1 Raison d'Maple

1.1 Four Properties of Maple

Here we intend to provide you with just enough information about the Maple language to give a headstart at using it productively in the problem solving process. The particular version of Maple that we are using as we describe it is Maple V, Release 4, running either in Windows 3.1 on DOS or Windows 95. You may be using an earlier release on a different platform, but most of what is in this document is still relevant.

Maple has at least four properties which make it very useful in problem solving.

It is an immediate mode language:

That means that you can type in commands and execute them, just like in the languages Basic and Logo. A Maple command is simply a string of characters ending in a semicolon ';' or colon ':'. For example, the command

```
2+5, factor(x^2+5*x +6),expand((x+y)^2);
```

$$7,\ (x+3)\,(x+2),\ x^2+2\,x\,y+y^2$$

tells Maple to do a sequence of three things: add 2 and 5, factor the quadratic, and expand the binomial. The strings `factor and expand are called Maple 'words'.` These are names of procedures which have been defined for performing the action (sometimes) suggested by the name on the stuff enclosed in parentheses just after the word. That stuff is usually called the 'input' of the word. The result of performing the procedure on the input is naturally called the 'output' of the word. If the command is terminated with a semicolon (colon), the output is displayed (not displayed).

It is a symbolic and graphical manipulator:

That means that it is built to work with algebraic expressions and draw pictures. There is a large vocabulary of Maple words, such as factor, simplify, and expand which are used to 'symbolically manipulate' expressions in the manner you are used to doing with pencil and paper. There are also a number of plotting words, which are used to draw graphs of functions of one or two variables, curves and surfaces. These drawings can be animated (ie displayed in sequence) to study change. Two of the plotting words most often used, `plot and` plot3d, are part of the regular Maple vocabulary. The rest, and there are several, are found in the the plots package, a sort of specialized vocabulary of words which is loaded separately.

It is an extensible language.

What this means is that you can define additional words and add them to the vocabulary. Initially, there will be little need to do this, except to define functions. The existing vocabulary is large enough to carry out the solution to many problems. After awhile, it becomes very useful to be able to add new words to the vocabulary. If you develop some words to work on a specialized class of problems, these can be put into a package of words for easy access between worksheets. Maple comes equipped with several such packages already, including plots, a package of drawing words, linalg - a very useful package of vector and matrix manipulation words, combinat - a package of words from combinatorics, and networks - a graph theory package.

It comes with a worksheet environment – see below.

We want to learn to use Maple to solve problems. In order make good use of the language for this purpose, we need to become familiar with the worksheet environnment, get to know the language, and foster our experimental urges.

1.2 The Worksheet: A handy place to solve problems.

When you click on the Maple Icon in Windows, an untitled worksheet opens up. Think of it as a clean sheet of paper. Typically, after awhile, the worksheet will contain a record of the work done to date on the problem or problems you have been working on. Very often, you might be in a problem-solving team working on the problems. The worksheet can be given a name and saved onto a disk for later working or for handing in. The worksheet file consists of a number of cells of three different types: Input cells, Output cells, and Text cells.

Input Cells:

These are started with a right angle bracket '>'. Here are a couple.

An input cell is the place where you put the 'commands' or 'statements' you want Maple to execute. The cell can contain one or more statements, each ending with a semicolon. The nice thing about these cells in a worksheet is that they can be modified and reexecuted over and over again. This enables you to correct typing mistakes with relative ease. For example, suppose I wanted Maple to calculate '3*(4+5*3)*(7+6);' but left out a parenthesis.

```
3*(4+5*3)*(7+6 ;
```

Syntax error, ';' unexpected

An error message is generated which may help you find your mistake. So you can make a change in that input cell and reexecute it. Use the mouse to put the cursor at the spot where the error occurs and make the correction

You can also use this ability to change and reexecute input cells to change the numbers in whatever problem you have worked out a solution to and see how the solution is affected.

Output Cells:

Almost every input cell, when executed, gives an output cell containing the results of the calculations. It is appended to the input cell which produced it. For example, let's add 3 to the 23rd and 4 to the 12th – then let's factor the result into prime factors.

```
s := 3^23 + 4^12;
ifactor(s);
```

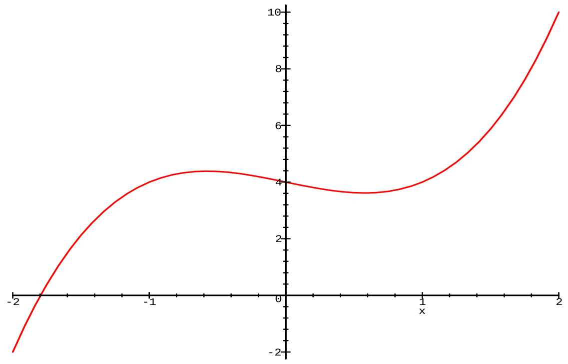
$$s := 94159956043$$

$$(727)\,(129518509)$$

You inspect the output cell to see if it is what you want. If it is not, then go back and change the input cell, and reexecute it.

