

```
> linalg[randmatrix](3,2,entries=rand(1..9));
```

$$\begin{bmatrix} 7 & 5 \\ 5 & 1 \\ 5 & 6 \end{bmatrix}$$

Here is one solution. The word **map map** is used to check the problem, since the output from ATcross is so extemely ugly and hard to read. We have also inserted the **standards= standards=** option in this problem.

```
> carryprob := proc(m,n)
  local A,sm,cr,hold,hnew,i,j;
  A := linalg[randmatrix](m,n,entries=rand(1..9));
```

```
> sm := vector(n+1);
  cr := vector(n+1);
  hold:= 0;
> cr[n+1]:=hold:
  for i from 1 to n do
  sm[n-i+2] := irem(hold+add(A[j,n-i+1],j=1..m) ,10,'hold');
> cr[n-i+1]:=hold;  od;
  if hold=0 then sm[1]:= `` else sm[1]:= hold fi:
  print(map(evalm,[cr,A,sm]));
> tagit(standards=["Can carry correctly."],"Complete the addition.    Put
  in the correct carrys.",
  _ATcross([["Carry:",seq([cr[i]],i=1..n),""],
> seq(["","",seq(A[j,i],i=1..n)],j=1..m-1),
  ["","+",seq(A[m,i],i=1..n)],
  ["",sm[1],seq([sm[i]],i=2..n+1)]
  ],txtboxsize=2));
> end:
```

To make a random problem for the student to add three 4-digit numbers, we would just execute

```
> carryprob(3,4);
```

9 Adding hints and solutions to homeworks and problems

A **timely hint timely hint** can be a real help to a student in need, so it is in your interest and your students to provide hints and even solutions to some problems. This can be done several ways in WHS homework. One way is to insert a link to a video or audio file in the problem statement. Then, if the student has high speed access, or if the files are available locally (on a CD, hard, or floppy drive). Another way is to insert the hint in a sentence or two at the end of the problem, or in a text section before the start of the problem. You can also put the hint in a hidden section, so that the student has the choice of whether to read it, look at it, or listen to it.

9.1 Adding a video hint using the pocket video studio

It is very nice to be able to create short video clips of your hands writing while you are talking about the problem, perhaps giving the solution or giving a hint to the solution. Then you can put a link in the problem so that the student can access your hint if needed.

Here are the steps:

1 Decide what your hint will be. Maybe you could practice it a couple of times.

2. Use the studio to make the hint file

3. Copy the hint to a floppy or to a cd

4. Install the homework which links to the hint, using the answer header method. (see examples below).

Important! The header for a homework with video hints must contain the arguments [1;0,0;0;1] in order to access a file in the root of a local drive. Otherwise, WHS looks in a directory whose

name is the homework identifier for the file.

5. Upload the hint to the video server with the homework selected. This is the button just below the Upload homework button in the Homework Installation table

6. Check to make sure the hint is accessible from the homework, on your browser, which should be IE6 or higher. If it isn't then put mathclass.org and mathclass.com in your list of trusted sites:

a. Go to Tools->Internet Options->Security b. Choose either Local internet or Trusted sites and add https://www.mathclass.org and https://www.mathclass.com to either zone. c. You may need to close and re-open the browser. d. Actually, you may need to adjust the security policy of the zone. To do so, you click on Custom Level and then select a level – Medium or lower and click on Reset. Then click on ok twice to get out.

9.2 Tagging audio or video hints by hand.

If we tagged the problem by hand, we can just add the hint by hand. It can go in the answer header AH, as described in the author documentation. We show three positionings below. If you want other positionings, you may need to experiment.

A **video hint video hint** inserted in a line above the statement of the problem.
QM_[0;5]

AH_[0]

AH_[3;addition.wmv;click]

AH_[0]

What is 2 + 3?
AC_[4]

SKIP_

The same hint inserted in the line below the problem.
QM_[0;5]

AH_[0]

What is 2 + 3?
AC_[4]

AH_[3;addition.wmv;click]

SKIP_

Here is the same hint inserted inline.
QM_[0;5]

AH_[0]

What is 2 + 3?

AH_[0;addition.wmv;click]

AH_[0]

AC_[4]

SKIP_

9.3 Tagging hints in tagit, using av_ and ah_

av_ av_ is a MCtools word to add the answer header tag with a link to a video or audio file.

```
> mctools(av_);

MCtools, version Oct 15 2003.

Sorry that's not in the list.

MCtools, version Oct 15 2003.


Current choices are [ARRW, Axes, BOXLENGTH_, CARR, DL, DV, ERROR_, GP,
GP2, GP3, HARDCOPY_, Header, LATEX_, MC_constants, MM, PA, PC, PNUM_,
PP, PT, addlink, addsectionhint, addvlink, ah_, colors, fill, hashang,
htmltable, lineit, maketable, mcpiecewise, mcprint, mctools, printit,
roundto, symbolize, tableit, tagit, zipit, zipp].


"mctools(X)" gives syntax and defaults for "X"
```

I will insert the hint into the problem using the MCtools word av_. Note the first argument of AH_ is a 3. This ensures that WHS puts the hint link on a newline after the problem.

```
> tagit("What is 2 + 3?",_AS([5,4,6,7]),av_("cathy.wmv","click for
  hint"),_TP());

QM_[0.05;5]

AH_[0]

What is 2 + 3?

AS_[6;5;7;4]

AH_[3;cathy.wmv;click for hint]

SKIP_
```

ah_ ah_ is a MCtools word to insert answer headers into a tagit line. Tagit inserts by default answer headers with an argument of 0. This can be changed with the brks= option.

In order to put the hint inline at the beginning of the problem, call it with a first argument of 0 and also insert another answer header just before the statement of the problem, using ah_(0).

```
> tagit(av_(0,"hint.wmv","click for hint"),ah_(0),"What is 2 + 3?",
  _AC(5));

QM_[0.05;5]

AH_[0]

AH_[0;hint.wmv;click for hint]

AH_[0]

What is 2 + 3?

AC_[5]
```

SKIP_

Problem. Create a video hint for one of the problems in your homework and install it.

9.4 Adding a hidden section or hyperlink to a homework by hand.

Hidden Sections

You may have a lot of introductory material which you would like the students to be able to access if they need it. This can be put in in a **hidden section hidden section** at the top.

Steps for adding a hidden section

1. Open the section,
2. add the material,
3. then close the section
4. then export it to html.

Be sure the section is between the header tag and the skip tag, so it will show up in the posted homework. You can also insert hidden sections into problems by this method.

But, and this is important, in order for the hidden sections to be accessible, you need to put your **magic number magic number** into the zipit line with the **Magic= Magic=** option when you zip up the homework. The correct syntax is: **zipit("first","u://firstwhs",Magic=xxx);** where xxx is your magic number

Your magic number is your acctId number. If you are logged in to mathclass, you can view the source html of the mathclass page and look about 50 lines down for the string 'var acctId = xxx'. The xxx is your magic number.

Hyperlinks to external material

You can also put in a **hyperlink hyperlink** to some material in the top section. For example, if you want to point your students to the Dr. Math page, you can use the insert menu at the top of maple to put the link in.

Steps for adding a hyperlink, using the Dr. Math hyperlink as an example.

1. Put your cursor at the top of the homework (between the header tag and the skip tag) where you want the link to appear,
2. then click the **Insert** menu,
3. then choose **hyperlink**,
4. then in the box that appears type the link http://mathforum.org/dr.math/ and the text you want to appear in the hyperlink (something like Click here for Dr. Math will do).

9.5 Adding a hidden section or hyperlink in a problem via tagit.

To add a hint in a **hidden section hidden section** in a problem, we can do this **by hand** or using the MCtools word addsectionhint. In order for either of these to work correctly, it is necessary to know your **Magic number Magic number** (your author id number). See below.

Steps to add a hidden section by hand.

1. Create the section with the hint above the tagit line which generates the problem.
2. then close it and copy and paste the section into the formatted problem.

Be sure to put it above the skip tag and two lines below the answer line.

Important:

If your problem is parameterized and you have several instances of it, you will have to paste the section into each instance of the problem if you want the hint to occur in each version of the homework.

Also, if you update the problem by rexecuting the tagit line, you will have to remove by hand the skip that is generated.

Example: We have created the hidden hint section above the tagit line.

9.5.1 hint

Bill mows 1/3 yard per hour. Sam mows 1/5 yard per hour. So together they mow 1/3 + 1/5 = 8/15 yard per hour.

```
> simp := proc(a,b)
  tagit("Write the following expression as a simple fraction:",a+b/a,
  _ALlabels([(a^2+b)/a, (a+b)/a,b,"none of the others"])) end:
> simp(x,y);
```

```
QM_[0.05;B]

AH_[0]

Write the following expression as a simple fraction:
```

$$x + \frac{y}{x}$$

```
lt_table gt_tr_

lt_td valign=middle gt_

lt_font color=red gt_ lt_b gt_A. lt_/b gt_ amp_nbsp lt_/font gt_

dt_lt_td valign=middle gt_
```

$$\frac{x+y}{x}$$

```
 dt_

lt_td valign=middle gt_

 lt_font color=red gt_ lt_b gt_B. lt_/b gt_ amp_nbsp lt_/font gt_

dt_lt_td valign=middle gt_
```

$$\frac{x^2+y}{x}$$

```
dt_

lt_td valign=middle gt_

 lt_font color=red gt_ lt_b gt_C. lt_/b gt_ amp_nbsp lt_/font gt_

dt_lt_td valign=middle gt_
```

$$y$$

```
dt_

lt_td valign=middle gt_

 lt_font color=red gt_ lt_b gt_D. lt_/b gt_ amp_nbsp lt_/font gt_
```

```
dt_lt_td valign=middle gt_

none of the others

dt_

rt_lt_/table gt_

AL_[A;B;C;D]

SKIP_
```

9.5.2 Adding a hidden section using addsectionhint:

Addsectionhint is a MCtools word that takes 2 arguments:

Here is what mctools() says about it.

```
> mctools(addsectionhint);
```

```
MCtools, version Oct 15 2003.

addsectionhint(message,sectionmaterial), where title is the clickable
message,

and sectionmaterial is a list of strings, lists, and expressions. This
word is

written to be used in the problem=, pretext=, and aftertext=
environments,

but can be mcprinted outside those environments.
```

Important. This method requires that you use zipit with the option **Magic=xxx Magic=xxx** , just as you must do when you add a hidden section by hand. Also, the word **addsectionhint addsectionhint** needs to be enclosed in single quotes. This delays execution of the word for one step.

```
> simp := proc(a,b)
  tagit("Write the following expression as a simple fraction:",b+ a+b/a,
> _ALlabels([(b*a+a^2+b)/a, (a*b+b)/a,(2*b+a)/a,"none of the others"]),
  addsectionhint("click for hint",["Write each term as a fraction with
  denominator ",a,"."],"") ) end:


> simp(2,5);
```

```
QM_[0.05;A]

AH_[0]

Write the following expression as a simple fraction:
```

$$\frac{19}{2}$$

```
lt_table gt_lt_tr gt_lt_td valign="middle" gt_ lt_font color="red" gt_
lt_b gt_A. lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{19}{2}$$

```
 lt_/td gt_
```

```
lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; B.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$\frac{15}{2}$$

```
lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; C.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_
```

$$6$$

```
lt_/td gt_

lt_td valign="middle" gt_lt_font color="red" gt_lt_b gt_ amp_nbsp; D.
lt_/b gt_ amp_nbsp; lt_/font gt_lt_/td gt_

lt_td valign="middle" gt_

none of the others

lt_/td gt_

lt_/tr gt_lt_/table gt_

AL_[A;B;C;D]

\textbf{Hint:}

Write each term as a fraction with denominator 2.


SKIP_
```

9.5.3 Adding a hyperlink using addlink

Addlink Addlink is an MCtools word for adding a hyperlink to a problem.

```
> mctools(addlink);

MCtools, version Oct 15 2003.

addlink(url,title), where url is the full path to the hyperlink

(except for http://), and title is the text to be clicked. If the file
is in the homework

zipfile, then start the url with the string local:

This word is written to be used in the problem=, pretext=, and
aftertext= environments,

but can be mcprinted outside those environments.
> tagit(addlink("msc.uky.edu/carl/ma310","Click here for help"),"Bill
  can mow a yard in 3 hours.  Sam can mow it in 5.    How long will it take
  them if they mow together?",
> _AC(1/(1/3+1/5)));


QM_[0.05;15/8]

AH_[0]
```

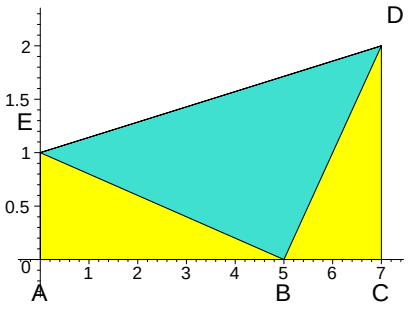
```
lt_a href="http://msc.uky.edu/carl/ma310" gt_ Click here for help
lt_/a gt_

Bill can mow a yard in 3 hours. Sam can mow it in 5.  How long will it
take them if they mow together?

AC_[5]

SKIP_
```

10 An extended example.

We want to include an example of how one can 'take off' on a problem, and end up with a series of interesting WHS homework problems. A version of this homework is posted in the WHS authors class on WHS.

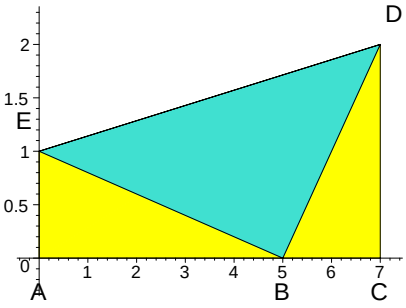
10.1 Trapezoid Problem

The trapezoid ACDE shown below has an inscribed equilateral triangle EBD. If AE = 8 and CD = 11, what is the area of the triangle?



10.1.1 drawing

```
> A:= [0,0]:  B := [5,0]:    C:=[7,0]:  De:= [7,2]:    E:= [0,1]:
> eps := .1:
> trap := plottools[polygon]([A,C,De,E],color=yellow):
> tri := plottools[polygon]([E,B,De],color=turquoise):
> labs := PT(A-3*[0,eps],"A"), PT(B-3*[0,eps],"B"),
  PT(C-3*[0,eps],"C"),
  PT(De+3*[eps,eps],"D"), PT(E+3*[-eps,eps],"E"):
> plots[display](labs,tri,trap);
```



A Solution which uses trig: Let x be the side of the triangle and alpha be the measure (in radians) of the angle ABE. Then

$\frac{8}{x} = \sin(\alpha)$

. Also, since angle EBD is a 60 degree $= 2/3^*\ \pi$ angle we have angle DBC has measure $\pi - \frac{\pi}{3} - \alpha$ $= \frac{2\pi}{3} - \alpha$. So another equation involving x and alpha is $\frac{11}{x} = \sin(\frac{2\pi}{3} - \alpha)$. Solving each equation for x and setting these equal we get $11\sin(\alpha) = 8\sin(\frac{2\pi}{3} - \alpha)$. Let's use Maple to calculate.

```
> x := 'x':  alpha:='alpha':
> eqn :=expand(11*sin(alpha) = 8*sin(2/3*Pi-alpha));
```

$$eqn := 11\sin(\alpha) = 4\sqrt{3}\cos(\alpha) + 4\sin(\alpha)$$

```
> eqn :=eqn - (4*sin(alpha)=4*sin(alpha));
```

$$eqn := 7\sin(\alpha) = 4\sqrt{3}\cos(\alpha)$$

```
> eqn:=subs(cos(alpha)=sqrt(1-(sin(alpha))^2),eqn);
```

$$eqn := 7\sin(\alpha) = 4\sqrt{3}\sqrt{1-\sin(\alpha)^2}$$

```
> eqn:= lhs(eqn)^2=rhs(eqn)^2;
```

$$eqn := 49\sin(\alpha)^2 = 48 - 48\sin(\alpha)^2$$

```
> eqn :=eqn + (48*(sin(alpha))^2=48*(sin(alpha))^2);
```

$$eqn := 97\sin(\alpha)^2 = 48$$

```
> eqn:= eqn/97;
```

$$eqn := \sin(\alpha)^2 = \frac{48}{97}$$

So, we get the sine of the angle

```
> sola:= sin(alpha)=sqrt(48/97);
```

$$sola := \sin(\alpha) = \frac{4\sqrt{291}}{97}$$

And from that the side of the triangle

```
> solx := x = 8/rhs(sola);
```

$$solx := x = \frac{2\sqrt{291}}{3}$$

So the area is $\frac{\sqrt{3}x^2}{4} = \frac{\sqrt{3}291}{9} = 56.00297612$

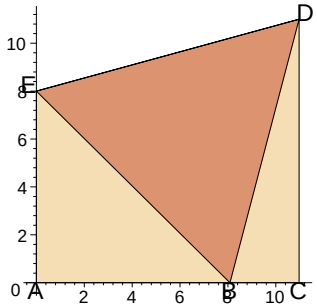
```
> sqrt(3.)/9*291.;
```

$$56.00297612$$

Note angle $\alpha = \arcsin\left(\frac{4\,291^{(\frac{1}{2})}}{97}\right)$

10.2 visual check.

We can check this by making a scale drawing of the trapezoid with it's inscribed equilateral triangle. So, $AB = x\cos(\alpha)$, $AE = x\sin(\alpha)$, $BC = x\cos(\frac{4\pi}{3} - \alpha)$, and $CD = x\sin(\frac{4\pi}{3} - \alpha)$.



10.2.1 drawing code

```
> x := 2/3.*291^(1/2):  alpha:= arcsin(4/97.*291^(1/2)):
> A:= [0,0]:  B := [x*cos(alpha),0]:
  C:=[x*cos(alpha)+x*cos(2/3*Pi-alpha),0]:
  De:= [x*cos(alpha)+x*cos(2/3*Pi-alpha),11]:    E:= [0,8]:
> eps := .1:
> trap := plottools[polygon]([A,C,De,E],color=wheat):
> tri := plottools[polygon]([E,B,De],color=tan):
> labs := PT(A-3*[0,eps],"A"),PT(B-3*[0,eps],"B"),PT(C-3*[0,eps],"C"),
  PT(De+3*[eps,eps],"D"),PT(E+3*[-eps,eps],"E"):
> plots[display](labs,tri,trap,scaling=constrained);
```

clearly BDE is an equilateral triangle

10.3 A parameterized version of the problem.

This problem has two obvious parameters: a=AE and b=CD, the heights of the trapezoid. We can go back through the solution and get the values of x and alpha in terms of a and b. Then we can investigate the behaviour of the solution as we vary the parameters.

The calculation and drawing sections below were obtained by modifying the calculation and drawing sections above.

10.3.1 the general calculations

First, assign each variable its own name, thus flushing any previously assigned values. (Note: the restart word does this also.

```
> x := 'x':  alpha:='alpha':  a  := 'a':  b := 'b':
```

Then replace occurences of 8 with a and occurences of 11 with b in the previous equations.

```
> eqn :=b*sin(alpha) = a*sin(2/3*Pi-alpha);
```

$$eqn := b\sin(\alpha) = a\sin(\frac{\pi}{3} + \alpha)$$

We are going to solve this equation for sin(alpha). First, expand the right hand side using the addition formula for sin

```
> eqn :=expand(eqn);
```

$$eqn := b\sin(\alpha) = \frac{1}{2}a\sqrt{3}\cos(\alpha) + \frac{1}{2}a\sin(\alpha)$$

Now, subtract a/2*sin(alpha) from both sides of the equation. This is done by subtracting an equation from eqn.

```
> eqn :=eqn - (a/2*sin(alpha)=a/2*sin(alpha));
```

$$eqn := b\sin(\alpha) - \frac{1}{2}a\sin(\alpha) = \frac{1}{2}a\sqrt{3}\cos(\alpha)$$

We can factor the equation. The maple word factor has been trained to work on equations.

```
> eqn :=factor(eqn);
```

$$eqn := -\frac{1}{2}\sin(\alpha)(-2b + a) = \frac{1}{2}a\sqrt{3}\cos(\alpha)$$

Now substitute cos(alpha)=sqrt(1-(sin(alpha))^2) by using the maple word subs.

```
> eqn:=subs(cos(alpha)=sqrt(1-(sin(alpha))^2),eqn);
```

$$eqn := -\frac{1}{2}\sin(\alpha)(-2b + a) = \frac{1}{2}a\sqrt{3}\sqrt{1-\sin(\alpha)^2}$$

Square both sides to remove the radical. Note: maple doesn't square an equation. But you can square the left hand side (lhs) and the right hand side (rhs) separately and make a new eqn.

```
> eqn:= lhs(eqn)^2=rhs(eqn)^2;
```

$$eqn := \frac{1}{4}\sin(\alpha)^2(-2b + a)^2 = \frac{3}{4}a^2(1-\sin(\alpha)^2)$$

Now, to solve for (sin(alpha))^2, expand the right hand side

```
> eqn:=lhs(eqn)= expand(rhs(eqn));
```

$$eqn := \frac{1}{4}\sin(\alpha)^2(-2b + a)^2 = \frac{3a^2}{4} - \frac{3}{4}a^2\sin(\alpha)^2$$

and add 3/4*a^2*(sin(alpha))^2 to both sides.

```
> eqn :=eqn + (3/4*a^2*(sin(alpha))^2=3/4*a^2*(sin(alpha))^2);
```

$$eqn := \frac{1}{4}\sin(\alpha)^2(-2b + a)^2 + \frac{3}{4}a^2\sin(\alpha)^2 = \frac{3a^2}{4}$$

factor

```
> eqn:=factor(eqn);
```

$$eqn := \sin(\alpha)^2(b^2 - ba + a^2) = \frac{3a^2}{4}$$

and divide the right hand side by the second factor of the left hand side. (Note: op(1,x*y) is x and op(2,x*y) is y).

```
> eqn:=op(1,lhs(eqn))=rhs(eqn)/op(2,lhs(eqn));
```

$$eqn := \sin(\alpha)^2 = \frac{3a^2}{4(b^2 - ba + a^2)}$$

So, we get the sine of the angle α (clearly we want the postive sin since alpha is between 0 and Pi/2)

```
> sola:= sin(alpha)=sqrt(rhs(eqn));
```

$$sola := \sin(\alpha) = \frac{\sqrt{3}\sqrt{\frac{a^2}{b^2 - b\,a + a^2}}}{2}$$

And from that the side of the triangle x

```
> solx := x = a/rhs(sola);
```

$$solx := x = \frac{2\,a\,\sqrt{3}}{3\sqrt{\frac{a^2}{b^2 - b\,a + a^2}}}$$

So the area is

```
> area = sqrt(3)/2*rhs(solx)^2;
```

$$area = \frac{2\,\sqrt{3}\,(b^2 - b\,a + a^2)}{3}$$

At this point, we have the angle alpha, the side x of the inscribed equateral triangle, and the area of the triangle expressed in terms of the parameters a and b, the heights of the trapezoid.

We can make a parameterized diagram to go with this.

10.3.2 parameterized drawing

The process of turning a drawing into a parameterized drawing can be done in more than one way. Here is one. 1. Copy the drawing section and connect the input cells with F4 if need be. 2. Decide on a name for our procedure, say pic.

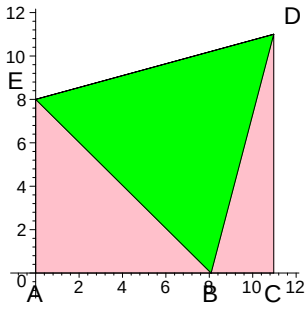
3. Put the 'proc line' at the top (in the same input cell, using shift enter to open lines without executing the cell). The parameters of the problem are the input parameters of the procedure. In this case, these are a and b. 4. Put calculation of the answer in the body of the procedure, before the actual drawing. In this case, we calculate x and alpha in terms of a and b. 5. Put the 'end line' at the bottom. That's it. Execute the definition of the procedure, and test it.

Notes: Any variables which are assigned a value in the body of a procedure is considered to be a local variable (one whose value is only known to the procedure) by Maple. A warning is issued if that variable is not in the 'local line'. In the pic procedure below, all the local variables have been declared in the local line.

```
> pic := proc(a,b,clrs)
  local x,alpha,A,B,C,De,E,eps,trap,tri,labs;
  x := 2/3*a*3^(1/2)/(a^2/(b^2-b*a+a^2))^(1/2):
> alpha:=arcsin(1/2*3^(1/2)*(a^2/(b^2-b*a+a^2))^(1/2));
  A:= [0,0]:  B := [x*cos(alpha),0]:
  C:=[x*cos(alpha)+x*cos(2/3*Pi-alpha),0]:
  De:= [x*cos(alpha)+x*cos(2/3*Pi-alpha),b]:    E:= [0,a]:
> eps := .3:
> trap := plottools[polygon]([A,C,De,E],color=clrs[1]):
> tri := plottools[polygon]([E,B,De],color=clrs[2]):
> labs := MCtools[PT](A-3*[0,eps],"A"),MCtools[PT](B-3*[0,eps],"B"),
  MCtools[PT](C-3*[0,eps],"C"),
  MCtools[PT](De+3*[eps,eps],"D"),MCtools[PT](E+3*[-eps,eps],"E"):
```

```
> plots[display](labs,tri,trap,scaling=constrained);
  end:
```

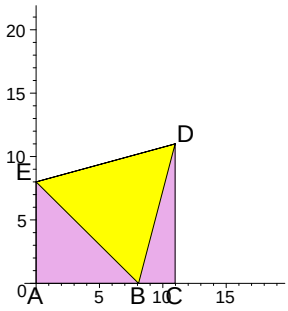
```
> pic(8,11,[pink,green]);
```



10.3.3 Animations

If you have a parameterized drawing, then it is easy to make an animation which shows what happens when one of the numerical parameters is changed. The idea is to make a sequence of drawings with the parameter changing and then display (using plots[display]) these together with the option insequence=true chosen. For example, to see what happens as the right hand height of the trapezoid increases from 11 to 21 we could play the animation below.