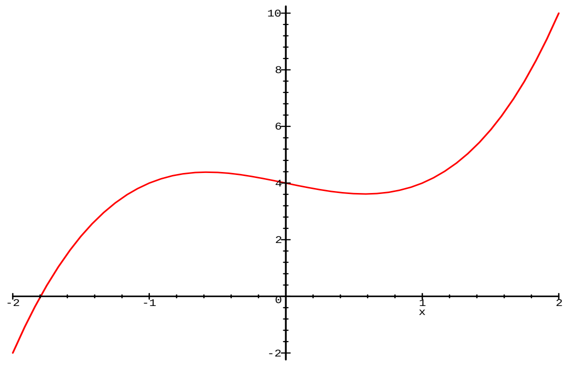
Certain Maple words such as plot generate a separate window containing a picture or a page of text (see factor in the next section). You can copy and paste these items into an output cell of your worksheet if the need arises.

```
plot(x^3-x+4, x=-2..2);
```



1.2.1 *Exercise: Execute the following plot command. Then copy the graph from the plot window (use the Edit Menu in that window) and paste it into your worksheet.*

```
plot(x^3-x+4,x=-2..2);
```



Text or Comment Cells:

A Text Cell is to record remarks and explanations of the solution to the problem you are working on. All of the comments here are typed into text cells. It is a good idea to be liberal with text cells. Undocumented calculations are for the most part worthless to anyone but the person who made the calculations, and even then the value is a rapidly decreasing function of time.

You have very good control over the cells in your worksheet with the Menus at the top of this worksheet. Depending on what platform you are using (Windows, X, or Mac), the menus may vary in their titles, but the effects are the same with minor differences:

You change an input cell to a text cell and back again.

You can split a cell (input or text) and join cells.
You open up new cells (input or text) between existing cells.
You can erase cells (input, text, or output) at your discretion.
You can copy cells (input or text) and paste them into another location.

These controls make it relatively easy to work away on a worksheet, doing computations, recording observations, explanations, etc and then go back and polish the worksheet up into a public document.

The worksheet does not replace pencil and paper or the chalk board in the problem solving process. When you are in the middle of analyzing a problem and deciding how to solve it, these tools are extremely useful. It is easier to draw a rough diagram by hand than by Maple. After you have decided on a plan and need to do some numerical calculation, some symbolic manipulation, or some plotting to carry out the plan, then a knowledge of the Maple language becomes useful.

1.3 Get to know the language

Maple has a large built in vocabulary of words especially defined to carry out many of the algorithms you have learned in your previous math classes. There are Maple words like factor , expand , simplify , etc. You can learn about them with online Help.

There is a Browser available in both X and Windows Maple which has the Maple vocabulary nicely indexed by category. Alternatively, you can ask for help in an input cell.

For example, to find out about factor just type

```
?factor
```

Knowing the word is one thing, but you also need to know the syntax of the word. What is syntax? Every algorithm requires certain input information in order to be performed. After it has been performed, certain output information is produced. To know the syntax of a Maple word is to know the input information needed and the output information produced by the word. The help screen gives you the syntax of the word, and thus tells you how to use it. For example, the help screen for factor tells us (in CALLING SEQUENCES) that there are one and possibly two inputs needed and one output produced by factor. The most useful part of the help screen is the bottom part, where there are examples of the usage of the word in question. These examples can be copied into a worksheet and tested out, which gives you a chance to develop a feel for the word by experimenting with its use. In fact, there is a Maple word example which brings up a page of examples of the usage of the word, in many cases.

1.3.1 Problems:

1.3.1 *Exercise: Discover the difference between factor and ifactor using ?ifactor .*

1.3.2 *Exercise: Use ifactor to factor your social security number.*

1.3.3 *Exercise: Use nextprime to find the first 30 primes after 5000.*

1.3.4 *Exercise: How do collect , coeff and expand work? Use them to expand* $(x + 2y)^4$ *, collect the result into a polynomial in x, and find the coefficient of the x^3 term of that result.*

1.4 Experiment!

Experimentation is a prime source of learning. We are constantly conducting little experiments, learning from them, and using that knowledge in some way. The same holds true when you are learning Maple and/or working on some math problem. The help facility is a great aid to experimentation. Say you are working on a math problem and you need to carry out some algorithm, like solve an equation you have set up. You know the Maple word solve will carry out the algorithm, but you have forgotten the syntax. You can use the word `example` to list some examples of its usage, rather than bring up the entire help sheet on solve .

```
example(solve);
```

Keep in mind that the help sheets are written for general use by both novices and experts, so don't be intimidated by unfamiliar terminology. Often the examples at the bottom of the sheet suffice to tell you what you need to know. Keep using help!

1.4.1 Problems

1.4.1 *Exercise: Spend a few minutes with the online help deciding what the terms list and set mean in Maple. In particular, what is the main difference between a list and a set?*

```
example(list);
example(set);
```

1.4.2 *Exercise: If you haven't already, find out the usage of the terms seq , NULL , subsop as used in the example sheet for set and list.*

Experiment is used at the beginning of solving a problem to generate data to conjecture a solution. Towards the end of the process, experiment is used to test the solution. Maple makes the act of experimentation easier to carry out and easier to modify. A Maple Worksheet makes it easier to carry on work from one session to the next, and to prepare public documents, ie, homework assignments, reseach papers, etc for public consumption.

2 A Short Introduction to the Maple Language

This section contains an introduction to some of the Maple vocabulary used for solving problems. It is not meant to 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.