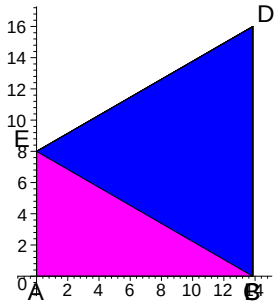
```
> plots[display](seq(pic(8,11+i,[plum,yellow]),i=0..10),insequence=true
  ,scaling=constrained);
```



An interesting observation occurs. After the height gets past 16, the inscribed triangle is no longer inscribed! This is something that we might not have noticed had we not done the animation. (Initially, it seemed that perhaps we had written incorrect code for the animation. However, we can see that the code is correct, because when the one height is more than twice the other, the side of the triangle is too large for the triangle to fit in the trapezoid. To verify this algebraically, go back and look at the algebra in the calculations and see that when CD = a is more than twice

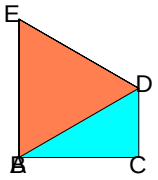
AE = b, then α is less than $\frac{\pi}{6}$, and so $\frac{2\pi}{3} - \alpha$ is greater than $\frac{\pi}{2}$ and so $\sin(\frac{2\pi}{3} - \alpha)$ is negative. So the segment AC $= x\sin(\alpha) + x\sin(\frac{2\pi}{3} - \alpha)$ is shorter than the segment AB $= x\sin(\alpha)$. So the equilateral triangle is not inscribed.

```
> pic(8,16,[magenta,blue]);
```



Now we can make an animation showing the inscribed equilateral triangle in each case (with a fixed AE).

```
> plots[display](seq(pic(20,10+i,[cyan,coral]),i=0..30),
  seq(pic(20,40-i,[cyan,coral]),i=0..30),
  insequence=true,axes=none,scaling=constrained);
```



10.3.4 A conjecture shot down

One might conjectured that the ratio of the area of the inscribed equilateral triangle to the area of the trapezoid remains constant. We can test that conjecture. From above,

```
> x := 2/3*a*3^(1/2)/(a^2/(b^2-b*a+a^2))^(1/2);assume(b>a,a>0);
  simplify(x);
```

$$x := \frac{2\,a\,\sqrt{3}}{3\,\sqrt{\dfrac{a^2}{b^2 - b\,a + a^2}}}$$

$$\frac{2\,\sqrt{3}\,\sqrt{b\tilde{}^2 - b\tilde{}\,a\tilde{} + a\tilde{}^2}}{3}$$

```
> alpha:=arcsin(1/2*3^(1/2)*(a^2/(b^2-b*a+a^2))^(1/2));
```

$$\alpha := \arcsin\left(\frac{\sqrt{3}\sqrt{\frac{a^2}{b^2-b\,a+a^2}}}{2}\right)$$

The the areas of the triangle and trapezoid are

```
> areatri := sqrt(3)/4*x^2;
```

$$areatri := \frac{\sqrt{3}\,(b^2-b\,a+a^2)}{3}$$

```
> areatrap:= (a+b)/2*x*(cos(alpha)+cos(2/3*Pi-alpha));
```

$$areatrap := \frac{1}{3}\,\frac{(a+b)\,a\,\sqrt{3}\left(\frac{\sqrt{4-\frac{3\,a^2}{b^2-b\,a+a^2}}}{2}-\sin\left(\frac{\pi}{6}-\arcsin\left(\frac{\sqrt{3}\sqrt{\frac{a^2}{b^2-b\,a+a^2}}}{2}\right)\right)\right)}{\sqrt{\frac{a^2}{b^2-b\,a+a^2}}}$$

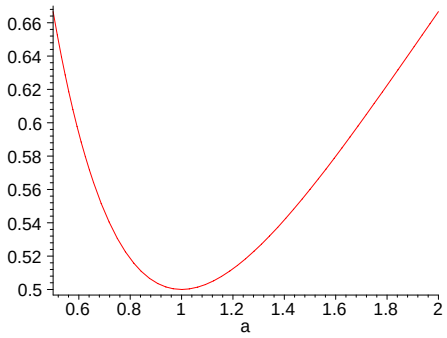
The ratio is

```
> rat := unapply(areatri/areatrap,a,b);
```

$$rat := (a,\,b) \rightarrow \frac{(b^2-b\,a+a^2)\sqrt{\frac{a^2}{b^2-b\,a+a^2}}}{(a+b)\,a\,(\frac{1}{2}\sqrt{4-\frac{3\,a^2}{b^2-b\,a+a^2}}-\sin(\frac{\pi}{6}-\arcsin(\frac{1}{2}\sqrt{3}\sqrt{\frac{a^2}{b^2-b\,a+a^2}})))}$$

Doesn't look constant, but looks can be deceiving. Let's plot the ratio over its range.

```
> plot(rat(a,1),a=.5..2);
```



Ok, it's not constant. What we can say that the maximum ratio 2/3 occurs at the extremes when the trapezoid is a 60 90 trapezoid and the minimum ratio 1/2 occurs at the midpoint when the trapezoid is a rectangle.

10.4 Creating WHS problems

After the analysis, one can come up with several interesting WHS problems related to the problem. Here are a few.

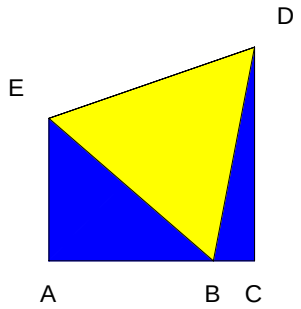
Here is the original problem tagitized.

```
> tagit("The trapezoid ACDE shown below has an inscribed equilateral
  triangle EBD. Angle A and angle C are right angles.    If AE = 8 and CD
  = 11, what is the side of the
> triangle?",plots[display](pic(8,11,[cyan,coral]),scaling=constrained,a
  xes=none),"Answer =",_AC(2/3*291^(1/2)));
```

We could turn this into a problem generator easily using the above calculations.

```
> trapprob:= proc(a,b,clrs)
  tagit(["The trapezoid ACDE shown below has an inscribed equilateral
  triangle EBD. Angle A and angle C are right angles.    If AE = ",a," and
> CD = ",b,", what is the side of the triangle?"],
  plots[display](pic(a,b,clrs),scaling=constrained,axes=none),
  "Answer =",_AC(3^(1/2)*(b^2-b*a+a^2)^(1/2))) end:


> trapprob(4,6,[blue,yellow]);
```

```
QM_[0.05;3^(1/2)*28^(1/2)]

AH_[0]

The trapezoid ACDE shown below has an inscribed equilateral triangle
EBD. Angle A and angle C are right angles.  If AE = 4 and CD = 6, what
is the side of the triangle?
```




```
Answer =

AC_[5]

SKIP_
```

Here is an interesting special case (as we have already noted).

```
> rectprob := proc(a,clrs)
  tagit(["A rectangle with height ",a," has an inscribed equilateral
  triangle BDE, as shown below.    What is the ratio of the longest side
> to the shortest
  side?"],plots[display](pic(a,a,clrs),scaling=constrained,axes=none),"A
  nswer =",_ALlabels([2/3*sqrt(3),1.1,2/3*sqrt(5)])) end:
```

```
> rectprob(8,[magenta,cyan]);
```

We ought to ask a question without a diagram.

```
> tagit("The trapezoid ACDE  has an inscribed equilateral triangle EDB.
  Angles A and C are right angles.  When is the ratio of the area of the
  triangle to the area of the trapezoid 2/3?","Answer:    When the
> trapezoid has an interior angle of ",_AS(["60 degrees","45 degree","30
  degree","70 degrees"]));
```

```
QM_[0.05;60 degrees]

AH_[0]

The trapezoid ACDE  has an inscribed equilateral triangle EDB. Angles
A and C are right angles. When is the ratio of the area of the
triangle to the area of the trapezoid 2/3?

Answer:  When the trapezoid has an interior angle of

AS_[60 degrees;70 degrees;30 degree;45 degree]

SKIP_
```

11 Another extended homework

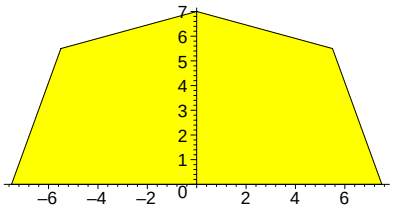
Note: a version of this homework is posted in the WHS Authors class on WHS.

Most of the homeworks you construct will be of the drill type. After all, one of the main reasons for putting homework in WHS is to provide students with lots of opportunity to practice what they have learned in class and receive automatic, instant feedback. This reduces the effort you need to expend at grading repetitive exercises, and frees you up do concentrate on developing classroom activities. The homework here is one which could be done in class, using a projecter to put the homework on a screen for the whole class to see. It was motivated by the shed design problem. Below is a picture of the finished shed.

11.1 Shed problem

Original problem: Here's a little picture for you to analyse. This will be the roof of a building I am constructing. It will have the height as 7 feet, with a base of 15 feet, and the angles starting at the bottom left going clockwise as 70, 125 and 150 degrees, with the other side being the same. Can you tell me the length of each side, and each half of the roof? Here is an approximate diagram.

```
> pentagon:=plottools[polygon]([[-7.5,0],[-5.5,5.5],[0,7],
  [5.5,5.5],[7.5,0]],color=yellow):
  plots[display](pentagon,scaling=constrained);
```



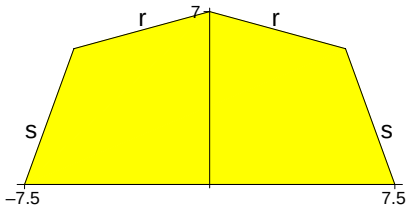
11.2 Building a homework set

We can build a WHS homework set around this problem. The statement of the problem and the diagram would be in the header section of the homework.

This will not be a versioned homework, although we could turn it into one making the height (currently 7) and width (currently 15) of the roof profile parameters.

A solution:

```
> lab1 := PT([7.2,2.25],"s"):    lab2 := PT([2.7,6.8],"r"):
  lab3 := PT([-7.2,2.25],"s"):    lab4 := PT([-2.7,6.8],"r"):
  pic:=lots[display](lab1,lab2,lab3,lab4,pentagon,scaling=constrained,
> xtickmarks=[-7.5,7.5],ytickmarks=[7]):  pic;
```



There is more than one way to solve this problem. We are going to to lead the student thru one solution, so we start by getting them to start thinking about why certain things are true. Like why do we only need 2 labels? A selection box answer format will do fine here.

```
> tagit("Let s and r be the lengths of the sides and roof.    We can show
  the labels",pic,"First, we know there are only
  ",_AS(["two","three","four"]),
  "unknowns, by the ",_AS(["symmetry","asymmetry"]),"of the shed.");
```

Here we are asking them to aknowledge that usually one looks for as many equations as we have unknowns.

```
> tagit("Since there are ",_AS(["two","three","four"]),
  "unknowns, we need",_AS(["two","three","four"])," equations in those
  unknowns.");
```

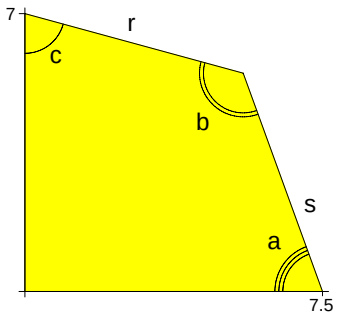
Now we are going to label some angles in the drawing. And the word **hashang** comes in handy here.

```
> mctools(hashang);
```

```
MCtools, version June 1 2004.

hashang(A,B,C,radius=1,numhashes=3,hashspacing=.1,withhead=`NO`,headth
ickness=1,headlength=1,clr=black,reversehead=`NO`,otherway=`NO`,fliphe
ad=`NO`)
```

Let's look at the right hand half of the shed and label the angles there a, b, and c. Here is the diagram code.

```
> ang1 := hashang([5.5,5.5],[7.5,0],[0,0]):  lab5:=PT([6.3,1.3],"a"):
  ang2 := hashang([0,7],[5.5,5.5],[7.5,0],numhashes=2):
  lab6:=PT([4.5,4.3],"b"):
> ang3 := hashang([0,0],[0,7],[5.5,5.5],numhashes=1):
  lab7:=PT([.8,6],"c"):
  pentagon:=plottools[polygon]([[0,0],[0,7],[5.5,5.5],[7.5,0]],color=yel
> low):
  pic:=plots[display](lab7,ang3,lab6,ang2,lab5,ang1,lab1,lab2,pentagon,
  scaling=constrained,xtickmarks=[-7.5,7.5],ytickmarks=[7]):  pic;
```



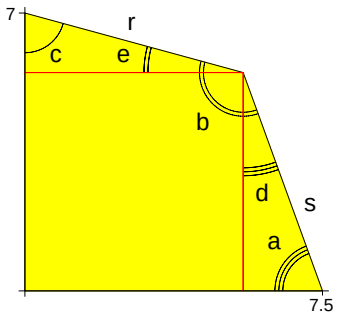
We can use this picture to advance in stating the next set of leading questions.

```
> tagit("The diagram shows the righthand half of the shed with three
  angles a, b and c.",pic,"From the statement of the problem, a =",
  _AS([80,[70],75])," degrees, b = ",
> _AS([150,[125],75])," degrees and c = ",
  _AS([150,[125],75])," degrees.",
  "\nTrue or false:    We only needed to be given two of the three
```

```
> angles.    The third one is determined by the other two.",
  _AS(["True","False"],randomize=no));
```

Now, we continue to add to the drawing. There are two additional angles which we can calculate from the problem, labelled d and e in the diagram

```
> ang4 := hashang([5.5,0],[5.5,5.5],[7.5,0],radius=2.4):
  lab8:=PT([6 ,2.5],"d"):
  ang5 := hashang([0,7],[5.5,5.5],[0,5.5],radius=2.4,numhashes=2):
> lab9:=PT([2.5,6],"e"):
  ln1 := DL([0,5.5],[5.5,5.5],color=red):
  ln2 := DL([5.5,0],[5.5,5.5],color=red):
> pic:=plots[display](ln2,ln1,ang5,lab9,ang4,lab8,lab7,ang3,lab6,ang2,la
  b5,
  ang1,lab1,lab2,pentagon,scaling=constrained,
  xtickmarks=[-7.5,7.5],ytickmarks=[7]):  pic;
```



This is so that we can add to the previous question.

```
> tagit(pic,"From the statement of the problem, d =",
  _AS([15,20,25])," degrees,and e =",
  _AS([15,20,25]));
```

In this question, we set up the system of two linear equations in r and s which will give the solution to the problem. Rather, we ask the student to pick out the correct second equation from a list of alternatives. The answer format ALlabels is a good one to use since we want the equatiion to be formatted.

```
> tagit("Now we can set up two equations in the unknowns r and s.  One
  is",
  r*cos(‘15‘*Pi/180) + s *cos(‘70‘*Pi/180) = 7.5,"This comes from the
> bases of the two small triangles in the above diagram.    The second
  equation is",
  _ALlabels([r*sin(‘15‘*Pi/180) + s *sin(‘70‘*Pi/180) = 7.5,
> [r*sin(‘15‘*Pi/180) + s *sin(‘70‘*Pi/180) = 7],
  r*tan(‘15‘*Pi/180) + s *tan(‘70‘*Pi/180) = 7],
  matsize=[4,1]));
```

The next question asks the student to solve the system. We use Maple to work out the correct solution.

```
> eqns:={r*cos(15*Pi/180) + s *cos(70*Pi/180) = 7.5,r*sin(15*Pi/180) +
  s *sin(70*Pi/180) = 7};
```

$$eqns := \{r\cos(\frac{\pi}{12}) + s\cos(\frac{7\pi}{18}) = 7.5,\ r\sin(\frac{\pi}{12}) + s\sin(\frac{7\pi}{18}) = 7\}$$

```
> solve(eqns,{r,s});
```

$$\{s = 5.884546073,\ r = 5.680940047\}$$

```
> sol:=evalf(fsolve(eqns),4);
```

$$sol := \{s = 5.884,\ r = 5.679\}$$

We choose to ask the student to answer the question to the nearest one thousandth of a foot. This requires us to reset the precision= option, since the default precision is .05.

```
> tagit("Now we can solve these two linear equations for r and s.  The
  answers to within a thousandth of a foot are ",
  _ATcross([["s =",[5.884]," and r =",[5.679]]],precision=`.0005`));
```

11.2.1 Follow up questions.

Now that we have answered the original question, we can search around for other questions about the same setting. We could ask the student to think about what happens to the solution as we vary the height and width of the shed. For example, suppose we kept the angles and the width fixed. How high can the shed be? We can use Maple to work out the details.

First set up the equations with the base and heigth set to b and h.

```
> eqns:=[r*cos(15*Pi/180) + s *cos(70*Pi/180) = b/2,r*sin(15*Pi/180) +
  s *sin(70*Pi/180) = h];
```

$$eqns := [r\cos(\frac{\pi}{12}) + s\cos(\frac{7\,\pi}{18}) = \frac{b}{2},\ r\sin(\frac{\pi}{12}) + s\sin(\frac{7\,\pi}{18}) = h]$$

Now solve the system for r and s in terms of b and h.

```
> sol:=solve({op(eqns)},{r,s});
```

$$sol := \left\{ s = -\frac{1}{2}\,\frac{-2\,h\cos(\frac{\pi}{12}) + b\sin(\frac{\pi}{12})}{\sin(\frac{7\,\pi}{18})\cos(\frac{\pi}{12}) - \cos(\frac{7\,\pi}{18})\sin(\frac{\pi}{12})},\right.$$
$$\left. r = \frac{1}{2}\,\frac{\sin(\frac{7\,\pi}{18})\,b - 2\,h\cos(\frac{7\,\pi}{18})}{\sin(\frac{7\,\pi}{18})\cos(\frac{\pi}{12}) - \cos(\frac{7\,\pi}{18})\sin(\frac{\pi}{12})} \right\}$$

Now, the maximum height of the shed would occur when r = 0, where the shed becomes a triangle. So, we can solve the second equation for h, then substitute in r=0 and b = 15

```
> maxh:=solve(rhs(sol[2]),h);
```

$$maxh := \frac{1}{2}\,\frac{\sin(\frac{7\,\pi}{18})\,b}{\cos(\frac{7\,\pi}{18})}$$

```
> evalf(subs(b=15,maxh));
```

$$20.60608068$$

We could ask a question which would ask them to realize this without actually making the computation.

```
> tagit("If we hold the angles fixed and hold the base of the shed
  fixed at 15 feet, then as we increase it's  height from 7 on up, there
  is ",
  _AS(["no limit",["a limit"]]),"to how high we can make the shed.");
```

We could also ask them to think whether there is a minumum height for the shed.

```
> tagit("Keeping the angles and the base fixed as before, then as we
  decrease the height, there is a positive number which the height must
  exceed.",
  _AS(["True","False"],randomize=no));
```

We could ask the student to dream up another question about this setting.

```
> tagit("In a sentence, ask another question about this setting.    I
  will be interested in what kind of question you
  ask.",_AX("",txtboxsize=6));
```

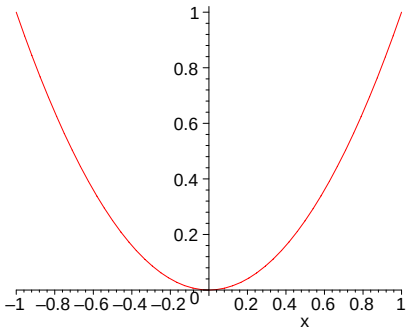
12 Creating Labelled Diagrams for Mathematics Problems: A tutorial

12.1 Starting from scratch

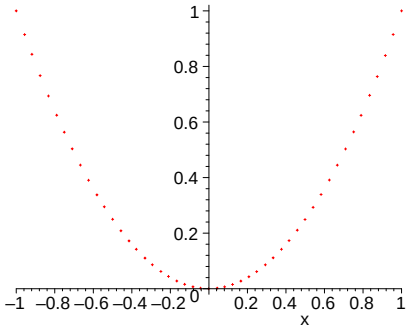
In this chapter, we go through a process of developing a diagram to be used in a WHS problem. Some of this has been covered before in a less complete or different fashion.

In addition to "everyday" spoken and written language mathematics is communicated through a vast geometric and symbolic vocabulary. Visual images tremendously facilitate the expression of mathematical concepts, problems, and solutions or proofs and are extremely useful in constructing sharply focused exercises for students. Our purpose in this section is to introduce some basic tools and techniques for constructing diagrams to illustrate mathematical exercises. As always we begin with the simplest cases first and will find, as usual, that that these will serve for the larger part of our needs.

The Maple command **plot** (x^2, x=-1..1); results in the following display.



Maple constructed this figure by actually calculating 50 "exact" points on the graph and connecting consecutive points by straight lines. We can see the points by modifying the command to plot(x^2, x=-1..1, style=point);



Notice that Maple is clever in the sense that the points are not equally spaced. There are more of them when the curve has a higher degree of curvature.

The picture we see on the screen is not the way Maple "thinks" of the plot - it can't interpret geometric objects. Maple stores the plot internally as a plot structure which we can view by giving it a name. For instance if we do the following

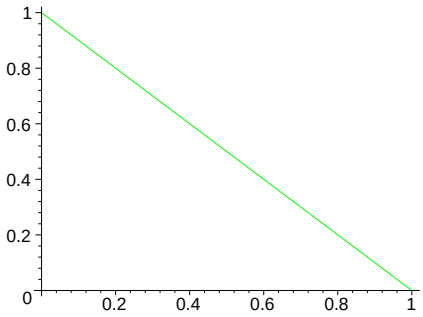
```
> joe:=plot([[0,1],[1,0]],color=green);
```

$$joe := \text{PLOT}(\text{CURVES}([[0., 1.], [1., 0.]]), \text{AXESLABELS}(\text{“”}, \text{“”}),$$
$$\text{COLOUR}(RGB, 0., 1.00000000, 0.), \text{VIEW}(DEFAULT, DEFAULT))$$

Maple has assigned the name "joe" to the plot structure (not the plot) generated by the plot command. This gives us a glimpse of how the software stores the information with which it generates the graph. For the moment it suffices to note that most of "joe" consists primarily of a listing of coordinates for points in the plane. If we were to go to the trouble to plot those points we would get exactly the "point plot" above.

Whenever we give Maple a word or sequence of words which it can interpret, it returns a representation of the result of that interpretation in a "human readable" format - unless instructed to do otherwise. At this point Maple would recognize "joe" as a particular plot structure so if we entered the command

```
> joe;
```



Maple will provide the "people readable" interpretation we expect

Viewing Multiple Plots

Of course Maple can "know" other plot structures. For instance we may enter

```
> sam:=plot(x^3,x=-1..1):
```

We know that "sam" will look pretty much like "joe" so we use the colon ":" to suppress the display. Now the **plots[display]** command is used to merge two or more plot structures into a single one. Note the component "[joe, sam]" in the following makes the expression sequence "joe,sam" into a **list.** The commands below would actually "work" without the list conversion. However the conversion is good practice as it will allow us to specify the order in which the components are assembled. Here sam goes down first, then joe. This will be important later when portions of one component cover portions of others.

```
> frank:=plots[display]([joe,sam]):
> frank;
```

Plots[display] is what permits us to **systematically** construct diagrams a piece at a time, adding components as we need them. All we need is a few tools for constructing the bits and pieces and, at least to start, a few procedures for organizing our work.

Using MCtools

MCtools is a table of Maple procedures which we have found useful in constructing diagrams to accompany exercises. Most of them are simply specializations of more general Maple commands.

When MCtools is needed one usually loads a worksheet with the current version into Maple, copies the MCtools section, and then pastes the section into the worksheet where one is working. The tools should remain a part of the worksheet (customarily at the top). Versions change frequently and there is no guarantee of forward or backward comapatabilty. However if the sheet contains the copy of the tools needed to build it then (relative to that problem set) there is probably no immediate concern over whether it is the most recent.

Then at the edit menu execute the entire worksheet. Then collapse the section.

If you add MCtools to a worksheet in progress you probably don't want to execute the entire worksheet. In this case simply expand MCtools, then expand the first section and place the cursor on the first execution line. Then hit return and continue to hit return until all of the MCtools execution cells have been executed.

12.2 Making Minimal Diagrams: Lines and Text

A remarkable number of diagrams require nothing more than the ability to draw line segments and label points of intersection, segments, etc. in figures comprised of line segments. Although Maple has general graphical tools for these and a few other basic constructions it is convenient to use specialized forms contained in the MathClassTools package.

Remember that the MCtools section must be executed in the current session (though not necessarily in the current worksheet) in order for them to work.

Once MCtools is executed Maple will recognize the various commands such as "DL". However if it simply returns the command as in the following then the MCtools have not been installed.

```
> DL([0,0],[1,1]);
```

12.2.1 Getting Started

We start with just two of the MCtools: **DL** and **PT**. The syntax for each of these is given by the command **mctools**

```
> mctools(DL);
```

The arguments not expressed in equations are **parameters** whose values must appear in the argument in the order given. Those whose value is indicated by an equation are options which may optionally be given in any order (after the variables are given). Options have default values which will be used unless other values are declared.

For instance **DL** (for "Draw Line") has the endpoints "A" and "B" of the line as variables and a number of options.

The "thknss", "styl", and "clr" options are what one would expect. (the Maple options "color", "style", "thickness", etc. can also be used with DL and have the same meaning). By convention "A" is the left end point and "B" is the right. The left and right "shrinkfactors" are fractions of the left or right end of the line which are not displayed. Thus if the left and right shrinkfactors are both .25 only the middle half of the line is displayed.

All of the other options have to do with placing hashmarks on the line to construct geometric diagrams. The "hashnum" is the number of marks, "hashlength" is their length and "hashspacing" is their separation as a fraction of that of the line itself. The "hashlocation" is the the position of the center of the set of hashmarks (as a fraction of the distance from A to B). Finally "hashrotation" is an angle in radians through which the hashmarks are to be rotated. In constrained scaling the hashmarks are perpendicular to the line. However in unconstrained diagrams they may appear to be rotated. The hashrotangle allows one to rotate them so that they appear normal in the diagram.

There are a fairly large number of pre-defined colors in Maple. We stick to these at first although, as we will see, it is simple to define one's own colors (see help on **plot[color]**). The MCtools command "colors" will display the "builtin" colors.

```
> MCtools[colors]();
```


```
> mctools(PT);
```

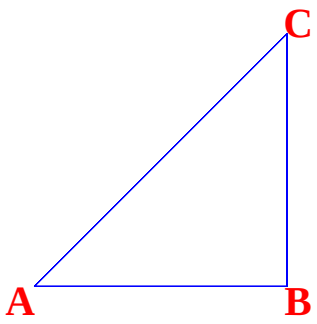
```
MCtools, version Oct 15 2003.

PT(Location,txt,fontsize=16,clr=black)
```

PT ("Put Text") places the string given by the variable "txt" , centered at the point specified by the variable "location", rendered in TIMES NEW ROMAN BOLD in point size specified by the option "font size" and color specified by th option "color".

In general darker colors are best for both line and text color in problem diagrams.

12.2.2 Making a triangle ABC

One of the simplest yet most common figures is a triangle. Some variant of this one appears as a portion of innumerable diagrams in geometry, algebra, calculus, etc:



Making the basic diagram

The figure evidently consists of six primary components: three lines and three text labels

With the **DL** command we can construct each of the lines and with the **PT** command we can construct each label. We begin with the lines. To describe the lines we must first describe the points A,B, and C.

In Maple a point in the plane is described as a list of two real numbers. Recall that lists are delineated by rectangular brackets "[]". If we choose A to be the origin it will be declared

```
> A:=[0,0];
```

$$A := [0, 0]$$

From experience we know that if we then let B = (1,0) and C = (1,1) we will get a triangle of approximately the indiciated shape.

```
> B:=[1,0];
```

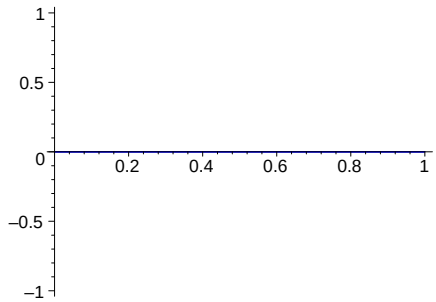
$$B := [1, 0]$$

```
> C:=[1,1];
```

$$C := [1, 1]$$

Now the three lines are defined individually by three separate commands: For example, the segment from A to B is

```
> DL(A,B,thickness=3,linestyle=1,color=blue);
```



Recall that the "3" indicates a thick line and the "1" that it should be unbroken.