2.1 Arithmetic

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 entering

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

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

is the same as entering

```
(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 typing and entering (pressing the enter key)

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

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

will get a return of 5/6. If you put a decimal point in one of the numbers, that forces Maple to return a decimal answer.

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

$$.8333333333$$

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

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

$$.8333333333$$

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

```
(3+2*I)*(2-I);
```

$$8 + I$$

will produce an output of 8+I .

The name for pi, the area of the circle of radius 1, in Maplese is Pi. So to calculate the area of a circle of radius 3, you would enter

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

$$9\,\pi$$

2.2 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.

2.3 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.

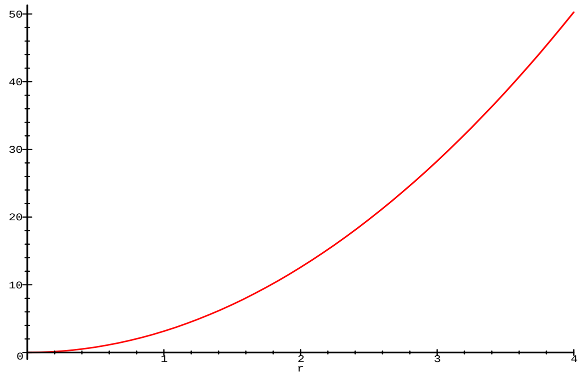
As an expression: The assignment

```
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;
```

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

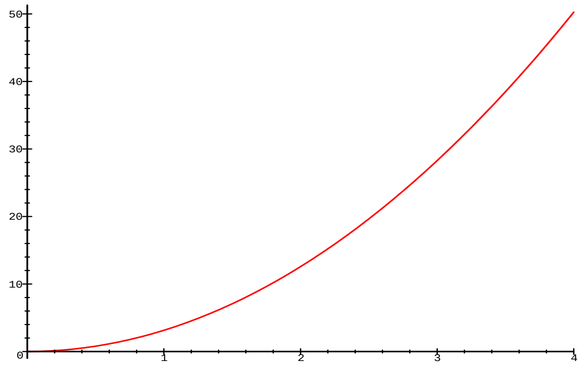
defines the area function also. Now to find the area of a circle of radius 3, we simply enter the statement

```
area(3);
```

$$9\,\pi$$

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

```
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;
```

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

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 '‘', 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;
```

$$area := \mathbf{proc}(r)\,\mathbf{if}\ r \le 0\ \mathbf{then}\ \mathrm{ERROR}(`\mathit{radius\ must\ be\ positive}`)\ \mathbf{else}\ \pi \times r^2\ \mathbf{fi}\ \mathbf{end}$$

```
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.

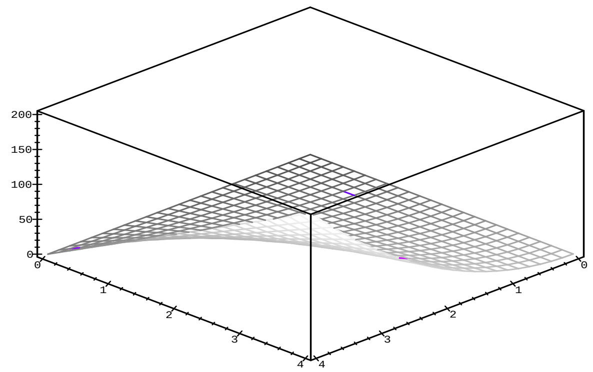
Functions of two variables can be defined and plotted just as easily in Maple as functions of one variable. For example, the volume V of a cylinder of height h and radius r is defined by

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

$$\mathrm{f}(x) := \begin{cases} x^3 + 8 & x \le -1 \\ 7 + 2\,x & x \le 2 \\ 11 - \cos(x) & x \le 4 \\ 3\,x & \mathit{otherwise} \end{cases}$$

```
f(2);
```

$$\mathrm{f}(2)$$

As it stands, f is not really a function. We need to use unapply to make it into a funtion.

```
g :=unapply(f(x),x);
```

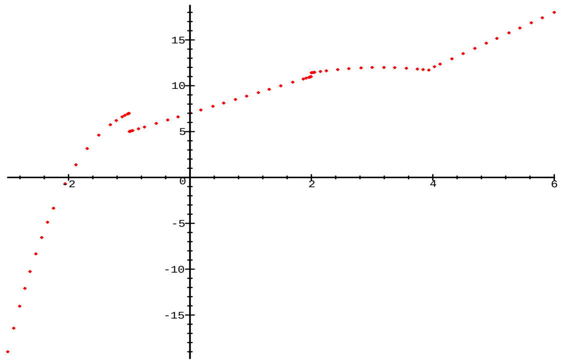
$$g := x \to \mathrm{piecewise}(x \le -1,\ x^3 + 8,\ x \le 2,\ 7 + 2\,x,\ x \le 4,\ 11 - \cos(x),\ 3\,x)$$

```
g(2);
```

$$11$$

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

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



2.4 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.);
```

$$.7472099577$$

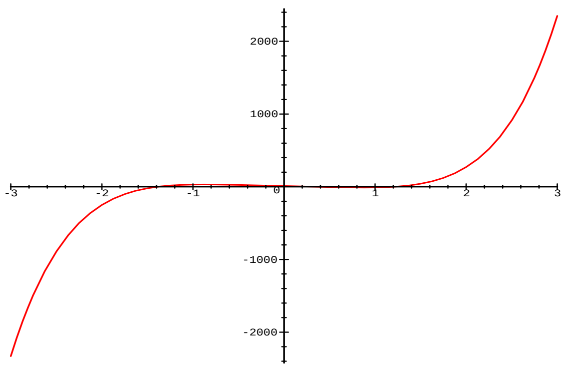
2.5 Using Maple as a fancy graphing calculator.

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 ;
```

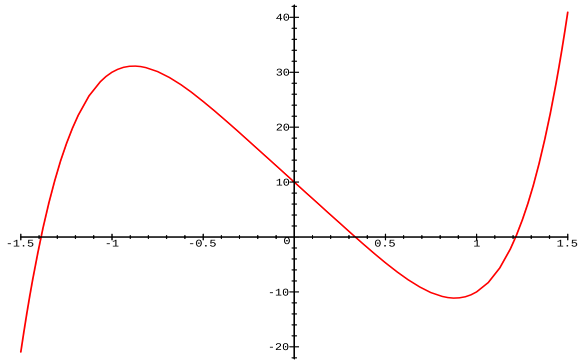
$$f := x \to 10\,x^5 - 30\,x + 10$$

```
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);
```

$$1.214648043$$

2.6 Data types, Expression Sequences, Lists, Sets, Arrays, Tables:

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);
```

$$fraction$$

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

$$+$$

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

$$=$$

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

$$exprseq$$

Expression Sequence.