As discussed previously and is obvious from the above each of these actually defines a single Maple plot structure. The **sequence** of them describes all of the lines at once. We give that sequence a convenient name that Maple can interpret and which has meaning to us.

```
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),
  DL(A,C,thickness=3,linestyle=1,color=blue),
  DL(B,C,thickness=3,linestyle=1,color=blue);
```

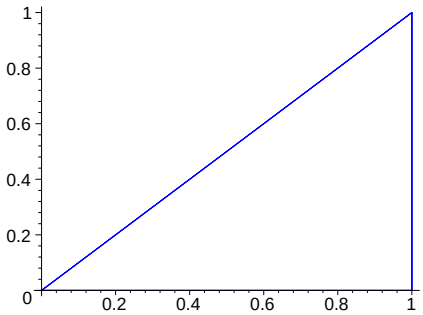
$$LNS := \text{PLOT}(\text{POLYGONS}([[0., 0.], [1., 0.]], \text{COLOUR}(RGB, 0., 0., 1.00000000), \text{THICKNESS}(3), \text{LINESTYLE}(1), \text{STYLE}(LINE))), \text{PLOT}(\text{POLYGONS}([[0., 0.], [1., 1.]], \text{COLOUR}(RGB, 0., 0., 1.00000000), \text{THICKNESS}(3), \text{LINESTYLE}(1), \text{STYLE}(LINE))), \text{PLOT}(\text{POLYGONS}([[1., 0.], [1., 1.]], \text{COLOUR}(RGB, 0., 0., 1.00000000), \text{THICKNESS}(3), \text{LINESTYLE}(1), \text{STYLE}(LINE)))$$

This is the sequence of **plot structures** describing the triangle. Recall that **plots[display]** converts a sequence or list of plot structures into a single plot structure.

```
> TRNG:=plots[display]([LNS]);
```

$$TRNG := \text{PLOT}(\text{POLYGONS}([[0., 0.], [1., 0.]], \text{COLOUR}(RGB, 0., 0., 1.00000000), \text{THICKNESS}(3), \text{LINESTYLE}(1), \text{STYLE}(LINE)), \text{POLYGONS}([[0., 0.], [1., 1.]], \text{COLOUR}(RGB, 0., 0., 1.00000000), \text{THICKNESS}(3), \text{LINESTYLE}(1), \text{STYLE}(LINE)), \text{POLYGONS}([[1., 0.], [1., 1.]], \text{COLOUR}(RGB, 0., 0., 1.00000000), \text{THICKNESS}(3), \text{LINESTYLE}(1), \text{STYLE}(LINE)))$$

```
> TRNG;
```



The reader will likely observe that one could accomplish the same thing without reference to A,B,C, LNS, or TRNG.

```
> plots[display](DL([0,0],[1,0],thickness=3,linestyle=1,color=blue),
  DL([0,0],[1,1],thickness=3,linestyle=1,color=blue),
  DL([1,0],[1,1],thickness=3,linestyle=1,color=blue));
```

This does produce the blue triangle. However it is better practice to take the extra time, and give names to the various parts of the plot. This will give you greater flexibility and ease of use

later. Here we have suppressed unneeded output by replacing semicolons (";") with colons (":") and merged most of the instructions into a single command cell.

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),
  DL(A,C,thickness=3,linestyle=1,color=blue),
  DL(B,C,thickness=3,linestyle=1,color=blue):
> TRNG:=plots[display]([LNS]):
> TRNG;
```

Simply by changing the definition of the points A, B, and C we use this to produce any blue triangle at all. The advantages of keeping such descriptions readable and flexible will be more and more apparent as we develop more interesting diagrams. For now we still have work to do on our triangle.

12.2.3 Developing the Basic Diagram: Adding Text

So far we have a blue triangle which is missing the labels and accompanied by some cartesian "scaffolding" that we need to suppress. The simple addition **of t**he option " **axes=none** " in the plots[display] command will do this. However while building a diagram is almost always much better to leave the reference frame there – for reference.

The labels "A", "B", "C" are nothing but plot structures. Collectively they comprise the text component of the diagram. We can use the PT command to generate them individually and as a first approximation we simply define a text expression sequence and insert it into the current description of the diagram. The "obvious" thing to try is the following which is almost right but may produce some unexpected results

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:


> mctools(PT);
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),DL(A,C,thickness=3,li
  nestyle=1,color=blue),DL(B,C,thickness=3,linestyle=1,color=blue):
> TXT:=PT(A,A,fontsize=16,color=red),PT(B,B,fontsize=16,color=red),PT(C
  ,C,fontsize=16,color=red):
> TRNG:=plots[display]([LNS,TXT]):
> TRNG;
```

The declaration A:=[0,0]: tells Maple to substitute [0,0] for A. Thus in the **PT** command we need to tell Maple that we want the string "A" rather than the value of the variable A. Fortunately this is simple. Whenever an expression is contained in double quotes (" ") the resulting expression is interpreted literally as the string of characters in inside the quotes, rather than as the any value that string may have been previously assigned. Thus what we wanted (almost) is:

```
> A:=[0,0]:
```

```
> B:=[1,0]:
> C:=[1,1]:
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),DL(A,C,thickness=3,li
  nestyle=1,color=blue),DL(B,C,thickness=3,linestyle=1,color=blue):
  TXT:=PT(A,"A",fontsize=16,color=red),PT(B,"B",fontsize=16,color=red),P
  T(C,"C",fontsize=16,color=red):
> TRNG:=plots[display]([LNS,TXT]):
> TRNG;
```

This is much closer to the objective. As mentioned earlier we can remove the reference frame simply by adding an option to the plots[display] command so the only problem is that the letters are not perfectly placed and they are too small.

12.2.4 Selecting Font Size

Increasing the size of the letters is simply a matter of increasing the font size in the **PT** command. Thus we can make several changes and select one that suits us. We try font sizes 60, 40, and 30 point for "A", "B", and "C", respectively.

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),DL(A,C,thickness=3,li
  nestyle=1,color=blue),DL(B,C,thickness=3,linestyle=1,color=blue):
  TXT:=PT(A,"A",fontsize=60,color=red),PT(B,"B",fontsize=40,color=red),P
  T(C,"C",fontsize=20,color=red):
> TRNG:=plots[display]([LNS,TXT]):
> TRNG;
```

The 40-point B is interesting so we change A and C to that size, realizing that it is a simple matter to change to another size later. We still have to adjust the placement of the characters. The problem is that the command **PT**(A,"A",fontsize=50,color=red) instructs Maple to place an "A" centered precisely at the point described by A (in this case [0,0]). We need to adjust the placement. This is done simply by adding (or subtracting) and **offset** to each of their placements. In other words we change **PT**(A,"A",fontsize=50,color=red) to **PT**(A+offset,"A",fontsize=50,color=red) where "offset" is a vector we add to the point A to shift it to where we want the center of the letter "A" to be. Before choosing the final font size, however, we need to adjust the size of the diagram to the approximate size we will want it to be when used.

12.2.5 Resize the Diagram

The diagram initially produced by Maple is usually much larger than we need to illustrate a problem. There is, at present, no way to specify the display sizes of plots. However the size of the diagram produced by a command can be adjusted visually simply by clicking the mouse on a corner and draging the corner until the approximate correct size is attained. Then, the next time that code is executed the diagram will be the selected size. The reason for doing this is that **the size of text placed in a diagram does not change as the diagram is re-scaled.** Consequently, the font size and amount of text offset required for a particular diagram depends on the size at

which it will be used. This is why we *select the size and offsets last* , after selecting the size of the diagram.

After shrinking our triangle to the approximate dimensions at which we will use it even the 40 point text looks a bit outsized and we change to 30 point (the reader may prefer it even smaller) before selecting the offsets. For ease of reference we reproduce the entire construction

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),DL(A,C,thickness=3,li
  nestyle=1,color=blue),DL(B,C,thickness=3,linestyle=1,color=blue):
  TXT:=PT(A,"A",fontsize=30,color=red),PT(B,"B",fontsize=30,color=red),P
  T(C,"C",fontsize=30,color=red):
> TRNG:=plots[display]([LNS,TXT]):
> TRNG;
```

Although it can be automated to a degree the selection of offsets is primarily a matter of esthetics (after all, what is really wrong with the diagram we have?). So even when it can partially be automated there will almost always remain some manual tweaking to be done. In the current example we can see with reference to the coordinate axes that the characters need to be parallel to the axes about .1 - in different directions. For instance we might replace A = (0,0) by A = (-.1,-.1). To do this we can take advantage of the fact that Maple's "+" operator adds lists of the same length coordinatewise. That is [a,b]+[c,d] = [a+c,b+d]. So we could do somehting like the following:

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),DL(A,C,thickness=3,li
  nestyle=1,color=blue),DL(B,C,thickness=3,linestyle=1,color=blue):
  TXT:=PT(A+[-.1,-.1],"A",fontsize=30,color=red),PT(B+[.1,-.1],"B",fonts
  ize=30,color=red),PT(C+[.1,.1],"C",fontsize=30,color=red):
> TRNG:=plots[display]([LNS,TXT]):
> TRNG;
```

12.2.6 Improving the Style

The above has the desired effect but it is not done with very good style. The problem is that the "offset vectors" are embedded in our code. Their inclusion makes the "TXT" line much less readable and there are six things to change if we want to proportionally move each of the letters. If we declare an OFFSET parameter at the beginning of the problem and interpret each of the individual offsets in terms of that value we have something that is easier to read and much easier to modify.

```
> A:=[0,0]:  B:=[1,0]:  C:=[1,1]:
> OFFSET:=.1:
```

```
> LNS:=DL(A,B,thickness=3,linestyle=1,color=blue),
  DL(A,C,thickness=3,linestyle=1,color=blue),
  DL(B,C,thickness=3,linestyle=1,color=blue):
> TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=30,color=red),
  PT(B+[OFFSET,-OFFSET],"B",fontsize=30,color=red),
  PT(C+[OFFSET,OFFSET],"C",fontsize=30,color=red):
> TRNG:=plots[display]([LNS,TXT]):


> TRNG;
```

Note: in simple diagrams (and most of them are simple) it works well to use a single offset parameter. For more complex situations it can be convenient to use two or more.

Once we realize the utility of declaring all paramters outside the actual graphics instructions we note that the point sizes, colors, and even line thicknesses and styles are also parameters and we can/should declare them externally too. By this time we need to remind ourselves what things are. Recall that the Maple ignores anything on an execution line which follows a pound sign ("#").

Finally, we add the option "axes=none" to the plots[display] command to hide that bit of scaffolding from view and merge all of the instructions into a single input cell (with "F4" key or the "edit/split" menu).

```
> # vertices of the triangle
> A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
> #vertex label offset
> OFFSET:=.1:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=blue:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+[OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
> TRNG:=plots[display]([LNS,TXT],axes=none):
> TRNG;
```

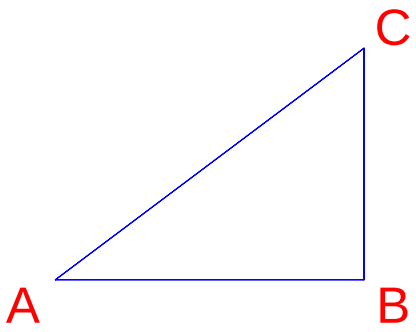
At this point most modifications to the triangle can be made simply by changing parameters

rather than delving into the code. Some "tweeking" typically remains to be done. Most readers would probably prefer that the "B" and "C" labels not come into contact with the triangle. The "A" is well-placed, however so we don't want to change the offset globally. The simplest thing to do is move the "B", for instance, by editing the " **PT**(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTC command to say, double the horizontal offset by changing it to " **PT**(B+[2*OFFSET,-OFFSET],"B",fontsize=TEX)" . These changes are left to the reader.

Finally, it is a good idea to section off the construction of a diagram. Once the diagram is done for a particular problem we seldom want to view the code again although we do want it handy. Such sections are easily copied and pasted into other problems or other worksheets.

12.2.7 Basic Triangle Diagram

```
> # vertices of the triangle
> A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
> #vertex label offset
> OFFSET:=.1:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=blue:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+[OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
> TRNG:=plots[display]([LNS,TXT],axes=none):
> TRNG;
```

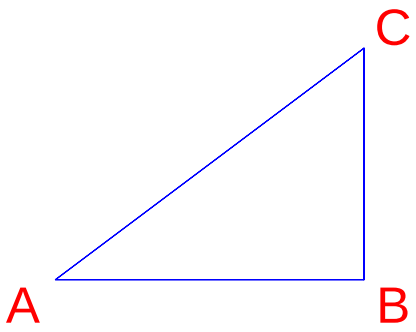


12.2.8 Making up a problem to go with the diagram

Often the diagram in a problem comes first. Once you have a picture to look at you can come with with an interesting question about it. Here's one.

```
> tagit(TRNG,"What it area of triangle ABC, if AB = 3 and BC =
  4?",_AC(3*4/2));
```

```
QM_[0.05;6]

AH_[0]
```




```
What it area of triangle ABC, if AB = 3 and BC = 4?

AC_[5]

SKIP_
```

12.2.9 Notes

1. **Name Conventions**: We have used a naming convention for component groups of our figure which consists of taking a descriptive word for the group, making it all upper case, and deleting as many of the vowels as we can while retaining the sense of the word. This we use LNS for "lines"

and TXT for "text". In principle there is nothing wrong with using the entire word (including the vowels) in lower or mixed case. The problem is that one may run into Maple "reserved words" and receive mysterious and frustrating error messages such as "**.. attempted recursive definition ..**" or ".. reserved word .."

"D", "I", and "O" are reserved words in Maple: they cannot be assigned alternative values. Even the most experienced user will periodically forget and attempt to assign a value to one of these which constructing a diagram. The author generally elects to use "DD", "II", and "OO". Thus for instance if we had been making a square rather than a triangle the points might have been

A:=[0,0]: B:=[1,0]: C:=[1,1]: DD:=[0,1]:

Note that **PT**(DD,"D",fontsize=30,color=green); would work fine since "D" refers to the character string "D" rather than the value Maple assigns the variable D. Generally one must be aware of the possibility of inadvertently encroaching on reserved words and be prepared to encounter them with changes in spelling, use of capitalization , etc.

2. **"Quick and Dirty" Offsets** In the above we used method of inspecting the diagram (after resizing) and estimating the offset from the reference frame in the picture. There is another, very convenient method. Simply place the mouse pointer at the point one would like to have the text centered. Then click the mouse and the coordinates of that point will appear in a small window in the upper left corner th window. For instance in our diagram if we place the mouse pointer at a point and click the coordinates apperar in the upper left corner. So if we place the pointer about where we would like to see the "C" centered we can make the coordinates of that position appear in the coordinate bar from where they can be copied. It is a simple matter to highlight the pair of coordinates, copy them, and then paste them into the "location" slot of the **PT** command

PT(A,"A",30,red) to get **PT**(0.60, 0.33 ,"A",fontsize=30,color=red) to which we need to add the brackets to finish with **PT**([0.60, 0.33],"A",fontsize=30,color=red). The result would be to move the "A" to the point (.6,.33) in the diagram.

This will work even with the coordinate frame suppressed as Maple still has to maintain those references to know where to place things.

When doing a "quick and dirty" diagram for an in-class quiz, a slide for a talk, etc. this is perfectly reasonable. However it is generally a **much** better idea to retain the "A" and do a little subtracting and rounding to get an offset.

You can even get Maple to do the calculation. Just start with the following line in an execution cell

> [ ] - C ;

and paste the coordinates into the brackets

```
> [1.09, 1.12] -C;
```

This gives us an offset which we can use directly by copying and pasting.

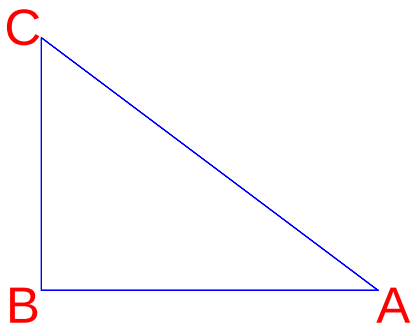
PT(C,"C",fontsize=30,color=red) becomes **PT**(C +[1.09,1.12] ,"C",fontsize=30,color=red):

Why bother with offsets, parameter declarations, etc?

For portability. Creating good diagrams is hard, time consuming work which one does not want to repeat. By developing our diagrams in this way, we can use this code to create **any** other triangle simply by redefining A,B, and C. The offsets of the resulting triangle may need a little adjusting but as we have seen that is very easy as long as the picture is approximately right. For instance suppose we really wanted the triangle ABC where A= (3,0) ,B=(2,0) , and C=([2,1). Most of the work is done simply by taking (a copy of) the original Basic Triangle Diagram and changing the declarations of A, B, and C. We can also adjust the offset by taking out the axes=none option and executing the code. This will aid in determining a suitable offset.

12.2.10 Basic Triangle Diagram (modified with above TXT line)

```
> # vertices of the triangle
> #A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
  A:=[3,0]:    B:=[2,0]:    C:=[2,1]:
> #vertex label offset
> #OFFSET:=.1:
  OFFSET := .05:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=blue:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  #TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> #PT(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  #PT(C+[OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
  TXT:=PT(A+[OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+[-OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+[-OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
> TRNG:=plots[display]([LNS,TXT],axes=none):
> TRNG;
```



The small amount of time invested in a systematic approach such as we have illustrated here will pay handsome dividends in reusable, adaptable, sharable materials, time savings, and reduced frustration.

12.2.11 Turning the Basic diagram into a procedure.

Once you have developed a diagram in the above fashion, it become very easy to convert it into a drawing procedure. All we need to do is add a proc line and an end line.

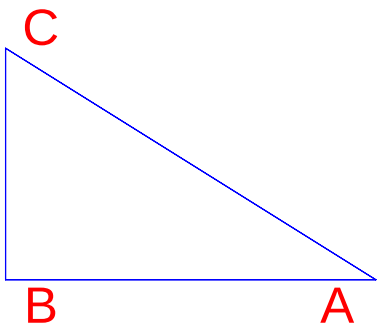
```
> trng := proc(A,B,C)
  # vertices of the triangle
> #A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
> #vertex label offset
> OFFSET:=.1:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=blue:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+[OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
> TRNG:=plots[display]([LNS,TXT],axes=none):
```

```
> TRNG;
  end:
```

```
Warning, ‘OFFSET‘ is implicitly declared local to procedure ‘trng‘

Warning, ‘EDGETHICKNESS‘ is implicitly declared local to procedure
‘trng‘

Warning, ‘EDGESTYLE‘ is implicitly declared local to procedure ‘trng‘

Warning, ‘EDGECOLOR‘ is implicitly declared local to procedure ‘trng‘

Warning, ‘TEXTSIZE‘ is implicitly declared local to procedure ‘trng‘

Warning, ‘TEXTCOLOR‘ is implicitly declared local to procedure ‘trng‘

Warning, ‘LNS‘ is implicitly declared local to procedure ‘trng‘

Warning, ‘TXT‘ is implicitly declared local to procedure ‘trng‘

Warning, ‘TRNG‘ is implicitly declared local to procedure ‘trng‘
```

The warnings can be ignored, but in we can take care of these by putting in a local line under the proc line. Also, while we are doing that we can decide which of the local variables should be input parameters rather than set in the procedure. The offset and edgecolor are ones which I would add in first.

```
> trng := proc(A,B,C,offset,edgecolor)
  local OFFSET,EDGETHICKNESS,EDGESTYLE, EDGECOLOR, TEXTSIZE, TEXTCOLOR,
  LNS, TXT, TRNG;
  # vertices of the triangle
> #A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
> #vertex label offset
  OFFSET:=offset:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=edgecolor:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+[OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
> TRNG:=plots[display]([LNS,TXT],axes=none):
> TRNG;
  end:
> trng([3,0],[2,0],[2,1],.1,blue);
```

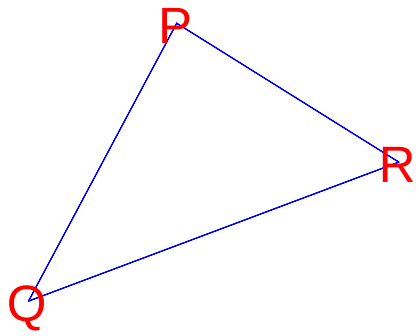


There are a couple of things wrong with this procedure. First, we need better control over the offsets. Second, we need to be able to change the labels. Let's talk about changing the labels. We can make the names of the labels inputs to the procedure, call them LA, LB, and LC. Then we modify the TXT line to use those, by replacing "A" with LA, etc. Now for the offsets: We will allow each vertex its own offset and it will be a list of two small numbers. Call them OA, OB, and OC.

```
> trng := proc(A,B,C,LA,LB,LC,OA,OB,OC,edgecolor)
  local OFFSET,EDGETHICKNESS,EDGESTYLE, EDGECOLOR, TEXTSIZE, TEXTCOLOR,
  LNS, TXT, TRNG;
  # vertices of the triangle
> #A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
> #vertex label offset
  OFFSET:=offset:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=edgecolor:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  TXT:=PT(A+OA,LA,fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+OB,LB,fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+OC,LC,fontsize=TEXTSIZE,color=TEXTCOLOR):
> TRNG:=plots[display]([LNS,TXT],axes=none):
> TRNG;
  end:
```

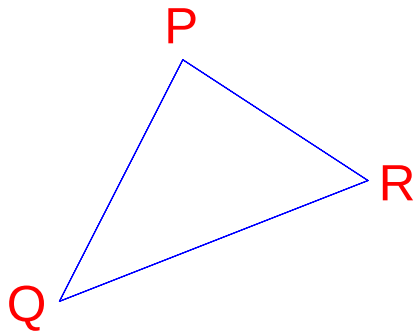
Now when we test, put single quotes around the names. Then if they have been used as names, this won't interfere. Start with the offsets at [0,0]. You can resize the diagram to where you want it, select the diagram, right click it and turn on the axes. Then you can estimate the needed offsets quickly.

```
> trng([3,4],[1,2],[6,3],'P','Q','R',[0,0],[0,0],[0,0],blue);
```



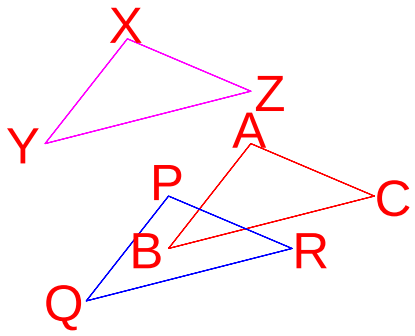
By inspection, we see that OA = [0,.5], OB = [-.5,0], and OC=[.5,0] will work.

```
> trng([3,4],[1,2],[6,3],'P','Q','R',[0,.3],[-.5,0],[.5,0],blue);
```



Now we can use this procedure to draw several triangles. Suppose we want 3 congruent triangles with different colors. We could do something like:

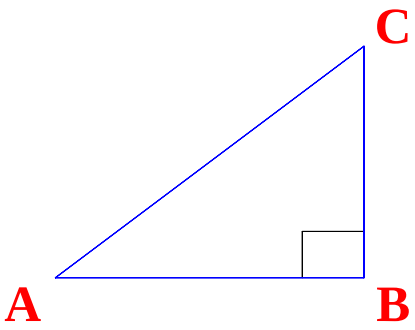
```
> TRNGL1 :=
  trng([3,4],[1,2],[6,3],'P','Q','R',[0,.3],[-.5,0],[.5,0],blue):
> TRNGL2 :=
  trng([3,4]+[2,1],[1,2]+[2,1],[6,3]+[2,1],'A','B','C',[0,.3],[-.5,0],[.
  5,0],red):
> TRNGL3 :=
  trng([3,4]+[-1,3],[1,2]+[-1,3],[6,3]+[-1,3],'X','Y','Z',[0,.3],[-.5,0]
  ,[.5,0],magenta):
> plots[display](TRNGL1,TRNGL2,TRNGL3);
```



12.3 Some Additional Examples

Example: Adapt the previous example to to indicate that the triangle is a right triangle by adding two small, black line segments, one parallel to each side.

Solution: The exercise asks use to create something like the following



This is accpmplished simply by adding a new construct which we label RTANGL below. It consists of two lines and might have been included in the LNS construct. There are two reasons not to do it that way. First, the code is more readable when the expression sequences comprising the components are not lengthy. Second, and much more important is that each of our component constructs is comprised of very similar commands. It is usually the case that once the first of such a sequence is constructed then the rest are very minor and very logical modifications of the first. This often permits us to construct the component by getting the first one right, copying and pasting additional copies of the first to it, and then quickly going down that sequence making changes according to some evident pattern.

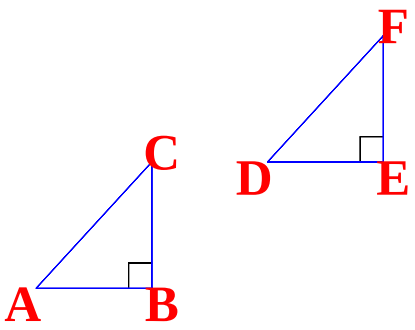
Since the color, thicknesses, and style for the RTANGL symbol are all contained in the single line and unlikely to change we don't bother to declare them independently (although in a more elaborate diagram in which there were several of these we almost certainly would.). Notice also that we have not forgotten to append the RTANGL construct to the list in the plots[display] command.

12.3.1 Basic Triangle Diagram(with rt angle symbol)

```
> # vertices of the triangle
> A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
> #vertex label offset
> OFFSET:=.1:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=blue:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+[OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
  #rt angle symbol
> RTANGL:=DL(B-[2*OFFSET,0],B-[2*OFFSET,0]+[0,2*OFFSET],thickness=1,line
  style=1,color=black),DL(B-[2*OFFSET,0]+[0,2*OFFSET],B+[0,2*OFFSET],thi
  ckness=1,linestyle=1,color=black):
> TRNG:=plots[display]([LNS,TXT,RTANGL],axes=none):
> TRNG;
```

Example: Add a second triangle, congruent to the first to the diagram by shifting the original triangle 2 units to the right and up 1. Let the second triangle be green with vertices labeled D, E, and F

Solution: We are asked to produce the following (or a similar one with the D,E,F labels permuted):



All we want to do is create to plot structures and merge them with a plots[display] command. Let us call our original triangle TRNG1. Then we modify a duplicate of it to make TRNG2. All we have to do is add the vector (2,1) to each of the vertices of the original triangle and rename the vertices. Since this will be done more than once and is likely something to be done again we identify the shift (2,1) as a parameter which we call SHIFT. Here also we encounter for the first time the unfortunate fact that "D" is a reserved word in Maple. As a workaround we use "DD".

12.3.2 Basic Triangle Diagram (TRNG1)

```
> # vertices of the triangle
> A:=[0,0]:    B:=[1,0]:    C:=[1,1]:
> #vertex label offset
> OFFSET:=.1:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
  EDGESTYLE:=1:
> #edge color
  EDGECOLOR:=blue:
> #text size (in points)
> TEXTSIZE:= 30:
  #text color
  TEXTCOLOR:=red:
  #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),
  DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR),
> DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
  #vertex labels
  TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),
> PT(B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),
  PT(C+[OFFSET,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
  #rt angle symbol
> RTANGL:=DL(B-[2*OFFSET,0],B-[2*OFFSET,0]+[0,2*OFFSET],thickness=1,line
  style=1,color=black),DL(B-[2*OFFSET,0]+[0,2*OFFSET],B+[0,2*OFFSET],thi
  ckness=1,linestyle=1,color=black):
> TRNG:=plots[display]([LNS,TXT,RTANGL],axes=none):
```

```
> TRNG1:=TRNG:
```

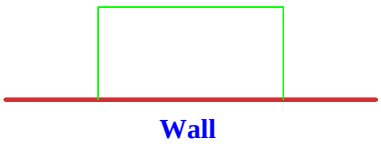
Now we simply "glue" the two triangles together

```
> TRNG3:=plots[display]([TRNG1,TRNG2]):
> TRNG3;
```

Example: A standard calculus problem asks for the largest rectangular area that can be bounded by a fixed amont of fencing if one edge of the enclosure lies on an extant wall. Construct a diagram to illustrate this problem

Solution:

The problem asks for a picture something like the one below:



Here we take advantage of the fact that we can make lines of different thicknesses.

In keeping with our desire to make every aspect of the diagram a function of our choice of points (up to text offsets) we place the label for the wall at approximately the point midway between the two points of contact of the fence with the wall - with an appropriate offset. Note the "scaling = constrained" option in the plots[display] command. This forces Maple to use rectangular coordinates where the scales on the x- and y-axes are the same. In the absence of this command Maple tries to adjust the aspect ratio of the figure to display the most information in the space available. While very useful this is not always exactly what we want to see. The reader should remove the option and observe the results.

12.3.3 Diagram: Fence with wall

```
> #fence
> A:=[0,0]:
> B:=[0,1]:
> C:=[2,1]:
> DD:=[2,0]:
  FNCTHICKNESS:=2:
  FNCSTYLE:=1:
> FNCCOLOR:=green:
  # wall
```

```
> E:=[-1,0]:
> F:=[3,0]:
  WALLTHICKNESS:=9:
  WALLSTYLE:=1:
> WALLCOLOR:=orange:
> #text
  TXTOFFSET:=.3:
> TXTSIZE:=16:
  TXTCOLOR:=blue:
> FNC:=DL(A,B,thickness=FNCTHICKNESS,linestyle=FNCSTYLE,color=FNCCOLOR)
  ,DL(C,B,thickness=FNCTHICKNESS,linestyle=FNCSTYLE,color=FNCCOLOR),DL(C
  ,DD,thickness=FNCTHICKNESS,linestyle=FNCSTYLE,color=FNCCOLOR):
> WLL:=DL(E,F,thickness=WALLTHICKNESS,linestyle=WALLSTYLE,color=WALLCOL
  OR):
> TXT:=PT(1/2*(A+DD)+[0,-TXTOFFSET],`Wall`,fontsize=TXTSIZE,color=TXTCO
  LOR):
> FENCEWITHWALL:=plots[display]([FNC,WLL,TXT],scaling=constrained,axes=
  none):

> FENCEWITHWALL;
```

Placing a Graph on Graph Paper: GP

One frequently finds it convenient to place a graph on "graphpaper" rather than simply have it located relative to coordinate axes with tickmarks. The command "graph paper" , **GP** is useful for doing this. Its syntax is:

```
> mctools(GP);

> ;
```

GP ("Graph Paper") has no variables, only options.
llcorner = the point [a,b] which is the lower left corner of the graphpaper
width = the width of the graphpaper
length = the length of the graphpaper
resolution = number of grid squares sides covering the interval [0,1] . The values 1,2,3,4 are the most common.
tickmarkfont = font in which tickmarks are printed

Notes: A tickmarkfontsize of 1 gives no tickmarks (this is used very frequently). **GP** is primitive - it puts tickmarks only at integer values.

The default value of **GP** is often good enough to get started

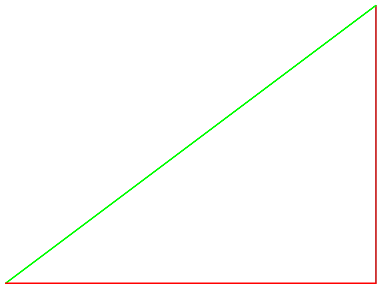
```
> GP();


> GP(llcorner=[-2,-2],wifth=6,height=10,resolution=1,tickmarkfont=5,axe
  scolor=blue);
> gp:=GP(llcorner=[-2,-2],width=6,height=6,resolution=1,tickmarkfont=1,
  axescolor=gray):
  gp;
```

Had we wanted to place the previous wall diagram on this graph paper we simply

```
> plots[display]([FENCEWITHWALL,gp],scaling=constrained,axes=none);
```

12.4 Exercises

Exercise 1: Construct the following figures



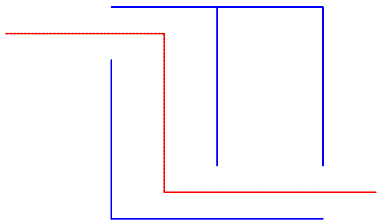
a. (The colors are red, green, and orange).
Start

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> LNS:=DL(A,B),DL(A,C),DL(B,C):
> TRNG:=plots[display]([LNS],axes=none):
  TRNG;
```

b. (The colors are green, orange, red, magenta; the thicknesses are 2,5,6,8).
code (start)

```
> A:=[0,0]:
> B:=[1,0]:
```

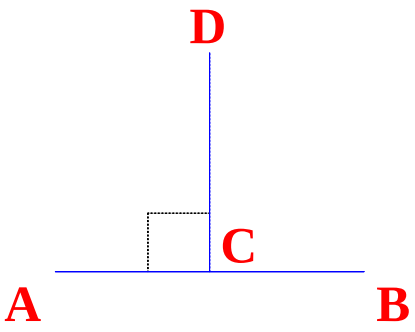
```
> C:=[1,1]:
  DD:=[0,1]:
> LNS:=DL(A,B),DL(A,DD),DL(B,C),DL(DD,C):
> TRNG:=plots[display]([LNS],axes=none):
  TRNG;
```



c. (the broken linestyle is 4. Note the appearance of broken, dotted, dashed linestyles is will change as image sizes are adjusted. When the images shrinks dashed lines appear continuous. This is less pronounced on thin lines)

c code (start)

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
  DD:=[0,3/4]:
  E:=[0,1]:
> F:=[1,1/4]:
  G:=[1/2,1/4]:
  H:=[1/2,1]:
> II:=[-1/2,7/8]:
  J:=[1/4,7/8]:
  K:=[1/4,1/8]:
  L:=[5/4,1/8]:
> LNS:=DL(A,B),DL(A,DD),DL(F,C),DL(E,C):
  PTH:=DL(II,J),DL(J,K),DL(K,L):
> TRNG:=plots[display]([LNS,PTH],axes=none,scaling=constrained):
  TRNG;
```



d. (the dashed linestyle is 4, be sure to improve on the placement of the letters)
code (start)

```
> A:=[0,0]:
> B:=[1,0]:
> C:=[1/2,0]:
  DD:=[1/2,3/4]:
> LNS:=DL(A,B):
  TXT:=PT(A+[-.1,-.1],"A"),PT(B+[.1,-.1],"B"),PT(C+[.1,.1],"C"),PT(DD+[0
  ,.1],"D"):
> RTANGL:=DL(C-[.2,0],C-[.2,0]+[0,.2]),DL(C-[.2,0]+[0,.2],C+[0,.2]):
> TRNG:=plots[display]([LNS,TXT,RTANGL],axes=none,scaling=constrained):
> TRNG;
```

Exercise 2:: Add two problems to the following pair and post them to WHS

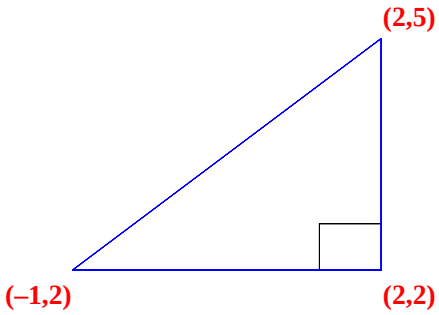
12.4.1 WHS problem: length of hypotenuse

MC_[.001;3*sqrt(2)]

AH_

To within .001 the length of the hypotenuse of the right triangle shown in the diagram is AC_[7]

.



SKIP_

12.4.2 calculations

```
> sqrt((5-2)^2+(2-(-1))^2);
```

12.4.3 diagram

```
> # vertices of the triangle
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> #vertex label offset
> OFFSET:=.1:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
> EDGESTYLE:=1:
  #edge color
> EDGECOLOR:=blue:
> #text size (in points)
> #TEXTSIZE:= 30:
  TEXTSIZE:= 16:
> #text color
  TEXTCOLOR:=red:
> #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR
  ),DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
> #vertex labels
> TXT:=PT(A+[-OFFSET,-OFFSET],"(-1,2)",fontsize=TEXTSIZE,color=TEXTCOLOR
  ),PT(B+[OFFSET,-OFFSET],"(2,2)",fontsize=TEXTSIZE,color=TEXTCOLOR),PT(
  C+[OFFSET,OFFSET],"(2,5)",fontsize=TEXTSIZE,color=TEXTCOLOR):
> RTANGL:=DL(B-[2*OFFSET,0],B-[2*OFFSET,0]+[0,2*OFFSET],thickness=1,line
> style=1,color=black),DL(B-[2*OFFSET,0]+[0,2*OFFSET],B+[0,2*OFFSET],thi
  ckness=1,linestyle=1,color=black):
> TRNG:=plots[display]([LNS,TXT,RTANGL],axes=none):
> TRNG;
```

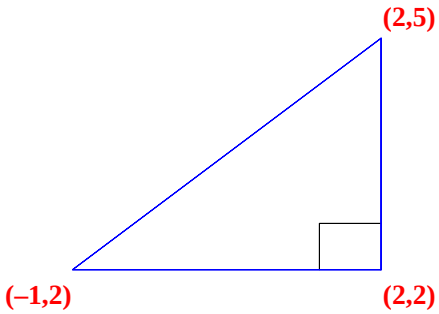
12.4.4 WHS problem: length of hypotenuse

MC_[.001;3*sqrt(2)]

AH_

To within .001 the length of the hypotenuse of the right triangle shown in the diagram is AC_[7]

.



SKIP_

12.4.5 calculations

```
> sqrt((5-2)^2+(2-(-1))^2);
```

12.4.6 diagram

```
> # vertices of the triangle
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> #vertex label offset
> OFFSET:=.1:
> #edge thickness
> EDGETHICKNESS:=3:
  # edge line style
> EDGESTYLE:=1:
  #edge color
> EDGECOLOR:=blue:
> #text size (in points)
> #TEXTSIZE:= 30:
  TEXTSIZE:= 16:
> #text color
  TEXTCOLOR:=red:
> #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR
  ),DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
> #vertex labels
> TXT:=PT(A+[-OFFSET,-OFFSET],"(-1,2)",fontsize=TEXTSIZE,color=TEXTCOLOR
  ),PT(B+[OFFSET,-OFFSET],"(2,2)",fontsize=TEXTSIZE,color=TEXTCOLOR),PT(
  C+[OFFSET,OFFSET],"(2,5)",fontsize=TEXTSIZE,color=TEXTCOLOR):
> RTANGL:=DL(B-[2*OFFSET,0],B-[2*OFFSET,0]+[0,2*OFFSET],thickness=1,line
> style=1,color=black),DL(B-[2*OFFSET,0]+[0,2*OFFSET],B+[0,2*OFFSET],thi
  ckness=1,linestyle=1,color=black):
```

```
> TRNG:=plots[display]([LNS,TXT,RTANGL],axes=none):
> TRNG;
```

12.4.7 WHS problem: sine of an angle

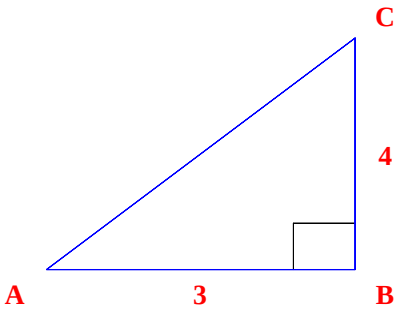
MC_[0;4/5]

AH_

With the lengths of the triangle edges as indicated the sine of the angle CAB in the diagram is

AS_[3/5;1/5;5/4;4/5;1]

.



SKIP_

12.4.8 diagram start)

```
> # vertices of the triangle
> A:=[0,0]:
> B:=[1,0]:
> C:=[1,1]:
> #vertex label offset
> OFFSET:=0:
> #edge thickness
> EDGETHICKNESS:=1:
  # edge line style
> EDGESTYLE:=1:
  #edge color
> EDGECOLOR:=black:
> #text size (in points)
> #TEXTSIZE:= 12:
  TEXTSIZE:= 12:
> #text color
  TEXTCOLOR:=black:
> #edges of the triangle
> LNS:=DL(A,B,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOL
  OR),DL(A,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR
  ),DL(B,C,thickness=EDGETHICKNESS,linestyle=EDGESTYLE,color=EDGECOLOR):
```

```
> #vertex labels
  TXT:=PT(A+[-OFFSET,-OFFSET],"A",fontsize=TEXTSIZE,color=TEXTCOLOR),PT(
> B+[OFFSET,-OFFSET],"B",fontsize=TEXTSIZE,color=TEXTCOLOR),PT(C+[OFFSET
  ,OFFSET],"C",fontsize=TEXTSIZE,color=TEXTCOLOR):
  LENGTHS:=PT(1/2*(A+B)+[0,-OFFSET],"3"),PT(1/2*(C+B),"4"):
> TRNG:=plots[display]([LNS,TXT,LENGTHS],axes=none):
  TRNG;
```

Exercise 3:

Modify the FENCEWITHWALL diagram to include an open gate in the fence. Create a problem illustrated by the resulting diagram and post it to MathClass.com

Exercise 4:

Modify the Basic Triangle by dividing the triangle into two parts with a line segment connecting the midpoints of the base and hypotenuse. Add labels indicating the lengths of some of the sides in the resulting figure. Create a similar triangles problem illustrated by the resulting diagram.

Exercise 5:

Modify the Basic Triangle (without right angle symbol) to display an equilateral triangle. Connect the midpoints of two of the edges with a line segment. Provide number labels for some of the sides from which it can be inferred that the triangle is equilateral and the new edge connects midpoints. Create a problem about the area of one of the sub regions which is illustrated by the resulting diagram.

13 Creating Diagrams for Mathematics Problems: Tutorial 2

13.1 Adding Circles and Arrows to Diagrams

As we add additional tools the importance of a systematic approach will become more and more evident. Even for the previous, simple diagrams we have emphasized the importance of separately identifying the reference points for the diagram and constructing the instructions relative to those points. In this section the addition of circles, circular arcs, and arrows will greatly expand the complexity of the figures we can produce.This will make adherence to the principle of identifying the problem reference parameters and constructing the basic figure realtive to them even more important.

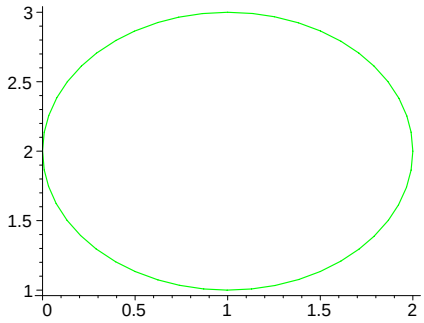
13.1.1 Circles

```
> mctools(PC);
```

```
MCtools, version Oct 15 2003.

PC(center,radius, thknss=2, styl=1, clr=black,scalefactor=1)
```

The forms "thickness", "linestyle", and "color" should be used in place of "thknss", "styl" and "clr".

```
> JOE:=PC([1,2],1,thickness=3,linestyle=1,color=green):JOE;
```



PC does not automatically invoke "scaling = constrained" or "axes=none" since sometimes one wants to omit one or both of them. One generally wants to specify constraining options in the final plots[display] statement so the circles may appear elliptical when viewed independently. Clicking on the plot and then selecting "1-1" at the toolbar will render it constrained. This can also be done by right clicking on the plot, selecting "projection" and then "constrained".

13.1.2 Circlular arcs

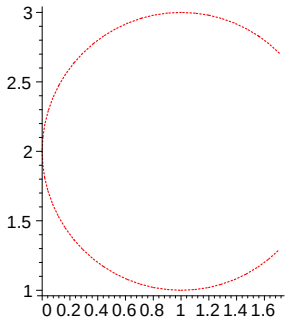
```
> mctools(PA);
```

```
MCtools, version Oct 15 2003.

PA(center,radius,theta1,theta2, thknss=3, styl=1,
clr=black,scalefactor=1)
```

PA for "place arc" is very similar to "PC".

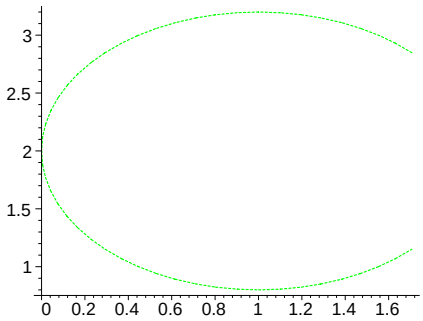
It draws the circular arc with the specified thickness and linestyle along the circle described parametrically by P(t) = center + radus*[cos(t), sin(t)] which corresponds to the values of t between theta1 and theta2. The same observation regarding scaling and axes apply to PA as to PC.

```
> PA([1,2],1,Pi/4,7*Pi/4,
  thickness=3,linestyle=4,color=red,scaling=constrained);
```



The above diagram is not a circular arc since the rendering is not to scale. This can be corrected by displayng it with scaling =constrained. Arcs, however, are usually part of larger diagrams which may not be constrained. In cases where one wants the arc to appear circular even though the axes have different scales one uses the scalefactor which is the ratio of the (perceived) unit lengths on the X and Y axes. In the above this appears to be about .5 so.

```
> PA([1,2],1,Pi/4,7*Pi/4,
  thickness=3,linestyle=2,color=green,scalefactor=1.2);
```



13.1.3 Arrows

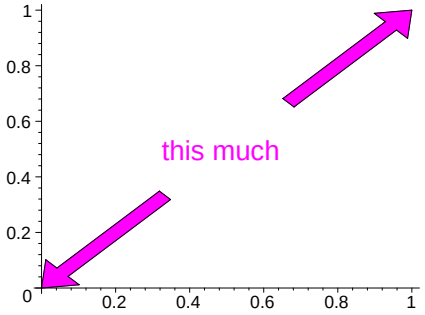
```
> mctools(ARRW);
```

```
MCtools, version Oct 15 2003.

ARRW(tail=[0,0],head=[1,2],headwidth=3,headlength=4,shaftthickness=.03
,clr=green,arrowtype=`DH`,doublearrowtxt="texthere",fontsize=16,txtclr
=black,dhgap=1/3)
```

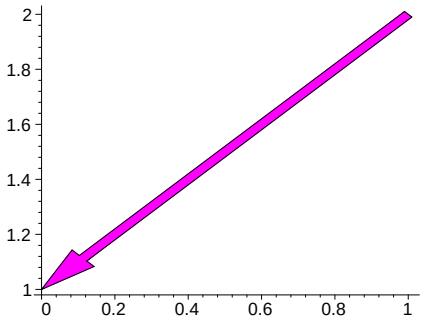
ARRW draws an arrow from point A to point B of the specified color. "DH" specifies a double-headed arrow and "SH" a single headed arrow. The argument "scale factor" is the ratio of the shaft thickness to the arrow length and is usually a number between .01 and .05. It generally must be adjusted to taste. The "doublearrowtxt" is a text string to be placed between two portions of a double-headed arrow. The "dhgap" is the middle fraction of the arrow that will be left blank for the text.

Note: ARRW is a simplification of Maple's plottools[arrow] command which gives the user more control over the shape of the arrow head. ARRW is a compromise that usually provides acceptable results.

```
> ARRW(tail=[0,0],head=[1,1],color=magenta,arrowtype=`DH`,shaftthicknes
  s=.03,doublearrowtxt=`this much`);
```



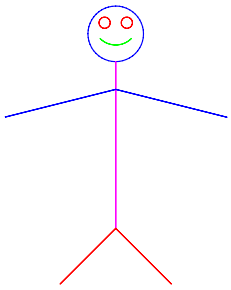

```
> plots[display](ARRW(tail=[1,2],head=[0,1],color=magenta,arrowtype=`SH
  `,shaftthickness=.02,doublearrowtxt=``));
```



13.2 Stickman drawings with lines, circles and arcs:

Example 1: Draw a stickman stick figure

Solution: There is of course no single solution. The figure below is crude but would satisfy the requirements. We have included labels for a number of points which we will suppress in the final version. Their purpose is to emphasize the basic organizational principle of first identifying and labeling parameters for the components of a figure and then constructing the figure relative to those points. A hand sketch is often useful in getting started. Something like this **stickman** .



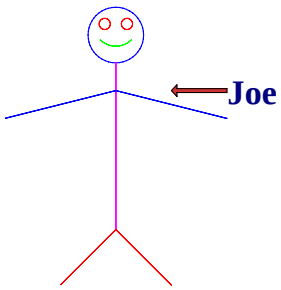
Note that we execute each of these lines as we enter them. This catches many errors. If everything works up to a line and then an error appears at that point we know that line contributed to the problem.

13.2.1 stickman

```
> LNETHICKNESS:=3:#all of the lines are thick
> LNESTYLE:=1:    #all of the lines are unbroken
  #Face parameters
> HDCNTR:=[0,4.5]:# head center
> HDRDS:=.5:# head radius
> HDCLR:=blue:  #head color
  #eyes
  LEYCNTR:=[-.2,4.7]:  # left eye center
> REYCNTR:=[.2,4.7]:# Right eye center
> EYCLR:=red:# eye color
> EYRDS:=.1:  #Eye radius
  #smile
  SMLCNTR:=[0,4.7]:  #smile center
> SMLRDS:=.4:  #smile radius
> SMLANGL1:=5*Pi/4:
> SMLANGL2:=7*Pi/4:
> SMLCLR:=green:# smile color
  #body parameters
  BDYTP:=[0,4]:  #body top
> BDYBDM:=[0,1]:  #body bottom
  BDYCLR:=magenta:  #body color
  #arms
> LHND:=[-2,3]:  #left hand
  RHND:=[2,3]:  #right hand
  SHLDR:=[0,3.5]:#shoulder(s)
> ARMCLR:=blue:  #arm color
  #legs
  LFT:=[-1,0]:  #left foot
> RFT:=[1,0]:  #right foot
  LGCLR:=red:  # foor color
  #construction of body components
```

```
> FACE:=PC(HDCNTR,HDRDS,thickness=3,linestyle=1,color=HDCLR):
  EYES:=PC(LEYCNTR,EYRDS,thickness=LNETHICKNESS,linestyle=LNESTYLE,color
  =EYCLR),
> PC(REYCNTR,EYRDS,thickness=LNETHICKNESS,linestyle=LNESTYLE,color=EYCLR
  ):
  SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=LNETHICKNESS,linest
> yle=1,color=SMLCLR):
  SML:
  BDY:=DL(BDYTP,BDYBDM,thickness=LNETHICKNESS,linestyle=LNESTYLE,color=B
> DYCLR):
  ARMS:=DL(LHND,SHLDR,thickness=LNETHICKNESS,linestyle=LNESTYLE,color=AR
  MCLR),
> DL(RHND,SHLDR,thickness=LNETHICKNESS,linestyle=LNESTYLE,color=ARMCLR):
  LGS:=DL(LFT,BDYBDM,thickness=LNETHICKNESS,linestyle=LNESTYLE,color=LGC
  LR),
> DL(RFT,BDYBDM,thickness=LNETHICKNESS,linestyle=LNESTYLE,color=LGCLR):
  #assemble figure
  STICKMAN:=plots[display]([ARMS,BDY,SML,EYES,FACE]):
> STICKMAN:=plots[display]([LGS,ARMS,BDY,SML,EYES,FACE],scaling=constra
  ined,axes=none):
> STICKMAN;
```

Example 2: Modify the stickman by inserting an arrow pointing to him indicating that he is Joe.



Solution: The above diagram would roughly meet the requirements.

The idea, of course, is to regard the arrow and associated text simply as additional components to be added to the current figure. In this case we can regard the arrow and name as making up a component to which we assign the label "HSNAME" which will be consist of an arrow and some text.

HSNAME:=ARRW(NMARTL,NMARHD,NMARCLR, 'SH', NMARSCLFCT),PT(NMCNTR,NMTXT,NMP

This sketch tells us which parameters we need to specify. The labels are chosen to be as short as possible, not conflict with reserved words, and stil l communicate enough of their function when read in context. The reader can undoubtedly improve on many of them. We need:

NMARTL (name arrow tail)
NMARHD (name arrow head)
NMARSCLFCT (name arrow scale factor)

NMARCLR (name arrow color)
We have elected to "hardwire" the fact that the arrow has a single head into the code, hence the 'SH' enrty

NMCNTR (center of the name text)
NMTXT (name text)
NMPTSZE (name point size)
NMTXTCLR (name text color)
We decide that we want the arrow to be horizontal and about the level of the shoulders.
A reasonable first approximation for the length of the arrow is the diameter of the head.
The scale factor .05 is usually a good starting point.
We arbitrarily make the arrow orange
We begin by placing the text at the tail of the arrow
The minimal text, we are given is "Joe"
We choose turquoise for the text color
Twenty point type is usually a good first approximation

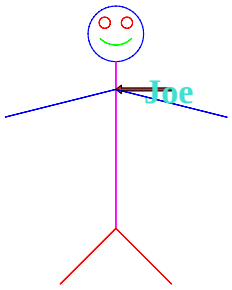
These then are the parameter declarations we will add (we will of course have to tune with offsets)

```
> #name arrow and text parameters
> NMARHD:=SHLDR: #arrow head at shoulder
> NMARTL:=SHLDR+[2*HDRDS,0]:  # arrow length = head diameter
> NMARSCLFCT:=.05:  #.05 scaling factor to start
> NMARCLR:=orange:  #arrow color
> NMCNTR:=SHLDR+[2*HDRDS,0]:  # text starts at arrow tail
> NMTXT:="Joe":
> NMPTSZE:=20:  #start with 20 pt text
> NMTXTCLR:=turquoise:  #text color
```

And we add this to the component constructions

```
> mctools(ARRW);
> mctools(PT);
> HSNAME:=ARRW(tail=NMARTL,head=NMARHD,color=NMARCLR, arrowtype= 'SH',
  scalefactor=NMARSCLFCT),PT(NMCNTR,NMTXT,fontsize=NMPTSZE,txtclr=NMTXTC
  LR):
```

And we add HSNAME to the plots[display] command in the "assembly" instructions. The result to this point is below. Now its time to add offsets for the arrow and text.



13.2.2 intermediate stickman with arrow

```
> #name arrow and text parameters
> NMARHD:=SHLDR: #arrow head at shoulder
> NMARTL:=SHLDR+[2*HDRDS,0]:  # arrow length = head diameter
> NMARSCLFCT:=.05:  #.05 scaling factor to start
> NMARCLR:=orange:  #arrow color
> NMCNTR:=SHLDR+[2*HDRDS,0]:  # text starts at arrow tail
> NMTXT:="Joe":
> PTSZE:=20:  #start with 20 pt text
> NMTXTCLR:=turquoise:  #text color
> #Face parameters
> HDCNTR:=[0,4.5]:# head center
> HDRDS:=.5:# head radius
> HDCLR:=blue:  #head color
> #eyes
> LEYCNTR:=[-.2,4.7]:  # left eye center
> REYCNTR:=[.2,4.7]:# Right eye center
> EYCLR:=red:# eye color
> EYRDS:=.1:  #Eye radius
> #smile
> SMLCNTR:=[0,4.7]:  #smile center
> SMLRDS:=.4:  #smile radius
> SMLANGL1:=5*Pi/4:
> SMLANGL2:=7*Pi/4:
> SMLCLR:=green:# smile color
> #body parameters
> BDYTP:=[0,4]:  #body top
> BDYBDM:=[0,1]:  #body bottom
```