An `exprseq, 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`;
```

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

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];
```

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

List.

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

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

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

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;
```

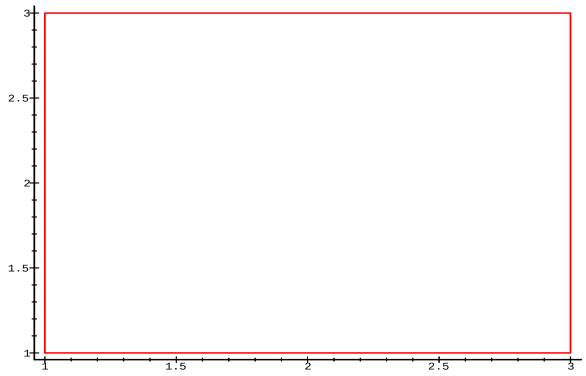
$$w := [-4,\ 6]$$

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);
```



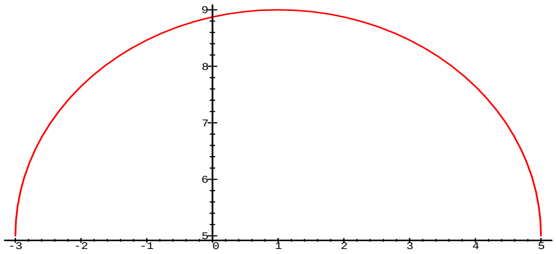
Notice in the graph that the origin is not included in the field of view. We can specify that by restricting the x and y coordinates.

```
plot(ab,x=0..4,y=0..4);
```

Error, (in plot) invalid arguments

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];
```

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

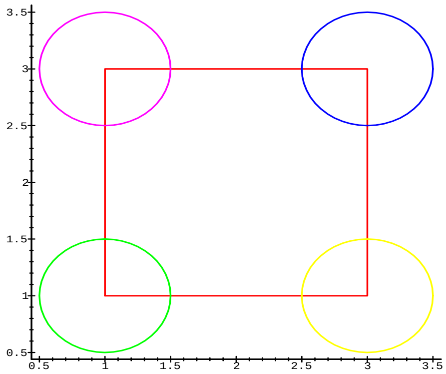
So if we wanted circles of radius 1/2 centered at the corners of the square ab we can construct the sequence of lists

```
circs := seq(circ(op(ab[i]),
1/2,0,2*Pi),i=1..4);
```

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

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);
```



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 doesn't work, however.

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

$$[1, 1], [3, 1], [3, 3], [1, 3], [1, 1]$$

Seq doesn't work well with a pure expression sequence as input. However, with some coaxing we can get it to do what we want.

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

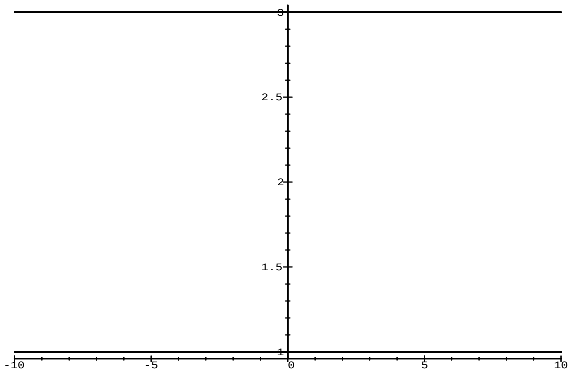
$$newab := [[1, 1], [3, 1], [3, 3], [1, 3], [1, 1]]$$

What did we do to change the input to seq ? We enclosed it in square brackets. If you feed such a list of points to plot, it knows what to do. If you wanted to strip out the inside brackets, that can be done too, but in release 4 of Maple, plot would treat it as a sequence of constant functions.

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

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

```
plot(newab,color=black);
```



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`};
```

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

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

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

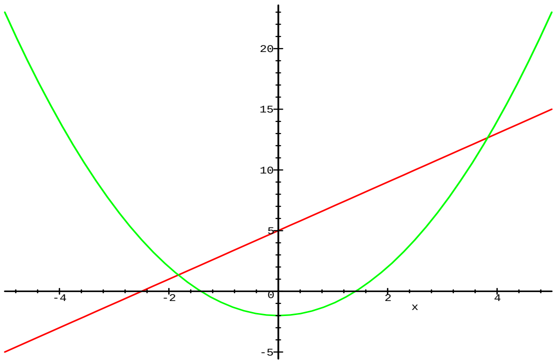
$$\mathit{Anotherset} := \{a,\ 1,\ 2,\ 3,\ 4,\ 7,\ \mathit{bill},\ \sin@\cos@(x \rightarrow x^2 + 3) + x + 1\}$$

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

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

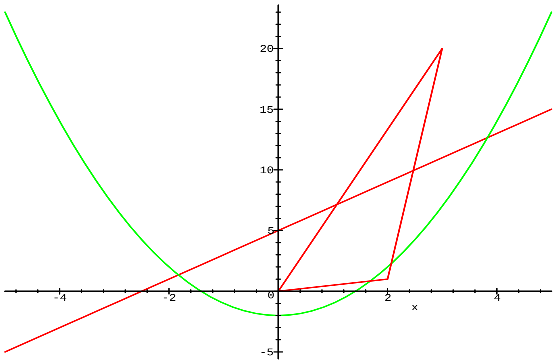
Sets are important when plotting more than one function at at time, to plot the quadratic function $x^2 - 2$ and the linear function $2\,x + 5$ on the same axes,

```
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]);
```



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.

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);
```

$$ang := .5410520681$$

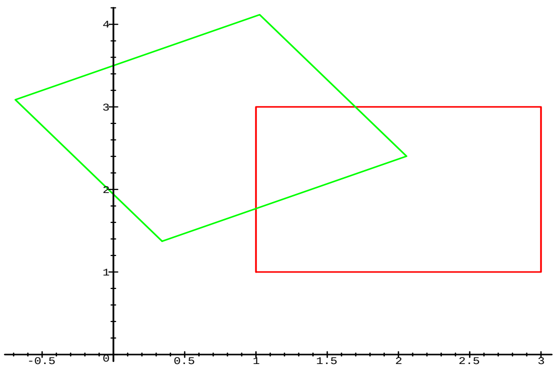
```
ab := [[1,1],[3,1],[3,3],[1,3],[1,1]];
```

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

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

$$rotab := [[.3421292258, 1.372205376], [2.056463827, 2.402281526], [1.026387677, 4.116616127], [-.6879469243, 3.086539977], [.3421292258, 1.372205376]]$$

```
plot({ [[0,0]],ab,rotab} );
```



2.7 Maple control statements

There are two especially important 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;

This statement can be used interactively or in a procedure to perform repetitive tasks or to do an iterative algorithm.

2.7.1 *Example: Add up the first 100 numbers.*

```
s := 0:  for i from 1 to 100 do s := s+i
od:
s;
```

$$5050$$

2.7.2 *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.;
```

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

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. Why have arrays at all? Well, for one thing, the terms in an array can be more easily modified. For example, to change the third term in aocube to 0 just enter $aocube_3 := 0$; . To change the third term in locube to 0, you have to make an entirely new list whose terms are all the same as locube except for the third one.

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

$$aocube_3 := 0$$

```
print(aocube);
```

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

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

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

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.

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

A solution:

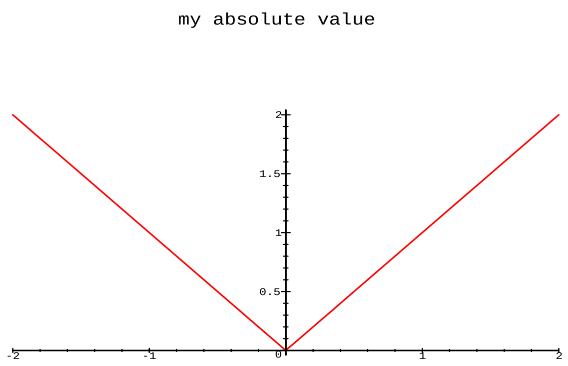
```
myabs := proc(x) if x > 0 then x else -x fi
end;
```

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

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

$$23$$

```
plot(myabs,-2..2,scaling=constrained,title=`m
y absolute value`); # to see what it looks like.
```



2.8 A Brief Vocabulary of Maple Words

Here are some Maple words useful in calculus 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.
```

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

```
collect(x*2 + 4*x,x); # collects like powers
of x.
```

$$6\,x$$

```
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
```

$$.001953125000$$

```
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.
```

$$4$$

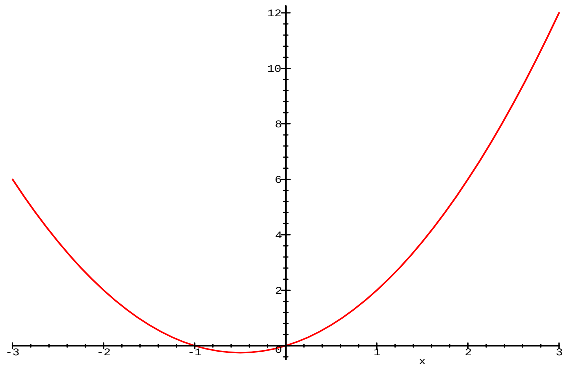
```
numer((a+b)/c); # gives numerator, here a+b
```

$$a + b$$

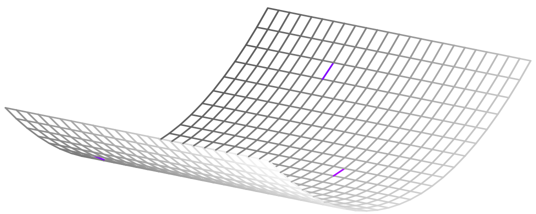
```
op([3,4,1,x]); # strips the brackets off the
list
```

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

```
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.
```

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

```
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
```

$$284$$

2.9 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.