```
> BDYCLR:=magenta:  #body color
> #arms
> LHND:=[-2,3]:  #left hand
> RHND:=[2,3]:  #right hand
> SHLDR:=[0,3.5]:#shoulder(s)
> ARMCLR:=blue:  #arm color
> #legs
> LFT:=[-1,0]:  #left foot
> RFT:=[1,0]:  #right foot
> LGCLR:=red:  # foor color
> #construction of body components
> FACE:= PC(HDCNTR,HDRDS,thickness=3,linestyle=1,color=HDCLR):
> EYES:=PC(LEYCNTR,EYRDS,thickness=3,linestyle=1,color=EYCLR),PC(REYCNT
  R,EYRDS,thickness=3,linestyle=1,color=EYCLR):
> SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=3,linestyle=1,colo
  r=SMLCLR):
> BDY:=DL(BDYTP,BDYBDM,thickness=3,linestyle=1,color=BDYCLR):
> ARMS:=DL(LHND,SHLDR,thickness=3,linestyle=1,color=ARMCLR),DL(RHND,SHL
  DR,thickness=3,linestyle=1,color=ARMCLR):
> LGS:=DL(LFT,BDYBDM,thickness=3,linestyle=1,color=LGCLR),DL(RFT,BDYBDM
  ,thickness=3,linestyle=1,color=LGCLR):
> mctools(ARRW);
> HSNAME:=ARRW(tail=NMARTL,head=NMARHD,color=NMARCLR,arrowtype=‘SH‘,
  scalefactor=NMARSCLFCT),PT(NMCNTR,NMTXT,fontsize=PTSZE,color=NMTXTCLR)
  :
> #assemble figure
> STICKMAN:=plots[display]([HSNAME,LGS,ARMS,BDY,SML,EYES,FACE],scaling=
  constrained,axes=none):
> STICKMAN;
```

By inspection is appears that an offset of [1,0] for the arrow and [1.5,0] for the text will be about right The text color is a bit pale so we replace it with "navy" and the arrow is a bit thin so we up the scale factor to .07.

13.2.3 stickman with arrow

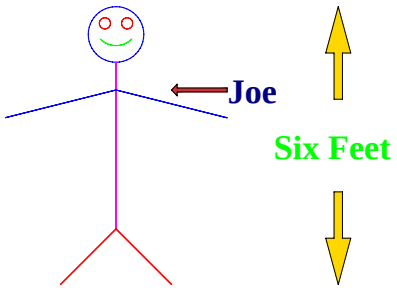
```
> #name arrow and text parameters
> ARROFFST:=[1,0]:#shift arrow 1 to right
  TXTOFFST:=[1.5,0]:#shift text 1.5 to right
> NMARHD:=SHLDR+ARROFFST: #arrow head at shoulder
> NMARTL:=SHLDR+[2*HDRDS,0]+ARROFFST: # arrow length = head diameter
> NMARSCLFCT:=.07:  #.05 scaling factor changed to .07
> NMARCLR:=orange:  #arrow color
> NMCNTR:=SHLDR+[2*HDRDS,0]+TXTOFFST: # text starts at arrow tail
> NMTXT:="Joe":
```

```
> PTSZE:=20:  #start with 20 pt text
> NMTXTCLR:=navy:  #text color
> #Face parameters
> HDCNTR:=[0,4.5]:# head center
> HDRDS:=.5:# head radius
> HDCLR:=blue:  #head color
> #eyes
> LEYCNTR:=[-.2,4.7]:  # left eye center
> REYCNTR:=[.2,4.7]:# Right eye center
> EYCLR:=red:# eye color
> EYRDS:=.1:  #Eye radius
> #smile
> SMLCNTR:=[0,4.7]:  #smile center
> SMLRDS:=.4:  #smile radius
> SMLANGL1:=5*Pi/4:
> SMLANGL2:=7*Pi/4:
> SMLCLR:=green:# smile color
> #body parameters
> BDYTP:=[0,4]:  #body top
> BDYBDM:=[0,1]:  #body bottom
> BDYCLR:=magenta:  #body color
> #arms
> LHND:=[-2,3]:  #left hand
> RHND:=[2,3]:  #right hand
> SHLDR:=[0,3.5]:#shoulder(s)
> ARMCLR:=blue:  #arm color
> #legs
> LFT:=[-1,0]:  #left foot
> RFT:=[1,0]:  #right foot
> LGCLR:=red:  # foor color
> #construction of body components
> FACE:= PC(HDCNTR,HDRDS,thickness=3,linestyle=1,color=HDCLR):
> EYES:=PC(LEYCNTR,EYRDS,thickness=3,linestyle=1,color=EYCLR),PC(REYCNT
  R,EYRDS,thickness=3,linestyle=1,color=EYCLR):
> SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=3,linestyle=1,colo
  r=SMLCLR):
> BDY:=DL(BDYTP,BDYBDM,thickness=3,linestyle=1,color=BDYCLR):
> ARMS:=DL(LHND,SHLDR,thickness=3,linestyle=1,color=ARMCLR),DL(RHND,SHL
  DR,thickness=3,linestyle=1,color=ARMCLR):
> LGS:=DL(LFT,BDYBDM,thickness=3,linestyle=1,color=LGCLR),DL(RFT,BDYBDM
  ,thickness=3,linestyle=1,color=LGCLR):
```

```
> mctools(PT);
> mctools(ARRW);
> HSNAME:=ARRW(tail=NMARTL,head=NMARHD,color=NMARCLR, arrowtype= ‘SH‘,
  scalefactor=NMARSCLFCT,doublearrowtxt=‘
  ‘),PT(NMCNTR,NMTXT,fontsize=PTSZE,color=NMTXTCLR):
> #assemble figure
> STICKMAN:=plots[display]([HSNAME,LGS,ARMS,BDY,SML,EYES,FACE],scaling=
  constrained,axes=none):
> STICKMAN;
```

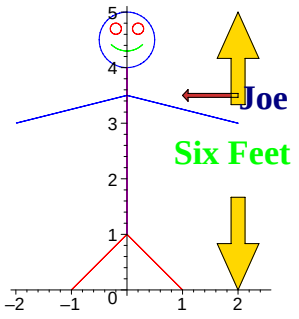
Example: Add a double arrow to the stickman indicating that the top of his head is six feet above the foor.

Solution: The diagram below meets the requirements. It can obviously be improved. We will make some improvements in the following exercises



The top of the stickman's head is his head radius plus the y-coordnate of the center of his head above the x-axis. This is HDRDS+HDCNTR[2]. His right hand is at RHND and we want the arrow a little to the right of that - the amount will be an offset we choose later. So we take:
HTARHD:=[RHND[1], HDRDS+HDCNTR[2]]: (height arrow head)
HTARTL:=[RHND[1], 0]: (height arrow tail)
We need a color for the arrow
HTARCLR:=gold:
We need a scaling factor for the arrow - we usually start with .05
HTARCLRFCT:=.05:
We will place the text at the midpoint of the arrow so:
HTTXTCNTR:=1/2*(HTARHD+HTARTL):
We are given the text
HTTXT:="Six Feet":
Finally, we need a font size and color for the text
HTPTSZE:=20: HTTXTCLR:=green:

Then all we need to do is construct the component
HSHGHT:=ARRW(HTARTL,HTARHD,HTARCLR, ‘DH‘, HTARSCLFCT),PT(HTTXTCNTR,HTTXT,HTF
Finally we will need to add HSHGT to the plots[display] assembly instructions.
Here is the intermediate result:

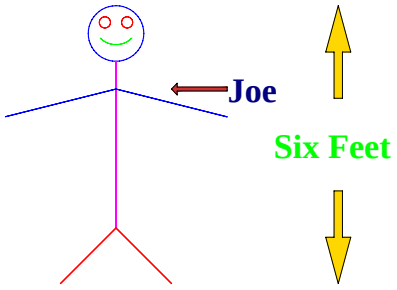


13.2.4 stickman with two arrows (intermediate)

```
> # height arrow and text
> HTARHD:=[ RHND[1], HDRDS+HDCNTR[2] ]:
> HTARTL:=[ RHND[1], 0 ]:
> HTARCLR:=gold:
> HTARCLRFCT:=.05:
> HTTXTCNTR:=1/2*(HTARHD+HTARTL):
> HTTXT:="Six Feet":
> HTPTSZE:=20:
> HTTXTCLR:=green:

> #name arrow and text parameters
> ARROFFST:=[1,0]:#shift arrow 1 to right
  TXTOFFST:=[1.5,0]:#shift text 1.5 to right
> NMARHD:=SHLDR+ARROFFST: #arrow head at shoulder
> NMARTL:=SHLDR+[2*HDRDS,0]+ARROFFST: # arrow length = head diameter
> NMARSCLFCT:=.07:  #.05 scaling factor changed to .07
> NMARCLR:=orange:  #arrow color
> NMCNTR:=SHLDR+[2*HDRDS,0]+TXTOFFST: # text starts at arrow tail
> NMTXT:="Joe":
> NMPTSZE:=20:  #start with 20 pt text
> NMTXTCLR:=navy:  #text color
> #Face parameters
> HDCNTR:=[0,4.5]:# head center
> HDRDS:=.5:# head radius
> HDCLR:=blue:  #head color
> #eyes
> LEYCNTR:=[-.2,4.7]:  # left eye center
> REYCNTR:=[.2,4.7]:# Right eye center
```

```
> EYCLR:=red:# eye color
> EYRDS:=.1:  #Eye radius
> #smile
> SMLCNTR:=[0,4.7]:  #smile center
> SMLRDS:=.4:  #smile radius
> SMLANGL1:=5*Pi/4:
> SMLANGL2:=7*Pi/4:
> SMLCLR:=green:# smile color
> #body parameters
> BDYTP:=[0,4]:  #body top
> BDYBDM:=[0,1]:  #body bottom
> BDYCLR:=magenta:  #body color
> #arms
> LHND:=[-2,3]:  #left hand
> RHND:=[2,3]:  #right hand
> SHLDR:=[0,3.5]:#shoulder(s)
> ARMCLR:=blue:  #arm color
> #legs
> LFT:=[-1,0]:  #left foot
> RFT:=[1,0]:  #right foot
> LGCLR:=red:  # foor color
> #construction of body components
> FACE:= PC(HDCNTR,HDRDS,thickness=3,linestyle=1,color=HDCLR):
> EYES:=PC(LEYCNTR,EYRDS,thickness=3,linestyle=1,color=EYCLR),PC(REYCNT
  R,EYRDS,thickness=3,linestyle=1,color=EYCLR):
> SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=3,linestyle=1,colo
  r=SMLCLR):
> BDY:=DL(BDYTP,BDYBDM,thickness=3,linestyle=1,color=BDYCLR):
> ARMS:=DL(LHND,SHLDR,thickness=3,linestyle=1,color=ARMCLR),DL(RHND,SHL
  DR,thickness=3,linestyle=1,color=ARMCLR):
> LGS:=DL(LFT,BDYBDM,thickness=3,linestyle=1,color=LGCLR),DL(RFT,BDYBDM
  ,thickness=3,linestyle=1,color=LGCLR):
> HSNAME:=ARRW(tail=NMARTL,head=NMARHD,color=NMARCLR, arrowtype= `SH`,
  scalefactor=NMARSCLFCT,doublearrowtxt=`
  `),PT(NMCNTR,NMTXT,fontsize=PTSZE,color=NMTXTCLR):

> HSHGHT:=ARRW(tail=HTARTL,head=HTARHD,color=HTARCLR,arrowtype=`DH`,
  scalefactor=HTARCLRFCT,doublearrowtxt=`
  `),PT(HTTXTCNTR,HTTXT,fontsize=HTPTSZE,color=HTTXTCLR):
> STICKMAN:=plots[display]([HSHGHT,HSNAME,LGS,ARMS,BDY,SML,EYES,FACE],s
  caling=constrained):
> STICKMAN;
```

We observe that the arrow needs to be shifted to the right. From the diagram a shift of [2,0] should work. The arrow is somewhat thick we reduce the scaling factor



13.2.5 code for stickman with two arrows

```
> # height arrow and text
> HTAROFFSET:=[2,0]:
> HTARHD:=[ RHND[1], HDRDS+HDCNTR[2] ] +HTAROFFSET:
> HTARTL:=[ RHND[1], 0 ] +HTAROFFSET :
> HTARCLR:=gold:
> HTARCLRFCT:=.03:  #reduced from .05
> HTTXTCNTR:=1/2*(HTARHD+HTARTL):
> HTTXT:="Six Feet":
> HTPTSZE:=20:
> HTTXTCLR:=green:


> #name arrow and text parameters
> ARROFFST:=[1,0]:#shift arrow 1 to right
  TXTOFFST:=[1.5,0]:#shift text 1.5 to right
> NMARHD:=SHLDR+ARROFFST: #arrow head at shoulder
> NMARTL:=SHLDR+[2*HDRDS,0]+ARROFFST: # arrow length = head diameter
> NMARSCLFCT:=.07:  #.05 scaling factor changed to .07
> NMARCLR:=orange:  #arrow color
> NMCNTR:=SHLDR+[2*HDRDS,0]+TXTOFFST: # text starts at arrow tail
> NMTXT:="Joe":
> NMPTSZE:=20:  #start with 20 pt text
> NMTXTCLR:=navy:  #text color
> #Face parameters
> HDCNTR:=[0,4.5]:# head center
> HDRDS:=.5:# head radius
```

```
> HDCLR:=blue:  #head color
> #eyes
> LEYCNTR:=[-.2,4.7]:  # left eye center
> REYCNTR:=[.2,4.7]:# Right eye center
> EYCLR:=red:# eye color
> EYRDS:=.1:  #Eye radius
> #smile
> SMLCNTR:=[0,4.7]:  #smile center
> SMLRDS:=.4:  #smile radius
> SMLANGL1:=5*Pi/4:
> SMLANGL2:=7*Pi/4:
> SMLCLR:=green:# smile color
> #body parameters
> BDYTP:=[0,4]:  #body top
> BDYBDM:=[0,1]:  #body bottom
> BDYCLR:=magenta:  #body color
> #arms
> LHND:=[-2,3]:  #left hand
> RHND:=[2,3]:  #right hand
> SHLDR:=[0,3.5]:#shoulder(s)
> ARMCLR:=blue:  #arm color
> #legs
> LFT:=[-1,0]:  #left foot
> RFT:=[1,0]:  #right foot
> LGCLR:=red:  # foor color
> mctools(PC);
> #construction of body components
> FACE:= PC(HDCNTR,HDRDS,thickness=3,linestyle=1,color=HDCLR):
> EYES:=PC(LEYCNTR,EYRDS,thickness=3,linestyle=1,color=EYCLR),PC(REYCNT
  R,EYRDS,thickness=3,linestyle=1,color=EYCLR):
> SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=3,linestyle=1,colo
  r=SMLCLR):
> BDY:=DL(BDYTP,BDYBDM,thickness=3,linestyle=1,color=BDYCLR):
> ARMS:=DL(LHND,SHLDR,thickness=3,linestyle=1,color=ARMCLR),DL(RHND,SHL
  DR,thickness=3,linestyle=1,color=ARMCLR):
> LGS:=DL(LFT,BDYBDM,thickness=3,linestyle=1,color=LGCLR),DL(RFT,BDYBDM
  ,thickness=3,linestyle=1,color=LGCLR):
> HSNAME:=ARRW(tail=NMARTL,head=NMARHD,color=NMARCLR, arrowtype= `SH`,
  doublearrowtxt="",shaftthickness=NMARSCLFCT),PT(NMCNTR,NMTXT,fontsize=
  NMPTSZE,color=NMTXTCLR):
> #assemble figure
```

```
> HSHGHT:=ARRW(tail=HTARTL,head=HTARHD,color=HTARCLR,arrowtype=‘DH‘,
  shaftthickness=HTARCLRFCT,doublearrowtxt=""),PT(HTTXTCNTR,HTTXT,fontsi
  ze=HTPTSZE,color=HTTXTCLR):
> STICKMAN:=plots[display]([HSHGHT,HSNAME,LGS,ARMS,BDY,SML,EYES,FACE],s
  caling=constrained,axes=none):
> STICKMAN;
```

13.3 Exercises

Exercise 1: Add a floor or ground for the stickman to stand on. Make the vertical arrow touch the ground.

Exercise 2: Move the arrow pointing to the stickman to the left and raise it to the level of the stickman's head.

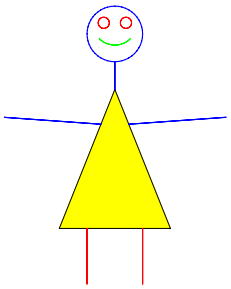
Exercise 3: Take the original stickman figure without the added arrows and text and add a right elbow bent up with the right hand "waving".

Exercise 4: Change the stickman to a stickwoman.

Exercise 5: Adapt the stickman's head and make a yellow "happy face", placing the text "Have a Nice Day" beneath it.

13.4 Stickwoman and other drawings using plots and plottools

One of the exercises in the previous section was to make a **stickwoman** . One variant on that characture is the following:



13.4.1 stickwoman

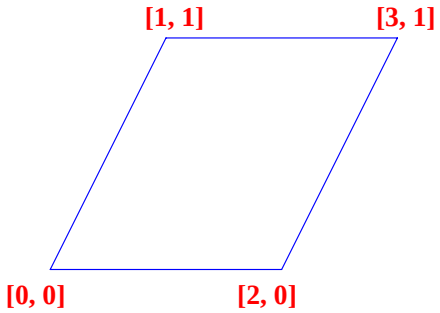
```
> #Face parameters
> HDCNTR:=[0,4.5]:# head center
> HDRDS:=.5:# head radius
> HDCLR:=blue:  #head color
> #eyes
> LEYCNTR:=[-.2,4.7]:  # left eye center
> REYCNTR:=[.2,4.7]:# Right eye center
```

```
> EYCLR:=red:# eye color
> EYRDS:=.1:  #Eye radius
> #smile
> SMLCNTR:=[0,4.7]:  #smile center
> SMLRDS:=.4:  #smile radius
> SMLANGL1:=5*Pi/4:
> SMLANGL2:=7*Pi/4:
> SMLCLR:=green:# smile color
> #body parameters
> #BDYTP:=[0,4]:  #body top
> #BDYBDM:=[0,1]:  #body bottom
> #BDYCLR:=magenta:  #body color
  #neck
> NCKTOP:=[0,4]:
> NCKBDM:=[0,3.5]:
> #stickwoman dress

> DRSTOP:=[0,3.5]:    #top of dress for stickwoman
> LDRSBDM:=[-1,1]:
  RDRSBDM:=[1,1]:



> #arms
> LHND:=[-2,3]:  #left hand
> RHND:=[2,3]:  #right hand
> SHLDR:=[0,3.5]:#shoulder(s)
> ARMCLR:=blue:  #arm color
> #legs
> LFTLEGTOP:=(3/4*LDRSBDM+1/4*RDRSBDM):

> RTLEGTOP:=(3/4*RDRSBDM+1/4*LDRSBDM):
> LFFT:=[LFTLEGTOP[1],0]:  #left foot
> RTFT:=[RTLEGTOP[1],0]:#right foot

> LGCLR:=red:  # foor color
> #construction of body components
> FACE:= PC(HDCNTR,HDRDS,thickness=3,linestyle=1,color=HDCLR):
> EYES:=PC(LEYCNTR,EYRDS,thickness=3,linestyle=1,color=EYCLR),PC(REYCNT
  R,EYRDS,thickness=3,linestyle=1,color=EYCLR):
> SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=3,linestyle=1,colo
  r=SMLCLR):
```

```
> #BDY:=DL(BDYTP,BDYBDM,3,1,BDYCLR):
> DRS:=plots[polygonplot]([DRSTOP,LDRSBDM,RDRSBDM],color=yellow):
> ARMS:=DL(LHND,3/4*DRSTOP+1/4*LDRSBDM,thickness=3,linestyle=1,color=AR
  MCLR),DL(RHND,3/4*DRSTOP+1/4*RDRSBDM,thickness=3,linestyle=1,color=ARM
  CLR):
> LGS:=DL(LFTLEGTOP,LFFT,thickness=3,linestyle=1,color=LGCLR),DL(RTLEGT
  OP,RTFT,thickness=3,linestyle=1,color=LGCLR):
> NCK:=DL(NCKTOP,NCKBDM,thickness=3,linestyle=1,color=blue):
  #assemble figure
> STICKWOMAN:=plots[display]([NCK,LGS,ARMS,DRS,SML,EYES,FACE],scaling=c
  onstrained,axes=none):
> STICKWOMAN;
```

Obviously she can be improved but first we want to discuss her dress which is a colored triangle - or more generally a colored polygon. The Maple command which creates such figures is **plots[polygonplot]**

A polygon is described in Maple by giving a list of its vertices as one treaverses its perimeter (in either order. Thus we can describe the parallelogram in the diagram below by: [[0,0],[2,0],[3,1],[1,1]] . This is not unique, there are seven more equally valid descriptions such as [[3,1],[2,0],[0,0],[1,1]]



To plot this polygon with plots[polygonplot]

```
> pgrm:= [[0,0],[2,0],[3,1],[1,1]] :
> PGRM:=plots[polygonplot](pgrm):
> PGRM;
```

The default color is white however we can specify the color just as with other commands and can suppress the axes just as before.

```
> PGRM:=plots[polygonplot](pgrm,color=blue,axes=none):
> PGRM;
```

If we want a transparent polygon with the edge of specified color, style, and thickness we elect style=line and deterimine the other paramenters as before:

```
> PGRM:=plots[polygonplot](pgrm,style=line,thickness=3,linestyle=3,colo
  r=blue,axes=none):
> PGRM;
```

In a previous exercise we made a "Happy Face" using circles and circular arcs. The face on such a figure is usually painted on a yellow disk rather than a white one. The **plottools[disk]** command places disks of a given radius and color at any point.

```
> hdisk:=plottools[disk]([0,0],1,color=yellow):
> plots[display](hdisk);
```

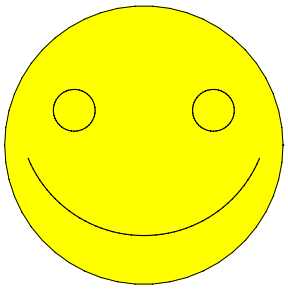
We omit the axes and constrain the scaling in plots[display] commans in the usual manner

```
> hdisk:=plottools[disk]([0,0],1,color=yellow):
> plots[display](hdisk,scaling=constrained, axes=none);
```

Now we can use polygons and disks as components of our diagrams

Example: Make a happy face on a yellow disk

Solution: The figure below is an example



The figure consists of the yellow disk, the smile and the eyes so we build these three components using plottoold[disk] for the yellow disk and PC and PA for the eyes.

```
> #face
> FCECNTR:=[0,0]:# face center
> FCERDS:=1:#face radius
> LEYECNTR:=FCECNTR+FCERDS*[-.5,.25]:#Left eye center
> REYECNTR:=FCECNTR+FCERDS*[+.5,.25]:#right eye center
> EYERDS:=.15*FCERDS:#eye radius
> FCECLR:=yellow:# face color
> EYECLR:=black:#eye color
> #smile (stolen from stickman)
> SMLCNTR:=FCECNTR+FCERDS*[0,.25]:  #smile center
> SMLRDS:=.9*FCERDS: #smile radius
> SMLANGL1:=9*Pi/8:
> SMLANGL2:=15*Pi/8:
> SMLCLR:=black:# smile color
> #components
> DSK:=plottools[disk](FCECNTR,FCERDS,color=FCECLR):
> EYS:=PC(REYECNTR,EYERDS,thickness=3,linestyle=1,color=EYECLR),PC(LEYE
  CNTR,EYERDS,thickness=3,linestyle=1,color=EYECLR):
> SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=3,linestyle=1,colo
  r=SMLCLR):
  #assembly
```

```
> SMFACE:=plots[display]([EYS,SML,DSK],scaling=constrained,axes=none):
  SMFACE;
```

It is often convenient to place diagrams on a background of a color of our chosing. For instance if we wanted to place the smiley face on a green background we can use plots[polygonplot] to make a green rectangle and them place the face on the rectangle with plots[display]

```
> bkgrnd:=[[-2,-2],[2,-2],[2,2],[-2,2]]:
> BKGRND:=plots[polygonplot](bkgrnd,color=green):
> plots[display]([SMFACE,BKGRND],axes=none,scaling=constrained);
```

Note: when we assemble a diagram by having plots[display] act on a list of figures it places the individual components in the list is the opposite order in which they appear in the list. That is plots[display]([A,B,C]) puts A on top of B on top of C. Failure to keep track of this can lead to some unexpected results. For instance if we reverse the order of the face and background in the previous example

```
> plots[display]([BKGRND,SMFACE],axes=none,scaling=constrained);
```

Example: Illustrating a standard calculus problem using plots[polygonplot]

A problem appearing in most calculus books asks the maximum possible volume of an open-top box made from a rectangular piece of cardboard by cutting squares of material from each corner folding up the resulting four edge panels. Create a diagram to illustrate this problem**.**

Solution: The following is the basic diagram

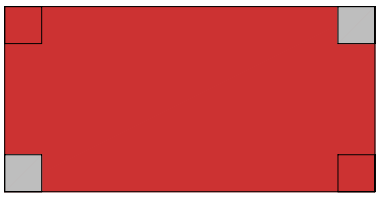
To systematically construct the diagram we proceed to identify parameters that define the figure. There is a large rectangle with four smaller squares. We elect to treat these as separate components although it would be equally reasonable to treat the four small squares collectively as one component. To start we elect to make the rectangle orange, the small squares grey, and the edge of the small squares 20% of the width of the rectangle.