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;
```

Syntax 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 ';' . Maple does nothing until it finds a semicolon. If you are getting no output when you should be, try feeding in a semicolon. This often works.

When in doubt, put in parentheses. For example, (x+3)/(x-3) is very different from x+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 can happen. One way to handle this is to keep an input cell of variables used.

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.

Are you using the correct quote symbol? In Maple, the forward quote ' is used to suppress evaluation. The back quote ` is used to enclose ascii strings. The double quote " is used to reference the last computation.

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.

Use debug. If you have defined a word, say 'something' 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.

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

3 Setting Up and Solving Problems

3.1 What is a problem?

For our purposes, a problem is anything that can be formulated as a question whose answer involves some mathematics. The main use of mathematics is to solve problems, and the best way to learn mathematics is to solve problems. This idea of a problem includes the 'skill' exercises that are always at the end of the sections in mathematics textbooks. For example, early in beginning algebra there is a section on solving linear equations, and at the end of that section there is a set of exercises consisting of lots of problems like 'solve $4\,x+3=2\,x-6$ for x'.

It also includes 'word problems' which can be solved using the methods mastered in the skill exercises. The word problems often are stated at the beginning of the section as motivation for the methods which are developed in the section. Problems usually arise in a context. Once the context is well understood, the problem can be formulated or posed and a method of solution worked out. From this solution, other problems may arise which require solving. We want to consider this problem identification and formulation as part of problem solving also.

The process of solving a problem is an active process, but can get bogged down for lack of knowing what to do next. So it is helpful to have a list of things to do. Here is one list of steps to carry out when you are solving a problem.

3.2 Setup – Solve – Interpret

SETUP the problem. This involves several steps.

Pose/Read the problem carefully, getting straight the meaning of all the terms used in the statement.

Draw a picture or diagram. This is a good way to focus thoughts, and gives you a place to put your labels.

Label or list the dimensions (or variables) important to the problem. Among these are the given dimensions, that is, the dimensions whose values are stated in the problem, the requested dimensions, that is, the dimensions whose values are requested in theproblem, and the intermediate dimensions, that is, the dimensions which arise in the process of trying to determine the requested dimensions from the given dimensions.

List or derive the equations relating the labeled dimensions.

SOLVE the equations we have set up for the required dimensions in terms of the given dimensions. This is one of places where Maple comes in very handy. It is easy to get bogged down in the calculations so that you lose all interest in solving the problem. This is less likely to happen if you have Maple at your disposal.

INTERPRET the obtained solutions. What are the realistic solutions given the context of the problem? Which should be ignored? Have all solutions been obtained?

Where does Maple come in handy in this process? In the Solve phase, mostly. The actual setting up of the problem as a mathematical problem has to be done by you. Regard Maple as a tool to carry out and record the solution you imagine. A very important word in the Maple vocabulary is `solve.` This word is used to solve a system of one or more equations which come up.

3.3 A Swimming Pool Problem:

3.3.1 *Problem: A swimming pool is three times as long as it is wide. It is also 40 feet longer than it is wide. Find its dimensions.*

Solution: Let l and w be the length and width of the pool. Then the first statement of the problem translates to the equation $l = 3w$, and the second statement to $l = w + 40$. We need to solve these two equations simultaneously for l and w. First, set up the equations.

```
eq1 := l = 3*w;
```

$$eq1 := l = 3\,w$$

```
eq2 := l = w + 40;
```

$$eq2 := l = w + 40$$

Then solve the system for l and w. We can do this by subtracting eq2 from eq1 and solving for w, getting $w = 20$, then substituting that value for w into eq1 getting $l = 60$.

```
eq3 := eq1 - eq2;
```

$$eq3 := 0 = 2\,w - 40$$

```
eq4 := lhs(eq3) - 2*w = rhs(eq3) - 2*w;
```

$$eq4 := -2\,w = -40$$

```
eq5 := -(1/2)*eq4;
```

$$eq5 := w = 20$$

```
eq6 := subs(eq5,eq1);
```

$$eq6 := l = 60$$

```
solution := {eq5,eq6};
```

$$solution := \{w = 20,\, l = 60\}$$

The word `solve` carries out this algorithm automatically.

```
solve({eq1,eq2},{l,w});
```

$$\{w = 20,\, l = 60\}$$

The pool is 20 feet wide and 60 feet long.

The word solve is a very good to know, but it is not infallible. It uses some methods of solving equations which you know and some which you probably don't know. Don't give up just because solve doesn't give you a solution.

3.4 Four methods of solving equations

So it is important to also be able to solve equations by various means. Here are four.

Guess and check. This method involves guestimating a solution somehow and checking your accuracy somehow. For example, suppose you had guestimated that the dimensions of the pool are about 25 by 55 feet and wanted to check that. Use the word `subs`.

```
subs({w=25,l=75},[eq1,eq2]);
```

This line says to substitute 25 and 75 for w and l in eq1 and eq2.

(Notice the use of braces and brackets here. Braces are used to enclose the members of a set and brackets are used to enclose the members of a list. In a set, the order is not important and repetitions are not counted; in a list, the order is important and repetitions are counted.) As we can clearly see, our guestimate is off. We have satisfied the first equation, but not the second. If we decrease the value of w by 1, then we have to decrease the value of l by 3 in order to continue to satisfy the first equation. Will we come closer to satisfying the second? Let's see.

```
subs({w=24,l=72},[eq1,eq2]);
```

$$[75 = 75,\ 75 = 65]$$

$$[72 = 72,\ 72 = 64]$$

Well, yes, if ever so slightly. The left-hand side of the second equation has decreased by 2 towards the right-hand side. We could pursue this method of guessing and then trying to improve our guess, but let's put that off until later.

By Hand, as with pencil and paper.

You can choose to try to solve your equations by hand by which I mean to manipulate the equations the same way you would do using pencil and paper. This can be done without using solve at all. This is what we did in our original solution to the swimming pool problem. On the other hand, we could simplfy the equations and then use the word solve. For example, let's solve the two equations, eq1 and eq2, by hand. Using the Maple word solve we need only solve one equation for one unknown. Here is a possible way to proceed:

```
eq3 := subs(l=solve(eq1,l),eq2);
```

$$eq3 := 3\,w = w + 40$$

```
sol1 := w = solve(eq3,w);
```

$$sol1 := w = 20$$

```
sol2 := l = solve(subs(sol1,eq1),l);
```

$$sol2 := l = 60$$

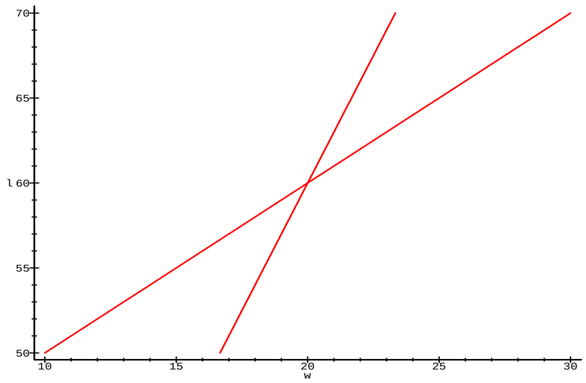
```
sol := {sol1, sol2};
```

$$sol := \{w = 20,\ l = 60\}$$

Graphical solution.

Another way to solve equations is graphically. Here we can plot each equation, using plot or implicitplot and use the pointer to locate the approximate solutions. These solution(s) are the located where the graphs of the equations coincide. Note: the word implicitplot can only be found in Maple V Release 2. Earlier versions of Maple only plot functions. This method also works very well on a graphing calculator.

```
with(plots):
implicitplot({eq1,eq2},w=10..30,l=50..70);
```



Using fsolve.

Another way to solve equations is with fsolve . This word employs methods of calculus to find floating point approximations to the equation you are trying to solve. If you do not supply fsolve with an interval in which to search for a solution, then it returns the first solution it finds. This word works well in conjunction with plot . You can use plot to narrow down the search interval, and then use fsolve to get the 'exact' answer.

3.5 Problems

3.5.1 *A. Solve the following systems of equations.*

1. $x^2 + y^2 = 10$, $y = 3\,x$

2. $a + 3\,b + c = 10,\ 2\,a - b + 72\,c = 20,\ 7\,a - 3\,b + 21\,c = 30$.

3. $x^5 - 5\,x^2 + 2 = 0$ use plot and fsolve here.

4. $x - 3\,y + z = a,\ 2\,x + y - 3\,z = b,\ 9\,x + 3\,y + 5\,z = c$ for $x,\ y$ and z.

3.5.2 *B. Solve the following word problems by setting up and solving a system of equations.*

1. The height of the Eiffel Tower in Paris is 125 feet less than twice the height of the Washington Monument. The latter is 75 higher than the Great Pyramid of Cheops and 105 feet higher than the dome of St. Peter's Church in Rome. If the sum of the heights of these four edifices is 2510 feet, find the height of each to the nearest foot.

2. A large conference table is to be constructed in the shape of a rectangle with two semicircles at the ends (see figure). Find the dimensions of the table, given that the area of the rectangular portion is to be twice the sum of the areas of the circular ends, and the perimeter of the whole table is to be 40 feet.

Here is a diagram to accompany the problem.

```
F:=plot([-cos(t)+1,sin(t)+1,t=Pi/2..-Pi/2]):
```

```
G:=plot([cos(t)+6,sin(t)+1,t=Pi/2..-Pi/2]):
```

```
H:=plot([[1,0],[6,0],[6,2],[1,2],[1,0]],style
=LINE):
```

```
plots[display]({F,G,H},axes=none,scaling=co
nstrained);
```

3. A local hardware store worker is making up a fertilizer mix from some left over fertilizer from the summer. There are three types, with 30 percent, 20 percent and 15 percent nitrogen content respectively. When he mixed all the fertilizers together and tested the nitrogen content, he found that the mix weighed 600 pounds and contained 25 percent nitrogen content. How many pounds of each type did he have left over? How many solutions does this problem have?

4. A 30 foot ladder and a 40 foot ladder are positioned so as to cross each other in an alley. That is, they are leaning up against opposite walls with their bases snug up against the base of the opposite

walls. Given that they cross at a point 10 feet above the floor of the alley, determine the width of the alley.

5. Make up your own algebra problem to set up and solve using Maple. It can be a variation on one of the problems above, or it can be something entirely different.

3.6 More About Plotting

We have already used a few words from the `with(plots); package.` When you want to learn more about any package (in this case with(plots);) start by looking at the commands in the package. Do this by typing

```
with(plots);
```

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

We have already used plot , implicitplot , and display (in problem 2 of part B). To learn more about these very useful commands simply type

```
?plot
?implicitplot
?display
?textplot
?animate
```

or anything else that might interest you. Look at the bottom of the help files for examples. Sometimes, you are your best teacher. Learn by experimenting with a few (or all) of these commands. Also, you will learn more about these words in worksheets to come.

3.7 Putting in a parameter.

One of the advantages of solving a problem in a Maple worksheet is that it gives the capability of going back and changing the numbers in the problem to study how the solution changes. In fact, you are led to the practice of putting a parameter into the problem. For example, to put a parameter in the swimming pool problem, we have two natural choices: Replace the number 3 or the number 40 by a parameter. Let's replace 3 with p.

Parameterized Swimming Pool Problem

3.7.1 Problem: *swimming pool is p times as long as it is wide. It is also 40 feet longer than it is wide. What are its dimensions in terms of p? Describe how the dimensions vary with p.*

Solution: We could just go back to the cell containing the original equations and put in a p for 3 in eq1.

```
restart;
```

The word `restart` clears all variables by rebooting Maple.

```
p := 'p';
```

$$p := p$$

```
eq1 := l = p*w;
```

$$eq1 := l = p\,w$$

```
eq2 := l = w + 40;
```

$$eq2 := l = w + 40$$

Now we would solve for l and w, just as before, except now the solution is given in terms of the parameter p.