```
> #diagram parameters
> WDTH:=10:# width of the rectangle
> LNGTH:=20:# length of the rectangle
> SQSIDE:=.2*WDTH:#edge of small square =.2*length
  LLRECT:=[0,0]:# lower left corner of rectangle
> RECTCLR:=orange:#color of rectangle
> SQCLR:=grey:#color of the squares


> #corners of the rectangle
> A:=LLRECT:
```

```
> B:=A+[LNGTH,0]:
> C:=A+[LNGTH,WDTH]:
> DD:=A+[0,WDTH];

> # make the rectangle
> rect:=[A,B,C,DD]:
> #make the squares:
> llsq:=[A+[0,0],A+[SQSIDE,0],A+[SQSIDE,SQSIDE],A+[0,SQSIDE] ]:#lower
  left
> lrsq:=[B+[0,0],B+[-SQSIDE,0],B+[-SQSIDE,SQSIDE],B+[0,SQSIDE] ]:#lower
  right
> ursq:=[C+[0,0],C+[-SQSIDE,0],C+[-SQSIDE,-SQSIDE],C+[0,-SQSIDE]
  ]:#upper right
> ulsq:=[DD+[0,0],DD+[SQSIDE,0],DD+[SQSIDE,-SQSIDE],DD+[0,-SQSIDE]
  ]:#upper left
> #create the plots
> RECT:=plots[polygonplot](rect,color=RECTCLR):
> LLSQ:=plots[polygonplot](llsq,color=SQCLR):
> LRSQ:=plots[polygonplot](lrsq,color=SQCLR):
> URSQ:=plots[polygonplot](ursq,color=SQCLR):
> ULSQ:=plots[polygonplot](ulsq,color=SQCLR):

> #assemble the figure
> DGRM:=plots[display]([LLSQ,URSQ,LRSQ,ULSQ,RECT],scaling=constrained,a
  xes=none):
> DGRM;
```

As we have noted before the order in which one assembles a diagram can be important. In the previous example the assembly instruction

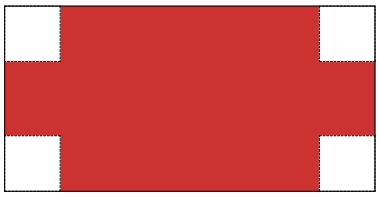
DGRM:=plots[display]([LLSQ,URSQ,LRSQ,ULSQ,RECT],scaling=constrained,ax es=none) lays the orange rectangle, then the upper left square is placed on top of that figure, then the lower right, ... ect. The effect of DGRM:=plots[display]([LLSQ,URSQ,RECT,LRSQ,ULSQ],scaling=constrained,ax es=none) would be to lay down the upper left and lower right squares before laying down the rectangle with the resulting diagram the following:



13.4.2 switched assembly

```
> DGRM_switched:=plots[display]([LLSQ,URSQ,RECT,LRSQ,ULSQ],scaling=cons
  trained,axes=none):
> DGRM_switched;
```

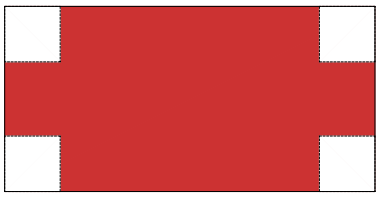
Although this is not an unreasonable diagram one may want to improve it. For instance one may wish to more clearly indicate that the squares are to be removed by having them white with dashed edges. To make this more visible we may want to increase the size of the small squares.



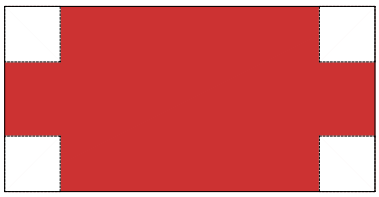
To increase the side length of the square we would edit the "SQSIDE" declaration
SQSIDE:=.3*WDTH:
To change the colors of the squares we edit the SQCLR declaration
SQCLR:=white:
To make the edges of the squares dashed we edit their plot declarations. For instance
LLSQ:=plots[polygonplot](llsq,color=SQCLR,linestyle=3):
Note we have added the option "linestyle=3" which specifies that the square border will be dashed - exactly as with DL instructions.
These changes produce:

13.4.3 switch

Although this is not an unreasonable diagram one may want to improve it. For instance one may wish to more clearly indicate that the squares are to be removed by having them white with dashed edges. To make this more visible we may want to increase the size of the small squares.



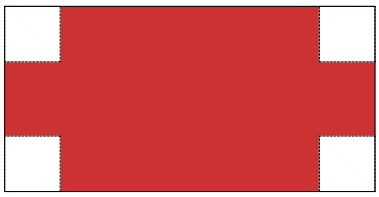
To increase the side length of the square we would edit the "SQSIDE" declaration
SQSIDE:=.3*WDTH:
To change the colors of the squares we edit the SQCLR declaration
SQCLR:=white:
To make the edges of the squares dashed we edit their plot declarations. For instance
LLSQ:=plots[polygonplot](llsq,color=SQCLR,linestyle=3):
Note we have added the option "linestyle=3" which specifies that the square border will be dashed - exactly as with DL instructions.
These changes produce:



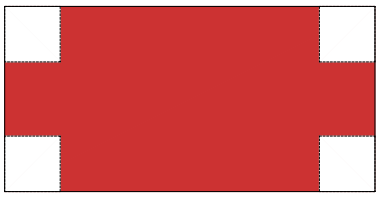
ed assembly

```
> DGRM_switched:=plots[display]([LLSQ,URSQ,RECT,LRSQ,ULSQ],scaling=cons
  trained,axes=none):
> DGRM_switched;
```

Although this is not an unreasonable diagram one may want to improve it. For instance one may wish to more clearly indicate that the squares are to be removed by having them white with dashed edges. To make this more visible we may want to increase the size of the small squares.



To increase the side length of the square we would edit the "SQSIDE" declaration
SQSIDE:=.3*WDTH:
To change the colors of the squares we edit the SQCLR declaration
SQCLR:=white:
To make the edges of the squares dashed we edit their plot declarations. For instance
LLSQ:=plots[polygonplot](llsq,color=SQCLR,linestyle=3):
Note we have added the option "linestyle=3" which specifies that the square border will be dashed - exactly as with DL instructions.
These changes produce:



13.4.4 rectangle, white, dashed inner squares

```
> #diagram parameters
> WDTH:=10:# width of the rectangle
> LNGTH:=20:# length of the rectangle
> SQSIDE:=.3*WDTH:#edge of small square =.3*length
  LLRECT:=[0,0]:# lower left corner of rectangle
> RECTCLR:=orange:#color of rectangle
> SQCLR:=white:#color of the squares

> #corners of the rectangle
> A:=LLRECT:
> B:=A+[LNGTH,0]:
> C:=A+[LNGTH,WDTH]:
```

```
> DD:=A+[0,WDTH];

> # make the rectangle
> rect:=[A,B,C,DD]:
> #make the squares:
> llsq:=[A+[0,0],A+[SQSIDE,0],A+[SQSIDE,SQSIDE],A+[0,SQSIDE] ]:#lower
  left
> lrsq:=[B+[0,0],B+[-SQSIDE,0],B+[-SQSIDE,SQSIDE],B+[0,SQSIDE] ]:#lower
  right
> ursq:=[C+[0,0],C+[-SQSIDE,0],C+[-SQSIDE,-SQSIDE],C+[0,-SQSIDE]
  ]:#upper right
> ulsq:=[DD+[0,0],DD+[SQSIDE,0],DD+[SQSIDE,-SQSIDE],DD+[0,-SQSIDE]
  ]:#upper left
> #create the plots




> RECT:=plots[polygonplot](rect,color=RECTCLR):
> LLSQ:=plots[polygonplot](llsq,color=SQCLR,linestyle=3):
> LRSQ:=plots[polygonplot](lrsq,color=SQCLR,linestyle=3):
> URSQ:=plots[polygonplot](ursq,color=SQCLR,linestyle=3):
> ULSQ:=plots[polygonplot](ulsq,color=SQCLR,linestyle=3):

> #assemble the figure
> DGRM:=plots[display]([LLSQ,URSQ,LRSQ,ULSQ,RECT],scaling=constrained,a
  xes=none):
> DGRM;
```

It appears that the inner edges of the cutouts are dashed in this picture but not the outer. Actually the perimeters of each of the squares is dashed but the outer portions are laying on top of the perimeter of the rectangle. One fix for this is to remover the border of the rectangle. This is done with the "patchnogrid" option. That is if we change the instructions that produced the previous diagram by changing the RECT plot declaration to

RECT:=plots[polygonplot](rect,color=RECTCLR,style=patchnogrid): we get

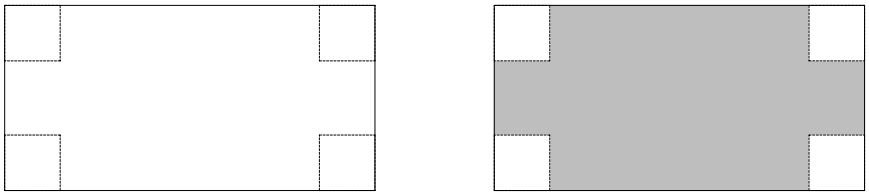
13.4.5 rectangle, white, dashed squares

```
> #diagram parameters
> WDTH:=10:# width of the rectangle
> LNGTH:=20:# length of the rectangle
> SQSIDE:=.3*WDTH:#edge of small square =.3*length
  LLRECT:=[0,0]:# lower left corner of rectangle
> RECTCLR:=orange:#color of rectangle
> SQCLR:=white:#color of the squares

> #corners of the rectangle
> A:=LLRECT:
> B:=A+[LNGTH,0]:
> C:=A+[LNGTH,WDTH]:
> DD:=A+[0,WDTH];

> # make the rectangle
> rect:=[A,B,C,DD]:
> #make the squares:
> llsq:=[A+[0,0],A+[SQSIDE,0],A+[SQSIDE,SQSIDE],A+[0,SQSIDE] ]:#lower
  left
> lrsq:=[B+[0,0],B+[-SQSIDE,0],B+[-SQSIDE,SQSIDE],B+[0,SQSIDE] ]:#lower
  right
> ursq:=[C+[0,0],C+[-SQSIDE,0],C+[-SQSIDE,-SQSIDE],C+[0,-SQSIDE]
  ]:#upper right
> ulsq:=[DD+[0,0],DD+[SQSIDE,0],DD+[SQSIDE,-SQSIDE],DD+[0,-SQSIDE]
  ]:#upper left
> #create the plots
```

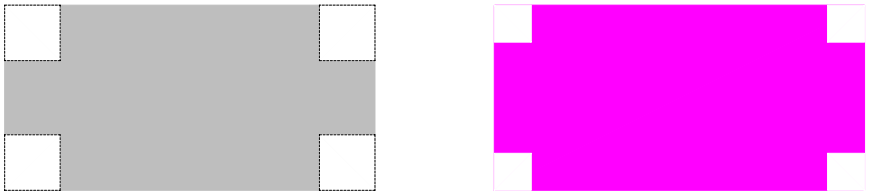
```
> RECT:=plots[polygonplot](rect,color=RECTCLR,style=patchnogrid):
> LLSQ:=plots[polygonplot](llsq,color=SQCLR,linestyle=3):
> LRSQ:=plots[polygonplot](lrsq,color=SQCLR,linestyle=3):
> URSQ:=plots[polygonplot](ursq,color=SQCLR,linestyle=3):
> ULSQ:=plots[polygonplot](ulsq,color=SQCLR,linestyle=3):

> #assemble the figure
> DGRM:=plots[display]([LLSQ,URSQ,LRSQ,ULSQ,RECT],scaling=constrained,a
  xes=none):
> DGRM;
```

13.5 Exercises:

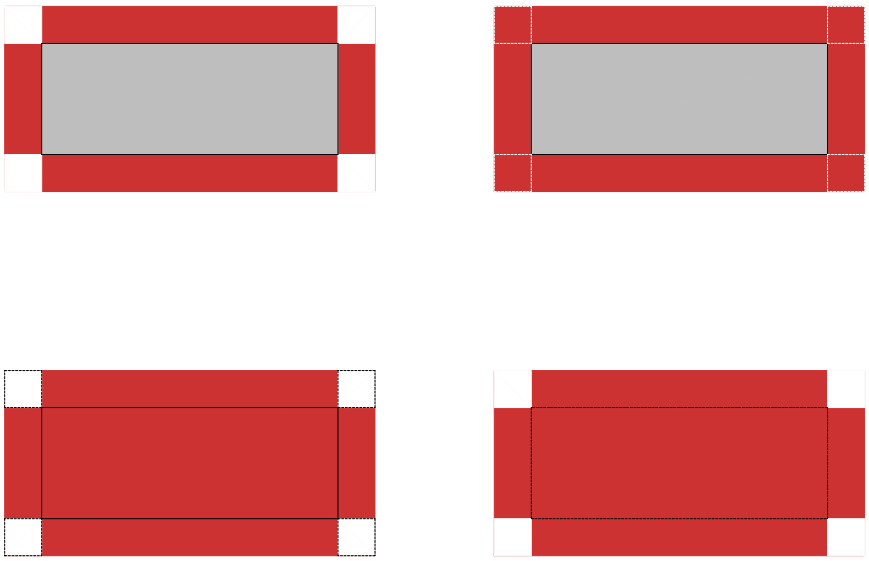
Exercise 1: Modify the instructions for the "cutout" diagram to produce:



Exercise 2: Modify the instructions for the "cutout" diagram to produce:

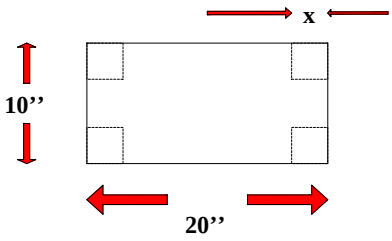


Exercise 3: Modify instructions for the "cutout" diagram to produce the following.



It will be necessary to add the new lines using either "DL" or plots[polygonplot]. The latter is preferable

Exercise 4: Modify the "cutout" diagram to produce the following diagram:



13.6 Adding Titles to Diagrams

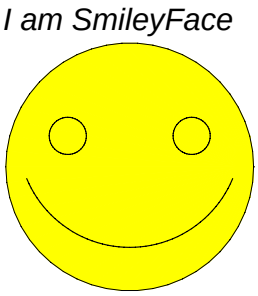
Two of the options to the plots[display] command are title and titlefont. With them we can give diagrams titles in font sizes of our chosing. For instance we can add the title "I am SmileyFace" to the smileyface diagram by editing the plots[display] command to

SMFACE:=plots[display]([EYS,SML,DSK],scaling=constrained,axes=none, title='IamSmileyFace',titlefont=[H

Unfortunately it is not yet possible to add nicely formatted mathematics to a title.

13.6.1 smileyfacewithtitle

```
> #face
> FCECNTR:=[0,0]:# face center
> FCERDS:=1:#face radius
> LEYECNTR:=FCECNTR+FCERDS*[-.5,.25]:#Left eye center
> REYECNTR:=FCECNTR+FCERDS*[+.5,.25]:#right eye center
> EYERDS:=.15*FCERDS:#eye radius
> FCECLR:=yellow:# face color
> EYECLR:=black:#eye color
> #smile (stolen from stickman)
> SMLCNTR:=FCECNTR+FCERDS*[0,.25]:  #smile center
> SMLRDS:=.9*FCERDS: #smile radius
> SMLANGL1:=9*Pi/8:
> SMLANGL2:=15*Pi/8:
> SMLCLR:=black:# smile color
> #components
> DSK:=plottools[disk](FCECNTR,FCERDS,color=FCECLR):
> EYS:=PC(REYECNTR,EYERDS,thickness=3,linestyle=1,color=EYECLR),PC(LEYE
  CNTR,EYERDS,thickness=3,linestyle=1,color=EYECLR):
> SML:=PA(SMLCNTR,SMLRDS,SMLANGL1,SMLANGL2,thickness=3,linestyle=1,colo
  r=SMLCLR):
> #assembly
> SMFACE:=plots[display]([EYS,SML,DSK],scaling=constrained,axes=none,ti
  tle=`I am SmileyFace`,titlefont=[HELVETICA,OBLIQUE,18]):
> SMFACE;
```



13.7 Diagrams including graphs of functions

Functions are of course the most fundamental object of study in calculus so many calculus problems invlove the graphs of functions. The basic Maple tool for creating graphs is the plot command. Once we realize that Maple plots are simply plot structures it is obvious how to add text, arrows, etc. to functions to create informative illustrations.

Example: On the same diagram plot f(x) = x^3-x and its derivative on the interval [-2,2]. Label each of the graphs.

Solution: The diagram below is one example.

First we graph the two functions. Although we could make them individually and merge them with plots[display]. Maple already graphs sets or lists of functions.

```
> F:=x^3-x:
> dF:=diff(F,x):
> plt1:=plot([F,dF],x=-2..2):  plt1;
```

The linestyle and thickness options are exactly as before. In addition we can control the colors of the graphs the color option and can control the size of the numbers on the axes with the axesfont option. For instance to make F blue, dF red, and the axes numbers 16 point, bold we can

```
> F:=x^3-x:
> dF:=diff(F,x):
> plt2:=plot([F,dF],x=-2..2,color=[blue,red],
  thickness=3,linestyle=[3,1], axesfont=[TIMES,BOLD,16]):
  plt2;
```

The x-axis is labeled with a faint "x" in the diagram. We can control the axes labels with the labels and labelfont options

```
> F:=x^3-x:
> dF:=diff(F,x):
> plt3:=plot([F,dF],x=-2..2,color=[blue,red],
  thickness=3,linestyle=[3,1],
  axesfont=[TIMES,BOLD,16],labels=[`x-axis`,`y-axis`],labelfont=[TIMES,B
> OLD,18]):
  plt3;
```

It is very common for us to want only a few special numbers to be labeled on the axes. This is controlled by the xtickmarks and ytickmarks options

```
> F:=x^3-x:
> dF:=diff(F,x):
> plt5:=plot([F,dF],x=-2..2,color=[blue,red],
  thickness=3,linestyle=[3,1],
  axesfont=[TIMES,BOLD,16],labels=["x-axis","y-axis"],labelfont=[TIMES,B
> OLD,18],xtickmarks=[-1,1],ytickmarks=[1,3]):
  plt5;
```

13.8 Viewing a portion of a plot

The view plot option permits us to "focus" on a rectangular portion of a graph. Its syntax is view = [a..b, c..d] which restricts the displayed portion of the graph to that within the rectangle $a \leq x \leq b$, $c \leq x \leq d$ For instance suppose we want to view only the portion of the previous plot in the window $-1 \leq x \leq 0$, $-2 \leq x \leq 3$. all we have to do is add a view option

```
> F:=x^3-x:
> dF:=diff(F,x):
> plt7:=plot([F,dF],x=-2..2,color=[blue,red],
  thickness=3,linestyle=[3,1],
  axesfont=[TIMES,BOLD,16],labels=["x-axis","y-axis"],labelfont=[TIMES,B
> OLD,18],xtickmarks=[-1,1],ytickmarks=[1,3],legend=["the function
  f(x)","the derivative of f(x)"],title="Graphs of F and F'",
  titlefont=[COURIER,BOLD,18],view=[-1..0,-2..3]):
  plt7;
```

13.9 Placing a Graph on Graph Paper

One frequently finds it convenient to place a graph on "graphpaper" rather than simply have it located relative to coordinate axes with tickmarks. The command GP (for "graph paper") is useful for doing this. Its syntax is:

GP(llcorner,width, length, resolution, font_ where

llcorner = the point [a,b] which is the lower left corner of the graphpaper

width = the width of the graphpaper

length = the length of the graphpaper

resolution = number of grid squares covering the inter [0,1] (1,2,3,4) are the most common values.

font = font in which tickmarks are printed

Notes:

A fontsize of 1 gives no tickmarks. This is used very frequently

GP is primitive - it puts tickmarks only at integer values

```
> mctools(GP);
> GPAP:=GP(llcorner=[-2,-2],width=4,height=4,resolution=2,tickmarkfont=
  10,axescolor=orange):
  GPAP;
```

Suppose we want to look at the function f(x) = x^2+x plotted on the graphpaper we just produced. We might begin simply by merging the graph of f(x) on the interval [-2,2] with the graphpaper using plots[display].

```
> F:=x^2+x:
> plt:=plot(F,x=-2..2,color=[blue,red], thickness=3,linestyle=[3,1],
  title="Graph of F(x)", titlefont=[TIMES,BOLD,20]):
> plt2:=plots[display]([GPAP,plt]):
  plt2;
```

This happens quite often, the graph extends off the graphpaper. We can avoid this (if desired) by setting the view be the portion of the merged plots to be that corresponding to the dimensions of the graphpaper

```
> F:=x^2+x:
> plt:=plot(F,x=-2..2,color=[blue,red], thickness=3,linestyle=[3,1],
  title="Graph of F(x)", titlefont=[TIMES,BOLD,20]):
> plt3:=plots[display]([GPAP,plt2],view=[-2..2,-2..2]):
> plt3;
```

13.10 Piecewise defined functions

One of the most important constructions in the study of functions is the **piecewise defined** function. The first one most people run into is the absolute value. Maple expresses piecewise defined functions a little differently from the way it is presented in most texts. For instance the absolute value is described as:

$$|x| = \begin{cases} -x & x \leq 0 \\ x & 0 \leq x \end{cases}$$

The conventional description might be written as

$$|x| = \begin{cases} -x & \textit{if} \quad x \leq 0 \\ x & \textit{otherwise} \end{cases}$$

In Maple only the inequalities and corresponding values are given, *in order*. The convention is that the value of the function is that given by the first (from top to bottom) one of the conditions that is satisfied. Thus if we set

$$f(x) = \begin{cases} 2 & x \leq -1 \\ 3 & x < 0 \\ -1 & x < 7 \end{cases}$$

```
> F:=x-> piecewise(x<=-1,2,x<0,3,x< 7,-1);
  `f(x)` =F(x);
```

Without the convention the value of f(-5) would not be properly defined since -5 satisfies all of the conditions. However the convention uniquely determines f(-5) to be 2 since -5 satisfties the first inequality. Similary f($-\frac{1}{2}$) = 3 since the second inequality is the first satisfied by $-\frac{1}{2}$. Note that f(7) is not defined since 7 satisfies none of the inequalities.

The form of a general piecewise function h is:

$$h(x) = \begin{cases} p_1(x) & x \leq a \\ p_2(x) & x \leq b \\ \ldots & x \leq \ldots \\ p_n(x) & x \leq z \\ \mathrm{p}(x) & \textit{otherwise} \end{cases}$$

Where p_1, ..., p_n, p are functions. Note that any of the " $\leq$ " symbols could be " $<$ " ,

To describe a piecewise defined function to Maple we use the "piecewise" command. The function f(x) described above is defined by:

f :=x -> piecewise(x<=-1,2,x<0,3,x< 7,-1);

```
> h:=x->piecewise(x<=a, p[1](x), x<=b, p[2](x) ,x<=‘.  .  .‘, ‘...‘,x<=z,
  p[n](x),p(x));
```

If a pair of values is given at the left of the expression, e.g.

f :=x -> piecewise(x<=-1,2,x<0,3,x< 7,-1,sin(x)); formatting as $f(x) = \begin{cases} 2 & x \leq -1 \\ 3 & x < 0 \\ -1 & x < 7 \\ \sin(x) & otherwise \end{cases}$

the leftmost value is the default - that is it gives the value of f(x) for any value of x not described by the preceeding inequalities.

```
> f :=x -> piecewise(x<=-1,2,x<0,3,x< 7,-1,sin(x));
> ‘f(x)‘=f(x);
```

This means that there is more than one way to describe many piecewise functions.

The absolute value function is can be described by:

f:=x-> piecewise(x<=0, -x, x<infinity, x) formatting as $f(x) = \begin{cases} -x & x \leq 0 \\ x & x < \infty \end{cases}$

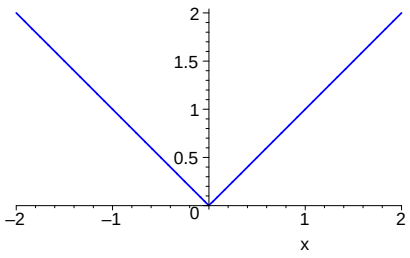
or

f:=x-> piecewise(x<=0, -x,x); formatting as $f(x) = \begin{cases} -x & x \leq 0 \\ x & otherwise \end{cases}$

```
> f:=x-> piecewise( x<=0, -x,x<infinity, x);
> ’f(x)’=f(x);
> f:=x-> piecewise( x<=0, -x,x);‘f(x)‘=f(x);
```

Piecewise functions are plotted using the plot command in exactly the same fashion as those described by expressions.

f:=x->piecewise(x<=0,-x,x): absplot:=plot(f(x),x=-2..2, thickness=3, linestyle=2,color=blue, scaline=constrained): absplot;



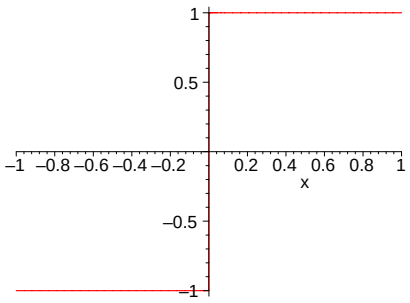

```
> f:=piecewise(x<=0,-x,x):
  absplot:=plot(f(x),x=-2..2, thickness=4, linestyle=1,color=blue,
  scaling=constrained):
  absplot;
```

13.11 Discontinuous Functions:

Piecewise descriptions are the most convenient way to define **discontinuous functions** . For instance to describe the function which takes on the value -1 for x <=0 and 1 for x >0 we can write:

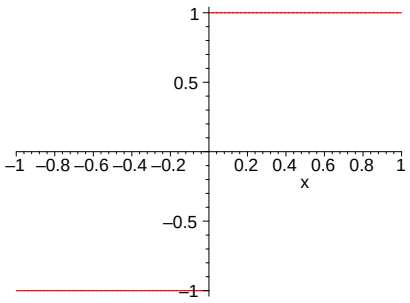
f=:=x->piecewise(x<=-1,1): plt:=plot(f(x),x=-1..1): plt;

However when we plot f(x) we get someting unexpected:



There is nothing wrong with the definition of f(x). The problem is with Maple's plot routine which assumes that all functions are continuous and does its best to plot them as such. The remedy for this is to tell Maple that the function in question may be discontinuous with the **discont** option to to the plot command.

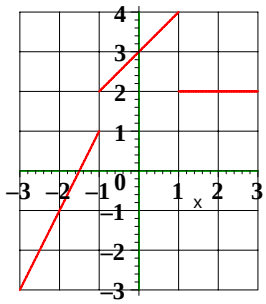
f=:=x->piecewise(x<=-1,1): plt:=plot(f(x),x=-1..1, discont=true): plt;




```
> f:=x->piecewise(x<=0,-1,1):
> plt:=plot(f(x),x=-1..1,thickness=3,discont=true):  plt;
```

Example:

Piecewise linear functions have graphs which are comprised of a collection of segments of straight lines. They are by far the most common and most important piecewise defined functions. When such a function is given analytically as a piecewise-defined function the individual component functions are always linear (i.e. of the form $p_n(x) = a_n\,x + b_n$). Give an analytic description of the function whose graph is sketched below.



The graph consists of three line segments. The leftmost passes through the points (-3,-3) and (-1,1) and is thus part of the line having equation $y = 2\,x + 3$. The second passes through the points (-1,2) and (1,4) so it is part of the line having equation y = x+3. The third is obviously part of the line having equation y = 2.

One piecewise description of this graph is

f:=x->piecewise(x<=-1,2*x+3,x <=1,x+3,x<=3,2);

We say that this is "one" possibility because from the figure we can't really tell what happens at the end points. That is

f:=x->piecewise(x< -1,2*x+3,x< 1,x+3,x<=3,2);

is equally valid.

```
> f:=x->piecewise(x<=-1,2*x+3,x<=1,x+3,x<=3,2);
> gp:=GP(llcorner=[-3,-3],width=6,height=7,resolution=1,tickmarkfont=14
  ,color=green):
> plt:=plot(f(x),x=-3..3,discont=true,thickness=5,axesfont=[TIMES,BOLD,
  12]):
  plots[display]([plt,gp],scaling=constrained);
```

13.11.1 End point Behavior

The reader remembering or teaching calculus will want to be able to describe the **end point behavior** in functions like the above. Traditionally, a solid disk at an end point on a graph indicates that the end point is indeed part of the graph while an open circle indicates that the end point is absent. We can easily reproduce this in our diagrams for plots that are constrained. That is when the scales on the x-axis and y-axis are identical. Then we can use PP and PC ("put point" and "put circle") tools.

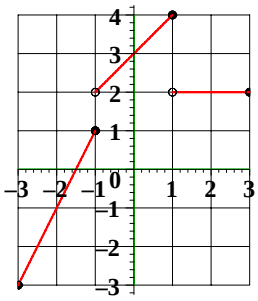
PP has the syntax: **PP(center, radius, color)** while PC has the syntax: **PC(center, radius, thickness, linestyle,color)**

Here is the code which produced the above diagram.

f:=x->piecewise(x<=-1,2*x+3,x <=1,x+3,x<=3,2);
gp:=GP([-3,-3],6,7,1,14,green):
plt:=plot(f(x),x=-3..3,discont=true,thickness=5,axesfont=[TIMES,BOLD,1 2]):
plots[display]([plt,gp],scaling=constrained);

We modifty it by adding points and circles of radius .1.

S:=PP([-3,-3],.1,black),PP([-1,1],.1,black),PP([1,4],.1,black),PP([3,2],.1,black):
CIRCS:=PC([-1,2],.1,1,1,black),PC([1,2],.1,1,1,black): f:=x->piecewise(x<=-1,2*x+3,x <=1,x+3,x<=3,2);
gp:=GP([-3,-3],6,7,1,14,green):
plt:=plot(f(x),x=-3..3,discont=true,thickness=5,axesfont=[TIMES,BOLD,1 2]): dgm:= plots[display]([CIRCS,
dgm:

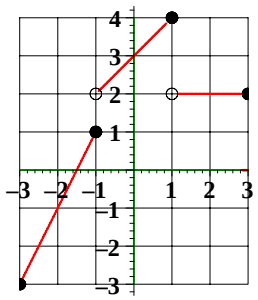


This isn't too bad but the lines clearly overlap the points and circles. This is easily remedied by adjusting the piecewise graph to shrink the lines by the radius of the circles and points. This means we have to convince Maple to skip some portions of the graph. One simple way to do this is to declare the value of the function to be complex on such invervals. Recall that I is the complex number $\sqrt{-1}$. We assign the radius of our circles and points to the variable RADIUS and modify the piecewise declaration to be:

$$f(x) = \begin{cases} I & x \le -3 + \frac{1\,RADIUS}{2} \\ 2\,x + 3 & x \le -1 - \frac{1\,RADIUS}{2} \\ I & x \le -1 + \frac{1\,RADIUS}{2} \\ x + 3 & x \le 1 - RADIUS \\ I & x \le 1 + RADIUS \\ 2 & x \le 3 - RADIUS \end{cases}$$

The "RADIUS/2" in the above is accounting for the slope of the line
$y = 2\,x + 3$
. In practice this adjustment is usually made visually. That is, mulipliers are substituted until the graph looks correct.

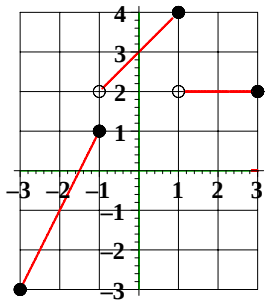
RADIUS:=.15: S:=PP([-3,-3],RADIUS,black),PP([-1,1],RADIUS,black),PP([1,4],RADIUS,bl ack),PP([3,2],RA
CIRCS:=PC([-1,2],RADIUS,2,1,black),PC([1,2],RADIUS,2,1,black):
f:=x->piecewise(x<=-3+RADIUS/2,I,x <=-1-RADIUS/2,2*x+3,x<=-1+RADIUS/2,I,x <=1-
RADIUS,x+3,x<=1+RADIUS,I,x <=3-RADIUS,2); gp:=GP([-3,-3],6,7,1,14,green):
plt:=plot(f(x),x=-3..3,discont=true,thickness=5,axesfont=[TIMES,BOLD,1 2]):
dgm:=plots[display]([CIRCS,S,plt,gp],scaling=constrained): dgm;



One final problem with the above daigram is that the points which meet the boundary may be truncated. This is caused by the limits on the function plot. It can often be corrected by ploting on a larger domain but it is generally easier to adjust the view of the plot. We change only the plots[display] command in the above, making it:

dgm:=plots[display]([CIRCS,S,plt,gp],scaling=constrained,view=[-3-RADI US.. 3+RADIUS,-3-RADIUS.. 4+RADIUS]):

Which yields



Of course we don't have to be so precise with the "slack" we introduce by expanding the view. The reader should remember that the above only works for constrained plots. Unconstrained plots will cause the circles and disks to appear elliptical.

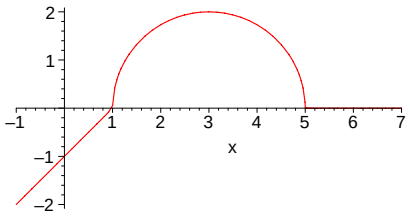
```
> RADIUS:=.15:
  WQPTS:=PP([-3,-3],RADIUS,color=black),PP([-1,1],RADIUS,color=black),
  PP([1,4],RADIUS,color=black),PP([3,2],RADIUS,color=black):
> CIRCS:=PC([-1,2],RADIUS,thickness=2,linestyle=1,color=black),
  PC([1,2],RADIUS,thickness=2,linestyle=1,color=black):
> f:=x->piecewise(x<=-3+RADIUS/2,I,x<=-1-RADIUS/2,2*x+3,x<=-1+RADIUS/2,I
  ,
  x<=1-RADIUS,x+3,x<=1+RADIUS,I,x<=3-RADIUS,2):
> gp:=GP(llcorner=[-3,-3],width=6,height=7,resolution=1,tickmarkfont=14
  ,color=green):
> plt:=plot(f(x),x=-3..3,discont=true,thickness=5,axesfont=[TIMES,BOLD,
  12]):
  dgm:=plots[display]([CIRCS,WQPTS,plt,gp],scaling=constrained,view=[-3-
```

```
> RADIUS..  3+RADIUS,-3-RADIUS..  4+RADIUS ]):
  dgm:=plots[display]([CIRCS,plt,gp],scaling=constrained):
  dgm;
```

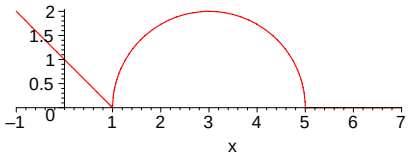
13.12 Exercises

Exercise 1: Many calculus books use piecewise defined functions made up of straight lines and portions of circles as examples where derivatives can be calculated at most points by graphical techniques. For instance the following code produces the diagram below:

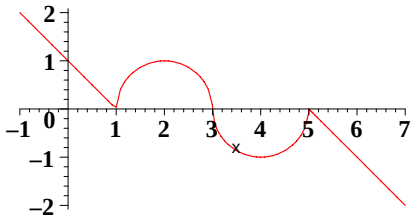
```
f:=x->piecewise(x<=1,x,x <=5,sqrt(4-x^2),x<=7,0): plt:=plot(f(x),x=-1..7,thickness=3):
plt;
```



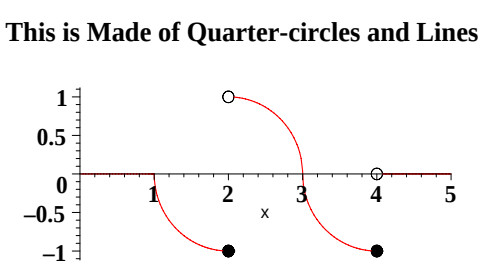
Adapt the above code to produce the following diagrams. Part (d) contains a portion of the graph of $\sin(\frac{1}{x})$.

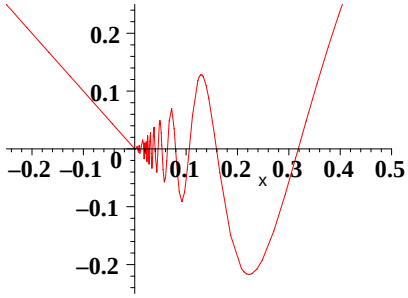


This is Made of Semi-circles and Lines



(a)





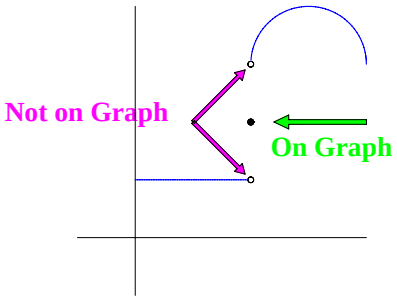
(c)

(e)

```
> f:=x->piecewise(x<=1,x-1,x<=5,sqrt(4-(x-3)^2),x<=7,0):
> plt:=plot(f(x),x=-1..7,thickness=3,scaling=constrained):
> plt;
> f:=x->piecewise(x<=1,1-x,x<=5,sqrt(4-(x-3)^2),x<=7,0):
> plt:=plot(f(x),x=-1..7,thickness=3,scaling=constrained):
> plt;
> f:=x->piecewise(x<=1,1-x,x<=3,sqrt(1-(x-2)^2),x<=5,-sqrt(1-(x-4)^2),x
  <=7,5-x):
> plt:=plot(f(x),x=-1..7,thickness=3,scaling=constrained,axesfont=[TIME
  S,BOLD,14],titlefont=[TIMES,BOLD,16],title=`This is Made of
  Semi-circles and Lines`):
> plt;
> RADIUS:=.075:
> PTS:=PP([2,-1],RADIUS,color=black),PP([4,-1],RADIUS,color=black):
> CIRCS:=PC([2,1],RADIUS,thickness=2,lineatyle=1,color=black),PC([4,0],
  RADIUS,thickness=2,lineatyle=1,color=black):
> f:=x->piecewise(x<=1,0,x<=2-RADIUS,-sqrt(1-(x-2)^2),
  x<=2+RADIUS,I,x<=3,sqrt(1-(x-2)^2),x<=4-RADIUS,-sqrt(1-(x-4)^2),x<=4+R
  ADIUS,I,0):
> plt:=plot(f(x),x=0..5,thickness=3,discont=true):
  diagrm:=plots[display]([plt,PTS,CIRCS],scaling=constrained,axesfont=[T
> IMES,BOLD,14],titlefont=[TIMES,BOLD,16],title=`This is Made of
  Quarter-circles and Lines`):
> diagrm;
> f:=x->piecewise(x<=0,abs(x),x<=1,x*sin(1/x)):
> plt:=plot(f(x),x=-1..1,thickness=1,discont=true):
  diagrm:=plots[display]([plt],axesfont=[TIMES,BOLD,14],view=[-.25..
  .5,-.25..  .25]):
> diagrm;
```

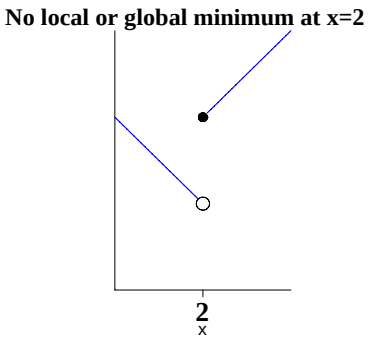
Exercise 2:

Create the diagram below:




```
> RADIUS:=.05:
  f:=x->piecewise(x<2-RADIUS,1,x<2+RADIUS/20,I,x<4,3+sqrt(1-(x-3)^2)):
  plt:=plot(f(x),x=0..4,discont=true,color=blue,thickness=3):
> PTS:=PP([2,2],RADIUS,color=black):
  CIRCS:=PC([2,1],RADIUS,thickness=2,linestyle=1,color=black),PC([2,3],
  RADIUS,linestyle=1,color=black):
> mctools(ARRW);
> ARRWS:=ARRW(tail=[1,2],head=[2-.1,3-.1],color=magenta,arrowtype=`SH`,
  shaftthickness=.05,headwidth=3,headlength=.1,doublearrowtxt=`
  `),ARRW(tail=[1,2],head=[2-.1,1+.1],color=magenta,arrowtype=`SH`,shaft
> thickness=.05,headwidth=3,headlength=.1,doublearrowtxt=`
  `),ARRW(tail=[4,2],head=[2.4,2],color=green,arrowtype=`SH`,shaftthickn
  ess=.05,headwidth=3,headlength=.1,doublearrowtxt=` `):
> TXT:=PT([-.7, 2.2],`Not on Graph`,fontsize=16,color=magenta),PT([3.5,
  1.6],`On Graph`,fontsize=16,color=green):
> diagram:=plots[display]([plt,PTS,CIRCS,ARRWS,TXT],scaling=constrained
  ,view=[-1..4,-1..4],xtickmarks=[],ytickmarks=[]):
  diagram;
```

Exercise 3:
Create the following diagram.




```
> RADIUS:=.05:
> f:=x->piecewise(x<2-RADIUS,3-x,x<2+RADIUS,I,x<4,x):
> plt:=plot(f(x),x=0..4,discont=true,color=blue,thickness=3):
> PTS:=PP([2,2],RADIUS,color=black):
```

```
> CIRCS:=PC([2,1],1.5*RADIUS,thickness=2,lineatyle=1,color=black):
> ARRWS:=ARRW(tail=[1,2],head=[2-.1,3-.1],color=magenta,arrowtype=‘SH‘,
  shaftthickness=.05,headwidth=3,headlength=.1,doublearrowtxt=‘
  ‘),ARRW(tail=[1,2],head=[2-.1,1+.1],color=magenta,arrowtype=‘SH‘,shaft
> thickness=.05,headwidth=3,headlength=.1,doublearrowtxt=‘
  ‘),ARRW(tail=[4,2],head=[2.4,2],color=green,arrowtype=‘SH‘,shaftthickn
  ess=.05,headwidth=3,headlength=.1,doublearrowtxt=‘ ‘):
> TXT:=PT([-.7, 2.2],‘Not on Graph‘,fontsize=16,color=magenta),PT([3.5,
  1.6],‘On Graph‘,fontsize=16,color=green):

> diagram:=plots[display]([plt,PTS,CIRCS],scaling=constrained,view=[-1.
  .4,-1..4],axesfont=[TIMES,BOLD,16],xtickmarks=[2],ytickmarks=[],titlef
  ont=[TIMES,BOLD,14],title=‘No local or global minimum at
> x=2‘,view=[1..3,0..3]):
  diagram;
```

Remarks: Although it is expected in future releases ut is not possible with Maple 9 or earlier versions to generate Maple diagrams with formatted mathematics. One of the simplest ways to partially address this deficiency is to use Power Point which has some very good tools.

One can simply copy Maple diagrams and the formatted text one would like to add to the diagram separately and paste them into a Power Point slide with the text over the diagram. The slide then be exported to a graphics form such as jpg, gif, etc. and copied onto the final document. This works well for images to be placed in Power Point presentations and Word documents but not in Maple for although such images can be copied into the Maple worksheet they cannot be resized, printed, or exported to html.

14 Homeworks with multiplicity, or versioned homework

When you post a homework in WHS with the header tag H_[n], where n is some small positive integer > 1, then only one problem from each successive group of n consecutive problems starting at the top is chosen at random (using a seed number which can be specified by the student at the time of homework retrieval) from the source html to make up the homework assignment. This is referred to as a **homework version** retrieved from a homework with **multiplicity n** .

Three choices are given at the time of homework retrieval:

The **common version** is version number -1. It is not chosen randomly, rather it is chosen as the first problem in each group of n problems. This is usually the homework that is discussed in class if anywhere.

The **personal version** is the version whose number is the student's WHS id number. The problems in the personal versions are chosen from the last n-1 problems in each set of n problems. So, no common problem occurs in any personal version (unless duplications are built into the problem set. This is the problem set the student has the responsibility of submitting solutions for before the deadline if any is set.

The **specific version** is the version you get by specifying a version number. If you specify -1, you get the common version. If you don't specify a version number one will be chosen at random.

So for example, if your source homework has 20 problems and has been posted with multiplicity 3, then any version of it has 10 problems, one from the first 3, one from the second 3, and so forth. Further, the order of occurence of the problems is scrambled, unless it is turned off by the author. This allows an author to create a homework with a large number number of different versions. based on as few as 10 problems. Just parameterize 10 problems and create 3 instances of each one in a source homework, and then post it with mulitplicity 3. There will be $3^{10}\,10! = 214277011200$ different homeworks. Of course, many of them will have the same problems; just in a different order. And too, there are only $2^{10}\,10! = 3715891200$ possible personal versions. Actually, if you read the author documentation on mathclass, there is a way to turn off randomization: Put the header tag in with a third argument of -1. So H_[3;0;-1] gives a homework of multiplicity 3 with each retrieved homework in the same order. If there are 10 problems, there will be only $2^{10} = 1024$ different versions possible.

Problem: How many different versions does a 20 question homework of multiplicity 2 have? How many different personal versions are there?

14.1 Sample: First 4 problems of the AMSP Practice homework for the Delayed Credit Pre-calculus exam.

We include these problems as an example of one way to versioned a versioned homework. Each problem sits in its own section, which is titled appropriately. Included in the section is at least the problem generator and a number of calls to it. In addition, any computations made to solve the parameterized problem should be included along with any text comments the author feels appropriate.

14.1.1 #1 inscribed circle

Problem. A circle is inscribed in a square of side s. Find the area inside the square but outside the circle.

Answer: s^2 - Pi*(s/2)^2

We first make a parameterized diagram for the problem. Note the diagram uses the word **ARRW** from the MCtools package along with some word from the plottools package.

```
> pic := proc(s,c1,c2,c3)
  plots[display](ARRW(tail=[0,-.1],head=[1,-.1],doublearrowtxt=s,color=c
  1),
> ARRW(tail=[-.1,0],head=[-.1,1],doublearrowtxt=s,color=c1),
  plottools[disk]([.5,.5],.5,color=c2),
  plottools[polygon]([[0,0],[1,0],[1,1],[0,1]],color=c3),
  scaling=constrained,axes=none) end:
```

Then we construct the problem generator.

```
> prob1 := proc(s,c1,c2,c3)
  local ans;
  ans:=[s^2-(s/2)^2*Pi,s^2*Pi-s^2,s^2-(s/2)^2*Pi+1,(s/2)^2*Pi-(s/2)^2,(s
> /2)^2*Pi-s^2];
  tagit(["A circle is inscribed in a square ",s," inches on a side as
  shown.  What is the area, in square inches, of the region inside the
> square and outside the circle?"],pic(s,c1,c2,c3),_ALlabels(ans)) end:


> prob1(1,red,pink,grey) ;   #1.1
> prob1(2,blue,yellow,grey); #1.2
```

14.1.2 #2 identify a factor

This is a checkbox problem, because we want the student to recognize both linear factors. It uses the MCtools word **Line** to keep the expression inline with the text.

```
> prob2 := proc(x,a,b,c,d)
  tagit(Line("Which of the following expressions is a factor of "
  ,expand(c*(d*x-a)*(x-b)),"?"),
  _ABlabels([[d*x-a-1],[x-b],d*x + a,x+b,c*x])) end:
> prob2(x,2,3,2,2);     #2.1;
> prob2(y,2,3,3,2);     #2.2;
```

14.1.3 #3 expand a quadratic

Here the list of alternatives was parameterized outside the problem generator.

```
> alternatives := proc(x,a,b,c,d)
  [a*(x^2-b*x)-c*(x^2-d*x),-a*(x^2-b*x)+c*(x^2-d*x),
  a*(x^2-b*x)-c*(x^2+d*x),a*(x^2-b*x)-c*(x^2-d/c*x),a*(x^2-b/a*x)-c*(x^2
  -d*x)]  end:
> alternatives(x,3,2,5,4);
```

$$[-2\,x^2+14\,x,\ 2\,x^2-14\,x,\ -2\,x^2-26\,x,\ -2\,x^2-2\,x,\ -2\,x^2+18\,x]$$

a is converted to a symbol so that it won't be multiplied out.

```
> prob3 := proc(x,a,b,c,d)
  tagit("In simplified form,
  ",convert(a,symbol)*(x^2-b*x)-convert(c,symbol)*x*(x-d)," =
  ?",inline=yes,_ALlabels(alternatives(x,a,b,c,d))) end:

> prob3(x,2,3,5,1); #3.1
> prob3(y,3,2,5,1); #3.2
```

14.1.4 #4 solve a linear rational equation

Here is another problem that requires the input parameters be symbolized in order to keep the computation from being made by Maple's parser.

```
> prob4 := proc(x,a,b,c)
  tagit("If the equation ",
  1/(x-convert(a,symbol))=convert(b,symbol)/convert(c,symbol),"  is
> solved for ",x,", then ",x," = ?"
  ,_ALlabels([(c+b*a)/b,c/(b+a*c),-(b/c+a),c/b-a,b/c-a])) end:

> prob4(x,3,3,5); #4.1
> prob4(y,5,3,4); #4.2
```

14.1.5 #5nickle and dime problem

This problem uses the MCtools word **roundto** .

```
> mctools(roundto);
```

```
MCtools, version June 1 2004.

roundto(x,places=3)
```

```
> prob5 := proc(z)
  tagit(["A child has x nickles, y dimes and no other coins in her piggy
  bank.  If the total amount of money in the bank is ",z," dollars, which
> of the following equations is satisfied by x and y?"],
  _ALlabels([`0.05`*x+`0.10`*y=roundto(z,places=2),5*x+10*y=roundto(z,pl
  aces=2),x+y=roundto(z,places=2),`0.05`*x+`0.10`*y=ceil(100*z)
  ,x+y=ceil(100*z)])) end:

> prob5(5.25);#5.1
> prob5(4.25);#5.2
```

15 Printing WHS homeworks (for quizzes and tests)

15.1 Ways to print out WHS homeworks

How can a teacher use WHS homework problems as a source for **hardcopy quizzes** or exams? There are several ways.

1. You can retrieve the homework in mathclass, and just print from the browser. I have used this many times to make up pre-tests for a class. You may want to cut and paste the problems to save a little paper.

2. You can bring up the source Maple worksheet (word document) with the homework problems in it, then cut and paste the parts you want into a source quiz or test document.

3. If the problems are generated from tagit in the freeform format, you can do this: a) copy the tagit source of each problem you want in the quiz or exam all into the same input cell b) at the top of the input cell, insert the assignements **PRINT_** = "yes": **PNUM_** := 1: c) if you want to provide space for the student to work on the test, insert the assignment SPACES_ := 5 to provide 5 blank lines after each problem. d) at the bottom of the input cell, insert the assignment PRINT_="no" e) execute the input cell. A reasonable copy of the quiz or test should come out. You will have to do some clean up.

4. If latex quality homeworks and exams are desired, then you can use the **latextools** package. This is a separate module from MCtools, but is coordinated with MCtools. You can get the most recent version, together with a manual (which you will need to peruse) and samples of use from ***http://www.msc.uky.edu/carl/communicating_math/MCtools_page.htm*** Here is an outline of the steps involved: a) Execute the latextools package, then the MCtools package. b) Open the homework worksheet and at the top set Latex_ (with all caps) := yes. Then go thru and re-execute all of the tagit lines. You will get output suitable for latex instead of html. c) Export the worksheet to latex. d) Execute **extractlatex** and **latexit** with appropriate options. Follow the examples given in the manual.

This has been used to produce multiple versions of an exam which can be given in latex hardcopy and also given in html on WHS.

15.2 Example: a quiz

```
> PRINT_:="yes":PNUM_:=1:SPACES_:=5:
  mcprint("Algebra I quiz   June 18, 2003   Name________ \n
  Instructions:    Show all your work, as unsupported answers will
> recieve no credit.");
  print(matrix([[`Problem`,`Possible Points`,`Credit`],
  [1,10,`________`],[2,10,`________`],
> [3,10,`________`],[`Total`,30,`________`]]));
  #1
> tagit(  "Bill can mow a yard in 3 hours.  Jim can mow the same yard in
  5 hours.    How many hours does it take Bill and Jim to mow the yard
  together?",_AC(1/(1/3+1/5),txtboxsize=4,"Assume they do not interfere
> with each other.")):
  #2
  tagit("3 + 4 =",_AC(7),"   5 + 12 =",_AC(17),"   4 + 1 =",_AC(5)):
```

```
> #3
  tagit("Bill can mow a yard in 3 hours.  Jim can mow the same yard in 5
  hours.    How many hours does it take Bill and Jim to mow the yard
> together, assuming they do not interfere with each other?  Select the
  most nearly correct answer.",_AS([1.875,2.125,4,"None of the
  others"])):
> PRINT_:= "no":
```

```
Algebra I quiz   June 18, 2003   Name________

 Instructions:  Show all your work, as unsupported answers will
recieve no credit.
```

Problem	*Possible Points*	*Credit*
1	10	_____
2	10	_____
3	10	_____
Total	30	_____

```
1.  Bill can mow a yard in 3 hours. Jim can mow the same yard in 5
hours.  How many hours does it take Bill and Jim to mow the yard
together?

----------

Assume they do not interfere with each other.



2.  3 + 4 =

----------

   5 + 12 =

----------

   4 + 1 =

----------



3.  Bill can mow a yard in 3 hours. Jim can mow the same yard in 5
hours.  How many hours does it take Bill and Jim to mow the yard
together, assuming they do not interfere with each other? Select the
most nearly correct answer.
```

Circle correct answer :, [2.125 4 "None of the others" 1.875]

15.3 Making test generators

You can take the input cell which generates the quiz and make a **quiz generator** out of if. Here is one such generator. I parameterized the date, and in each problem parameterized one of the numbers. **Note:** In order to keep those from printing out centered on a separate line, you must enclose the problem statement in square brackets, thus making a list out of it.

```
> quiz1 := proc(dat,a,b,c)
  global PRINT_,PNUM_,SPACES_;
  PRINT_:="yes":PNUM_:=1:SPACES_:=5:
```

```
> mcprint("Algebra I quiz   ",dat,", 2003   Name________ \n
  Instructions:    Show all your work, as unsupported answers will
  recieve no credit.");
> print(matrix([[‘Problem‘,‘Possible Points‘,‘Credit‘],
  [1,10,‘________‘],[2,10,‘________‘],
  [3,10,‘________‘],[‘Total‘,30,‘________‘]]));
> #1
  tagit(  ["Bill can mow a yard in ",a," hours.  Jim can mow the same
> yard in 5 hours.    How many hours does it take Bill and Jim to mow the
  yard together?"],_AC(1/(1/a+1/5),txtboxsize=4,"Assume they do not
  interfere with each other.")):
> #2
  tagit([c," + 4 ="],_AC(c+4),"   5 + 12 =",_AC(17),"   4 + 1
  =",_AC(5)):
> #3
  tagit(["Bill can mow a yard in 3 hours.  Jim can mow the same yard in
  ",b," hours.    How many hours does it take Bill and Jim to mow the yard
> together, assuming they do not interfere with each other?  Select the
  most nearly correct
  answer."],_AS([evalf(1/(1/3+1/b),4),evalf((1/3+1/b),4),evalf(1/(1/3+1/
> b),4)+1,"None of the others"],randomize="no")):
  PRINT_:= "no":
  NULL:
  end:
> quiz1("Makeup  June 21",4,7,6);
```

16 Using standards= to make tests and homework based on State Standards

The **standards=** option in tagit is useful for aligning problems you are constructing or have already constructed with the set of standards your state uses to measure the progress of your students. What is needed is a uniform way of naming the standards, so that you can avoid having to type in the standards each time and so avoid typos. Tennessee and Kentucky are two states we are most familiar with. In the sections below, we describe the **Tennessee standards K-8** and then give some examples of problems which have been composed with tagit using the standards= option. Also, we describe a new set of national standards: the **American Diploma Project** standards.

16.1 Tennessee Course Standards K thru 8

There are 5 Standards in the Tennessee Setup.

1 Number Sense and Number Theory: The student will recognize, represent, model, and apply real numbers and operations verbally, physically, symbolically, and graphically.

2 Estimation, Measurement, and Computation The student will apply appropriate tools and units of measurement; develop effective estimation and computation strategies for producing reasonable results; and calculate using appropriate tools such as mental mathematics, technology, manipulatives, and pencil-and-paper.

3 Patterns, Functions, and Algebraic Thinking: The student will describe, extend, analyze, and create a wide variety of patterns and functions using appropriate materials and representations in real world problem solving.

4 Statistics and Probability: The student will collect, organize, represent, and interpret data; make inferences and predictions; present and evaluate inferences and predictions; present and evaluate arguments based on data analysis; and model situations to determine theoretical and experimental probabilities.

5 Spatial Sense and Geometric Concepts: The student will investigate, model, and apply geometric properties and relationships.

The standards are organized by grade (3 thru 8) , then by standard (1 thru 5), then by performance indicator (state or teacher). Each indicator has a number to distinguish it from the others in that category. So each indicator has a label of the form **x.y.zpi.w** where x is the grade level, y is the standard, zpi is spi (state performance indicator) or tpi (teacher performance indicator), and w is the specific indicator number in that branch. Thus **5.3.tpi.2** is the label for the 5th grade, 3rd standard, 2nd teacher performance indicator.. After you excute the input cell below, you can use **standards=[G53tpi2]** in the tagit line of the problem generator if the problems it generates test that indicator. This makes it relatively painless to make your homework standards based.

There are three **levels** of expectation in the Tennessee system. The indicators of any given type are arranged in or of increasing expectation. Thus for the 3rd grade, standard 1, state performance indicators, 3.1.spi.1 is level 1 and 3.1.spi.16 is level 3. The breakpoint from one level to the next are not part of the labelling scheme, perhaps intentionally.

To see the breakpoints and get more information about the standards, go to the state site.
http://www.state.tn.us/education/ci/cistandards2001/math/cimath.htm

There is a complete list of the 533 indicators for the grades 3 thru 8 for Tennessee.

Examples of how to reference these definitions in WHS homework are given below. Below, we give truncated sections showing a few of the standards for each grade. The complete standards are defined in the standards worksheet at:
http://www.msc.uky.edu/carl/communicating_math/MCtools_page.htm

16.1.1 K-3rd grade

```
> G31spi1 :="3.1.spi.1.  count by 10s, 100s, and 1000s":
  G31spi2 :="3.1.spi.2.  identify whole numbers as odd or even":
  G31spi3 :="3.1.spi.3.  add and subtract efficiently and accurately with
> single-digit whole numbers":
  G31spi4 :="3.1.spi.4.  represent whole numbers to 9999 with models":
  G31spi5 :="3.1.spi.5.  identify the place value of a given digit up to
> thousands":
  G31spi6:="3.1.spi.5.  recognize the value of combinations of coins and
  bills up to 5 dollars.":
> G31spi7:="3.1.spi.7.  compare and order whole numbers to 9999 using the
  appropriate symbol for less than, greater than and equal.":
```

Standards omitted ..

```
> G35tpi2 :="3.5.tpi.2.  create tables using tally marks":
  G35tpi3 :="3.5.tpi.3.  pose questions and gather data to answer
  questions":
> G35tpi4 :="3.5.tpi.4.  develop an appropriate method to collect data":
  G35tpi5 :="3.5.tpi.5.  select an appropriate method to display data":
  G35tpi6 :="3.5.tpi.6.  explain whether an event is certain, possible,
> or impossible":
  G35tpi7 :="3.5.tpi.7.  explain whether an event is likely or unlikely":
  G35tpi8 :="3.5.tpi.8.  make conjectures based on data gathered and
> displayed":
  G35tpi9 :="3.5.tpi.9.  predict outcomes of events based on data
  gathered and displayed":
```

16.1.2 4th grade

```
> G41spi1 :="4.1.spi.1.  represent whole numbers to 9999":
  G41spi2 :="4.1.spi.2.  compare and order whole numbers to 9999 using
  the appropriate symbol for greater than, less than, or equal.":
> G41spi3 :="4.1.spi.3.  solve one-step real-world problems involving
  addition or subtraction of whole numbers.":
  G41spi4 :="4.1.spi.4.  read and write numbers from hundred-thousands to
  hundredths":
```

Standards omitted ...

```
> G45spi5 :="4.5.spi.5.  determine the median of a data set":
  G45tpi1 :="4.5.tpi.1.  collect data using observations, surveys, and
  experiments":
> G45tpi2 :="4.5.tpi.2.  construct bar graphs and line graphs from data
  in a table":
  G45tpi3 :="4.5.tpi.3.  evaluate how well various representations show
> the collected data.":
  G45tpi4 :="4.5.tpi.4.  understand how data collection methods affect
  the nature of the data set":
> G45tpi5 :="4.5.tpi.5.  design investigations to address a question":
  G45tpi6 :="4.5.tpi.6.  explain differences in measures of center (mean,
  median, mode).":
```

16.1.3 5th grade

```
> G51spi1 :="5.1.spi.1.  read and write numbers from millions to
  thousandths":
  G51spi2 :="5.1.spi.2.  connect symbolic representations of proper and
> improper fractions to models of proper and improper fractions":
  G51spi3 :="5.1.spi.3.  represent whole numbers and two-place decimals
  in expanded form.":
```

Standards omitted ..

```
> G55tpi6 :="5.5.tpi.6.  create a sample space to predict the
  probability of an event":
  G55tpi7 :="5.5.tpi.7.  explain the importance of sample size in
> investigations":
  G55tpi8 :="5.5.tpi.8.  examine various representations of data and
  evaluate how accurately the data is depicted by the graph":
```

```
> G55tpi9 :="5.5.tpi.9.  understand how data collection methods affect
  the nature of the data set":
  G55tpi10 :="5.5.tpi.10.  explain which measure of central tendency best
> represents a given data set":
```

16.1.4 6th grade

```
> G61spi1 :="6.1.spi.1.  identify the place value of a given digit":
  G61spi2 :="6.1.spi.2.  solve one-step real-world problems involving
  whole numbers and decimals":
> G61spi3 :="6.1.spi.3.  represent numbers using a variety of models and
  equivalent forms (i.e., whole numbers, mixed numbers, fractions,
  decimals, and percents)":
> G61spi4 :="6.1.spi.4.  connect whole numbers, mixed numbers, fractions,
  and decimals to locations on the number line":
```

Standards omitted ....

```
> G65tpi4 :="6.5.tpi.4.  determine an appropriate sample to test a
  hypothesis":
  G66tpi5 :="6.6.tpi.5.  conduct simple experiments to compare
> theoretical and experimental probabilities":
  G65tpi6 :="6.5.tpi.6.  explain the importance of sample size in surveys
  and research":
> G65tpi7 :="6.5.tpi.7.  make conjectures from data to formulate new
  questions for further investigation":
```

16.1.5 7th grade

```
> G71spi1 :="7.1.spi.1.  identify prime and composite numbers up to 50":
  G71spi2 :="7.1.spi.2.  compute efficiently and accurately with whole
  numbers, fractions, and decimals":
> G71spi3 :="7.1.spi.3.  represent numbers using a variety of equivalent
  forms (i.e., mixed numbers, fractions, decimals, percents, and
  integers)":
> G71spi4 :="7.1.spi.4.  compare rational numbers using the appropriate
  symbol for greater than, less than, and equal":
```

Standards omitted ...

```
> G75tpi5 :="7.5.tpi.5.  identify appropriate data sources to address
  questions or hypotheses":
  G75tpi6 :="7.5.tpi.6.  construct tree diagrams to determine outcomes of
> a simple compound event":
  G75tpi7 :="7.5.tpi.7.  make conjectures to formulate new questions for
  future studies":
> G75tpi8 :="7.5.tpi.8.  describe why or why not predictions can be made
  for a population based on survey results for a given sample":
  G75tpi9 :="7.5.tpi.9.  communicate the difference in theoretical and
  experimental probabilities":
```

16.1.6 8th grade

```
> G81spi1 :="8.1.spi.1.  identify the opposite and the reciprocal of a
  rational number":
  G81spi2 :="8.1.spi.2.  compare rational numbers using the appropriate
> symbol for greater than, less than and equal":
  G81spi3 :="8.1.spi.3.  use ratios and proportions to represent
  real-world situations (i.e., scale drawings, probability)":
```

Standards omitted ...

```
> G85tpi5 :="8.5.tpi.5.  develop meaning for lines of best fit":
  G85tpi6 :="8.5.tpi.6.  determine an appropriate sample to test a
  hypothesis":
> G85tpi7 :="8.5.tpi.7.  make conjectures to formulate new questions for
  future studies":
  G85tpi8 :="8.5.tpi.8.  use a variety of methods to compute
> probabilities for compound events (i.e., multiplication, organized
  lists, tree diagrams, area models)":
  G85tpi9 :="8.5.tpi.9.  develop meaning of mutually exclusive events":
> G85tpi10 :="8.5.tpi.10.  find the probability of dependent and
  independent events":
```

16.2 Kentucky Core Content Standards

The Kentucky setup has 4 **strands**

1 Number/Computation
2 Geometry/Measurement
3 Probability/Statistics
4 Algebraic Ideas.

which are divided into 3 **types**

1 Concepts
2 Skills
3 Relations

Each strand has a number of individual standards which are grouped by school level into Elementary, Middle, and High School. So the individual standards have labels of the form MA-X-x.y.z where X = E, M or H, x = strand number, y = type number, and z is the specific standard in that branch.

So the label MA-M-1.2.3 denotes the 3rd skills standard in the Geometry/Measurement Strand of Middle School.

To get more information on the **Kentucky core content assessment standards** , go to the state site:

http://www.kde.state.ky.us/KDE/Instructional+Resources/Curriculum+Doc uments+and+Res

There are 175 core content assessment standards for Kentucky.

Below, we give truncated sections showing a few of the standards for grade group. The complete standards are defined in the standards worksheet at :

http://www.msc.uky.edu/carl/communicating_math/MCtools_page.htm

open the appropriate section below and execute to define the standards

16.2.1 Elementary School K-5

```
> MAE111:="MA-E-1.1.1:Whole numbers, fractions, mixed numbers, and
  decimals through thousandths":
  MAE112:="MA-E-1.1.2:The operations of addition, subtraction,
> multiplication, and division":
  MAE113:="MA-E-1.1.3:Odd and even numbers, composite and prime numbers,
  multiples, and factors":
  MAE114:="MA-E-1.1.4:Place value, expanded form, number magnitude":
```

Standards omitted ...

```
> MAE424:="MA-E-4.2.4:Locate whole numbers, fractions, and decimals on
  a number line":
  MAE425:="MA-E-4.2.5:Graph ordered pairs on a positive coordinate
> grid":
  MAE431:="MA-E-4.3.1:How patterns are alike and different":
  MAE432:="MA-E-4.3.2:How rules involving number patterns can be
> explained":
```

16.2.2 Middle School 6-8

```
> MAM111:="MA-M-1.1.1:Rational numbers (integers, fractions, decimals,
  percents)":
  MAM112:="MA-M-1.1.2:Irrational numbers (square roots and  only)":
> MAM113:="MA-M-1.1.3:Meaning of proportion (equivalent ratios)":
  MAM114:="MA-M-1.1.4:Place value of whole numbers and decimals":
  MAM115:="MA-M-1.1.5:Positive whole number exponents":
```

Standards omitted ...

```
> MAM425:="MA-M-4.2.5:Represent and use functions through tables,
  graphs, verbal rules, and equations":
  MAM426:="MA-M-4.2.6:Write and solve equations that represent everyday
> situations":
  MAM1431="MA-M-4.3.1:How everyday situations, tables, graphs, patterns,
  verbal rules, and equations relate to each other":
> MAM1432="MA-M-4.3.2:How the change in one variable affects the change
  in another variable":
```

16.2.3 High School 9-11

```
> MAH111:="MA-H-1.1.1:Students will describe properties of, define,
  give examples of, and apply real numbers, and understand that
  irrational numbers cannot be represented by terminating or repeating
> decimals.":
  MAH112:="MA-H-1.1.2:Students will recognize, define, give examples of,
  and apply finite arithmetic and geometric sequences and series.":
> MAH113:="MA-H-1.1.3:Students will understand how matrices are used to
  represent  data.":
```

Standards omitted ...

```
> MAH433:="MA-H-4.3.3:Students will demonstrate how slope shows rate of
  change in linear functions arising in problems":
  MAH434:="MA-H-4.3.4:Students will show how changes in parameters
```

```
> affect graphs of functions [e.g., compare the graphs y = x2,  y = 2x2,
  y = (x - 4)2 , and y = x2 + 3]":
  MAH435:="MA-H-4.3.5:Students will show how equations and graphs are
> models of the relationship between two quantities (e.g., the
  relationship between degrees Celsius and degrees Fahrenheit)":
```

16.3 American Diploma Project Standards

The source for these standards:

http://www.achieve.org/achieve.nsf/ADP-Benchmarks-Samples?openform

The standards are part of a proposed set of national standards that any high school graduate should satisfy, whether or not they plan to go to college. Because major areas of study at post-secondary institutions have different prerequisites, certain mathematics benchmarks are marked with an asterisk (*). These asterisked benchmarks represent content that is recommended for all students, but is required for those students who plan to take calculus in college, a requisite for mathematics and many mathematics-intensive majors. Note: Only Areas J and K have asterisked benchmarks.

Below, we give truncated sections showing a few of the standards for each thread. The complete standards are defined in the standards worksheet at:

http://www.msc.uky.edu/carl/communicating_math/MCtools_page.htm

16.3.1 I Number Sense and Numerical Operations

```
> I1:="Compute fluently and accurately with rational numbers without a
  calculator:":
  I1d1:="Add, subtract, multiply and divide integers, fractions and
> decimals.":
  I1d2:="Calculate and apply ratios, proportions, rates and percentages
  to solve problems.":
> I1d3:="Use the correct order of operations to evaluate arithmetic
  expressions, including those containing parentheses.":
```

Standards omitted...

```
> I4:="Understand the capabilities and the limitations of calculators
  and computers in solving problems:":
  I4d1:="Use calculators appropriately and make estimations without a
> calculator regularly to detect potential errors.":
  I4d2:="Use graphing calculators and computer spreadsheets.":
```

16.3.2 J Algebra

```
> J1:="Perform basic operations on algebraic expressions fluently and
  accurately:":
  J1d1:="Understand the properties of integer exponents and roots and
> apply these properties to simplify algebraic expressions.":
  J1d2:="*Understand the properties of rational exponents and apply
  these properties to simplify algebraic expressions.":
> J1d3:="Add, subtract and multiply polynomials:  divide a polynomial by
  a low-degree polynomial.":
```

Standards omitted ...

```
> J5d3:="Recognize and solve problems that can be modeled using a
  quadratic equation, such as the motion of an object under the force of
  gravity.":
> J5d4:="Recognize and solve problems that can be modeled using an
  exponential function, such as compound interest problems.":
  J5d5:="*Recognize and solve problems that can be modeled using an
> exponential function but whose solution requires facility with
  logarithms, such as exponential growth and decay problems.":
  J5d6:="Recognize and solve problems that can be modeled using a finite
> geometric series, such as home mortgage problems and other compound
  interest problems.":
  J6:= "*Understand the binomial theorem and its connections to
  combinatorics, Pascal's triangle and probability.":
```

16.3.3 K Geometry

```
> K1:="Understand the different roles played by axioms, definitions and
  theorems in the logical structure of mathematics, especially in
  geometry:":
> K1d1:=" Identify, explain the necessity of and give examples of
  definitions, axioms and theorems.":
  K1d2:="State and prove key basic theorems in geometry such as the
> Pythagorean theorem, the sum of the angles of a triangle is 180
  degrees, and the line joining the midpoints of two sides of a triangle
  is parallel to the third side and half its length.":
```

Standards omitted ...

```
> K12d2:="*Know and use the basic trig identities,  and  formulas for
  sine and cosine, such as addition and double angle formulas.":
  K12d3:="*Graph sine, cosine and tangent as well as their reciprocals,
> secant, cosecant and cotangent; identify key characteristics.":
  K12d4:="*Know and use the law of cosines and the law of sines to find
  missing sides and angles of a triangle.":
```

16.3.4 L Data Interpretation, Statistics and Probability

```
> L1:="Explain and apply quantitative information:":
  L1d1:="Organize and display data using appropriate methods (including
  spreadsheets) to detect patterns and departures from patterns.":
> L1d2:="Read and interpret tables, charts and graphs.":
  L1d3:="Compute and explain summary statistics for distributions of
  data including measures of center (mean, median) and spread (range,
  percentiles, variance, standard deviation).":
```

Standards omitted ...

```
> L4d2:="Explain how the relative frequency of a specified outcome of
  an event can be used to estimate the probability of the outcome.":
  L4d3:="Explain how the law of large numbers can be applied in simple
> examples.":
  L4d4:="Apply probability concepts such as conditional probability and
  independent events to calculate simple probabilities.":
> L4d5:="Apply probability concepts to practical situations to make
  informed decisions.":
```

16.4 Sample problems aligned with the various standards

It is a good idea when aligning problems with standards to have a section at the top of the source worksheet with the standards defined in an input cell. In this example, we have only put the standards that are cited, but the homework template worksheet should probably have all the standards just in case one is needed.

16.4.1 subtraction problem

This generator illustrates a way of putting the standards in: make the standard an argument to the generator. This way one can quickly switch from one set of standards to another.

```
> subtractprob:=proc(a,b,s)
  tagit(standards=s,["What is ",a," -
  ",b,"?"],_AS([a-b,b-a,a+b,a-b+1]));
  end:
```

Here we have aligned it with the Kentucky standards.

```
> subtractprob(5,2,[MAE112]);
```

```
Number 6


QM_[0.05;3][MA-E-1.1.2:The operations of addition, subtraction,
multiplication, and division]

AH_[0]

What is 5 - 2?

AS_[-3;3;4;7]

SKIP_
```

Usually, standards homework would be with multiplicity greater than 1, so in the same section with the generator, one would make a fixed number of additional calls to the generator. Here we have aligned this call with the ADP standards.

```
> subtractprob(7,3,[I1d1]);
```

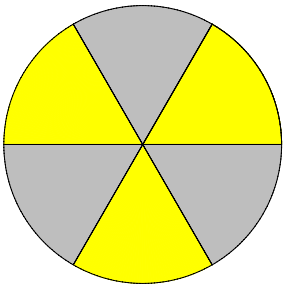
```
Number 7


QM_[0.05;4][Add, subtract, multiply and divide integers, fractions and
decimals.]

AH_[0]

What is 7 - 3?

AS_[10;-4;5;4]

SKIP_
```

16.4.2 oddeven problem

Here is a standards problem with a parameterized diagram. In this case, we have defined the diagram outside the generator. We could just as well have defined it inside the generator. However, since we defined and tested the diagram first, it made sense to leave it above the generator.

```
> pie := proc(center,numslices,clrs)
  plots[display](seq(plottools[pieslice]
  (center,1,i*2*Pi/numslices..(i+1)*2*Pi/numslices,
> color=clrs[i mod nops(clrs)
  +1]),i=0..numslices),scaling=constrained,axes=none)
  end:

> oddevenprob:=proc(n,clrs,s)
  local ans,distractor;
  if irem(n,2)=1 then ans := "odd":    distractor:= "even"
> else ans := "even":  distractor:="odd" fi:
  tagit(standards=s, pie([0,0],n,clrs),"The pie has an
  ",_AS([ans,distractor])," number of slices.",_TP());
  end:
> oddevenprob(6,[yellow,grey],[MAE113]);
```

```
QM_[0.05;even][MA-E-1.1.3:Odd and even numbers, composite and prime
numbers, multiples, and factors]

AH_[0]
```



```
The pie has an

AS_[even;odd]

 number of slices.

SKIP_
```

```
> oddevenprob(5,[wheat,magenta,turquoise],[MAE113]);
> oddevenprob(7,[yellow,pink,blue,red],[MAE113]);
```

16.4.3 Double reflection in parallel lines

Here is a problem which test's the students knowledge that composing two reflections about parallel lines is a translation in a direction perpendicular to the lines with magnitude twice the distance between the line. We are using some html pretags to italicize the names of the lines.

```
> m_ := "lt_i gt_ m lt_/i gt_":
  n_ := "lt_i gt_ n lt_/i gt_":
> tagit(standards=[MAH211],["Given:  ",m_," || ",n_,".    Point P is
  reflected about ",m_," onto image P' which is then reflected about
  ",n_," onto P''.  Then PP'' is "]
> ,_AW(["perpendicular","PERPENDICULAR","Perpendicular"]),[" to ",m_,".
  If PP'' = 16, what is the distance between the lines ",l_," and
  ",m_,"?"],_AS([8,4,16,32]));
```

```
QM_[0.05;perpendicular#PERPENDICULAR#Perpendicular;8][MA-H-2.1.1:Stude
nts will describe properties of and give examples of geometric
transformations and apply geometric transformations (translations,
rotations, reflections, dilations), with and without a coordinate
plane,.]

AH_[0]

Given: lt_i gt_ m lt_/i gt_ || lt_i gt_ n lt_/i gt_.  Point P is
reflected about lt_i gt_ m lt_/i gt_ onto image P' which is then
reflected about lt_i gt_ n lt_/i gt_ onto P''.  Then PP'' is

AW_[15]

AH_[0]

 to lt_i gt_ m lt_/i gt_. If PP'' = 16, what is the distance between
the lines l_ and lt_i gt_ m lt_/i gt_?

AS_[16;8;32;4]

SKIP_
```

16.4.4 numberline problem

This is an example of a problem with a diagram which is essential to the problem. The student must look at the picture and estimate the value of a number marked on the numberline. The MCtools words **PT** and **DL** are used to draw the numberline.

```
> mctools(PT);
```

```
MCtools, version Oct 15 2003.

PT(Location,txt,fontsize=16,clr=black)
```

```
> mctools(DL);
```

```
MCtools, version Oct 15 2003.

DL(A,B,thknss=2,
styl=1,clr=blue,leftshrinkfactor=0,rightshrinkfactor=0,hashnum=3,hashl
ength=.3,hashspacing=.1,hashlocation=.5)
```

The first word draws a numberline from a to b and puts the requested tickpoints in.

```
> numberline := proc(a,b,eps1,eps2,ticks)
  local ba,pba,A,B,len;
  len := b-a;
> A:=[a,0]:
  B:=[b,0]:
  ba := B-A;
```

```
> pba := [-ba[2],ba[1]]:
  plots[display](DL(A,B),
> seq(op([DL(A+(ticks[i]-a)/len*ba,A+((ticks[i]-a)/len+eps1)*ba,hashnum=
  1),
  PT(A+(ticks[i]-a)/len*ba+eps2*pba,ticks[i])]),i=1..nops(ticks)),
> scaling=constrained,axes=none);
  end:
```

The second word inserts the point into the diagram that the student will estimate.

```
> estimatepic := proc(a,b,eps1,eps2,ticks,x,rad)
  plots[display](PP([a,0]+x/(b-a)*([b-a,0]),rad),numberline(a,b,eps1,eps
  2,ticks));
  end:
```

Now the problem generator has the same arguments as estimatepic

```
> estimateprob :=proc(a,b,eps1,eps2,ticks,x,rad)
  tagit(standards=[MAM224],estimatepic(a,b,eps1,eps2,ticks,x,rad),"Estim
  ate the coordinate of the indicated point on the
> numberline.",_AS([x,x-.1*x,1.2*x,.5*x]))
  end:
> estimateprob(0,10,.01,.05,[0,2,3,4,9],2.6,.08);
```

```
QM_[0.05;2.6][MA-M-2.2.4:Estimate measurements in standard units]

AH_[0]
```


```
Estimate the coordinate of the indicated point on the numberline.

AS_[1.30;2.34;2.6;3.12]

SKIP_
```

```
> estimateprob(0,10,.01,.05,[0,1.5,2,3,4,9],3.6,.08);
> estimateprob(0,10,.01,.05,[0,1.5,6,7,8,9],6.2,.08);
```

16.4.5 Sample video problem

Problem sets aligned with a set of standards can and probably should contain hints of all kinds. In the next example, a video hint is supplied. Note: If the 5th argument of the homework header is 1, the hint can be put in the top directory of a CD or hard drive and accessed by selecting the appropriate drive letter. Otherwise, you must put the hint in a directory who name is the number

identifier of the homework. In either case, you can access it from the video server if you upload the hint to the server.

```
> angleprob := proc(a,b)
  tagit(standards=MAM226,["If two angle of a triangle are ",a," and
  ",b," degrees respectively, what is the third angle?"],
  _AC(180-(a+b),av_("shelby.wmv"))) end:
> angleprob(14,57);

QM_[0.05;109][MA-M-2.2.6:Estimate and determine measurement of angles]

AH_[0]

If two angle of a triangle are 14 and 57 degrees respectively, what is
the third angle?

AC_[5]

AH_[3;shelby.wmv;Press]

SKIP_
```

16.5 Collaboration teams for assessment tests

Developing sets of good, coherent, standards-referencing mathematics problems is a lot of work. However with a little organization it a task that is readily shared among a team of collaborators. By a **collaboration team** we mean any group of teachers who have author priviledges on WHS who have decided collectively to prepare an assessment test, or a set of course homeworks to post on WHS. The members of the team can be scattered all over the state or region; they communicate with email and WHS class uploads and downloads.

1. The group agrees precisely on standards relative to which they will develop a collection of diagnostic instruments. It also decides on the scope of the effort. The collection could range from a single problem set (e.g. a few problems referencing a single course standard - or to a complete set of instruments for entire course.

2. Once the standards and the scope of the project are selected the various instruments (individual problem sets) are usually developed sequentially - generally in the order in which they might be studied.

3. Individuals or teams are assigned the task of developing various parts of the instrument. Depending on the organization these smaller units may simply take sets of problems defined by the group (or other working subgroups) and render them in WHS form. At the other extreme they may have the full responsibility of taking a subset of the standards, producing the problems, and then the WHS form. What ever the path the final outcome of the smaller group is a WHS problem set which each problem is expected to be:

a. Complete, well-stated and unambiguous, properly illustrated (if appropriate), in proper English, mathematically correct, and fully tested

b. Contained in its own section which includes not only the problem but all of the code needed to produce the problem and information on which standards it references. The sectoned problem

should have a descriptive title which indicates the mathematical topic (e.g. "Solve two eqns in two unknowns." or the like).

Once each team assigned with part of the project has completed its work the problem then reduces to collecting the individual components in one place and assembling them into a single problem set. The assumption is that they are not all in the same geographic location and this has to be accomplished via the net.

The first thing to be done is that some member of the team is selected to be the **collator** . The collator will collect all of the problems and assemble them into a single document. This is easily done if you have the worksheets containing all the pieces open on the same computer. One simply closes the sections on the various worksheets, then copies them to a single worksheet.

The problem the is to get the individual problem sets to the collator. Although there are several ways to do this the following is simple, secure, systematic, and imposes the least on the collator. The process is:

a. The collator creates a WHS class for the project.

b. Each person on the team adds class, requesting registration.

c. The collator admits each of the people from part b to the class. This is done at Teacher Resources -> Access Records (for the class in question) -> Registration and Passwords

d. Each person with a worksheet for the collator goes to the Assignments screen

Assignments -> Uploads (for the class in question) and uploads the worksheet

Once everyone has done this the files are all in the Student Uploads for the class. The collator goes

Teacher Resources -> Access Records (for the class in question) -> Student Uploads. All of the files uploaded by the team members are arranged chronologically in a table which includes a link to the file and an email link to the person who uploaded it. The collator then downloads each of the files into a convenient directory.

a. The collator then opens a new, blank Maple worksheet, gives it the name of the compiled collection, and saves it in a diectory of the same name, as is customary. The then opens the directory containing each of the downloaded files and double clicks on one of them. This opens that worksheet in a new copy of Maple. The collator then copies each of the sections (problems) he/she wants to include in the compilation into the new worksheet and when done closes the upload just opened, opens another and does the same thing. When all of the desired problems have been saved the new, compiled worksheet is saved, exported to html, and installed under an appropriate name.

b. The collator makes the new worksheet available to the collaboration team by adding it to the class downloads.

Teacher Resources -> Access Records (for the class in question) -> Class Downloads

The new instrument is available for viewing as soon as it is installed. However being viewable is not the same as downloadable (which requires toggling the homework set to permit downloading). The above permits the collator to make the source for the new problem set available to the development team without making it available to the rest of the world.

Note: Collaboration classes can also be used to provide source worksheets to a select group.

The standard way to share worksheets in the past has been to toggle the download button on the homework in the Homework Installation table. However, that makes it available to anonymous people with author priviledges. Certain worksheet, such as future tests, you would not want to share in this fashion.

Exercise: Form a two person collaboration team

a. Everyone should clean his "standards" problem set from Wed. Section off the problems, include description of the standard addressed, make sure the answer is correct and problem well-stated, etc.

b. In turn each member of the group should become the collator, using his/her extant class or making a new one. Each of the other two members should register for the collator's class, the collator should admit them to the class, each of the "students" should upload their worksheets at the assignment screen in the class, the collator should save each download each of the "student" uploads into a directory and add his/her own to the directory. Then he/she should compile the three into a single set, install it on WHS, and then put the new worksheet in the class downloads for the class.

16.5.1 Pre-tests and post-tests

As we know from experience, not all students have the same knowledge of mathematics coming in to our course. It is good to have a **pre-test** which we could give at the beginning of the course to set a baseline.

16.5.2 Project

Make a post test for students coming out of middle school.

Make a pre-test for students coming into high school.

Index