```
sol := solve({eq1,eq2},{l,w});
```

$$sol := \{w = \frac{40}{-1+p},\ l = 40\,\frac{p}{-1+p}\}$$

Now by inspection, we can see that as p gets large, both w and l get small. Also, as p approaches 1 from the left, l and w get large. In more complicated situations, we could use plot to study how the solution changes as the parameter changes.

```
plots[animate]({p*w,w+40},w=0..80,p=0..2);
```

The word `animate,` which is used out of the with(plots); package, is very useful in seeing the range of possible solutions for this problem. Press your pointer on play to see the animation.

3.8 Problems

3.8.1 *1. What happens to the solution to the swimming pool problem as the difference of the width and length is allowed to vary? (Keep the ratio of the width to the length constant.)*

3.8.2 *2. A large conference table is to be constructed in the shape of a rectangle with two semicircles at the ends (see figure). Find the dimensions of the table, given that the area of the rectangular portion is to be p times the sum of the areas of the circular ends, and the perimeter of the whole table is to be 40 feet.*

3.8.3 *3. Study the solution to the mixture problem, no.3, in the previous problem set. Vary a parameter of your choice and describe how the solution changes.*

3.8.4 *4. A 30 foot ladder and a 40 foot ladder are positioned so as to cross each other in an alley. That is, they are leaning up against opposite walls with their bases snug up against the base of the opposite walls. Given that they cross at a point h feet above the floor of the alley, determine the width of the alley.*

3.9 Defining your own Maple words.

This is a good place to learn how to develop and define Maple words in a worksheet. The idea is very simple:

A. Name the given quantities –

As you are solving a problem or developing an algorithm, assign the given quantities to appropriate names which you have chosen. Put these assignments into a single input cell, which we will call the parameter cell.

B. Compute the desired quantity –

Use the names assigned in your maple statements which you make in your algorithm. This part will change and grow as you develop the definition, but try to keep all the statements in one input cell, which we will call the definition cell. Once you get the desired output,

C. Make the Definition –

This involves mostly choosing a name for the procedure, inserting it at the top of the definition cell in the line

name := proc(p1,p2,...,pn)

which begins the definition by assigning the name and declaring the input parameters p1, p2, etc.

At the bottom of the definition cell is the word

end;

which signals the end of the definition.

Now, when you execute the definition cell, you should get a nicely formatted version of the definition as output.

This is not the only set of steps you could follow when developing a definition, but it is very natural one and works well for small definitions.

By way of example, suppose we wanted to define a word swim which would return the solution to the swimming pool problem above.

Copy all of the input cells used to obtain the solution into one input cell.

Then all we need to do to define the word is to insert a proc line at the top and an end at the bottom.

When inserting the proc line, you must decide what the inputs for the word will be. We will let the ratio of the length of the pool to the width of the pool be the only input in this word. We were calling that p in the equations, so use the same name in the proc line.

```
swim := proc(p)
```

```
eq1 := l = p*w;
eq2 := l = w + 40;
sol := solve({eq1,eq2},{l,w});
end;
```

Warning, ‘eq1‘ is implicitly declared local
Warning, ‘eq2‘ is implicitly declared local
Warning, ‘sol‘ is implicitly declared local

$$
\begin{array}{l}
\mathit{swim} := \mathbf{proc}(p) \\
\quad \mathbf{local} \mathit{eq1},\ \mathit{eq2},\ \mathit{sol}; \\
\qquad \mathit{eq1} := l = p \times w\,;\ \mathit{eq2} := l = w + 40\,;\ \mathit{sol} := \mathrm{solve}(\{\mathit{eq1},\ \mathit{eq2}\},\ \{w,\ l\}) \\
\quad \mathbf{end}
\end{array}
$$

When you execute the input cell containing the definition, the word swim has been added to the vocabulary and can be used like any other Maple word. The assignments made in the definition are declared local and a warning is issued unless you declare the names either local or global. Usually, you will want any assignments to be local, just in case you are using the same name outside the word to mean something else.

The output from a word is the output from the last line before the end line. So, for example, if the pool is 3 times as long as it is wide then p = 3 and we would say

```
swim(3);
```

$$\{l = 60,\ w = 20\}$$

On the other hand, if we put in a ridiculous input what would happen?

```
swim(-4);
```

$$\{w = -8,\ l = 32\}$$

You can redefine the word if there is something that needs changed. For example, the swimming pool problem doesn't really have a solution if p is not positive, so we could insert an error trap here.

```
swim := proc(p)
if not type(p,name) and not p > 0 then
ERROR(‘oops‘)
else eq1 := l = p*w;
eq2 := l = w + 40;
sol :=
solve({eq1,eq2},{l,w});
fi
end;
```

Warning, ‘eq1‘ is implicitly declared local
Warning, ‘eq2‘ is implicitly declared local

Warning, ‘sol‘ is implicitly declared local

$$
\begin{aligned}
&swim := \mathbf{proc}(p)\\
&\quad \mathbf{local}\, eq1,\ eq2,\ sol;\\
&\quad\quad \mathbf{if}\ \ \mathbf{not}\ (\mathrm{type}(p,\ name)\ \mathbf{or}\ 0 < p)\ \mathbf{then}\ \mathrm{ERROR}(oops)\\
&\quad\quad \mathbf{else}\ eq1\ := l = p \times w\ ;\ eq2\ := l = w + 40\ ;\ sol := \mathrm{solve}(\{eq1,\ eq2\},\ \{w,\ l\})\\
&\quad\quad \mathbf{fi}\\
&\quad \mathbf{end}
\end{aligned}
$$

Now redefine swim and check –

```
swim(-4);
```

Error, (in swim) oops

Usually, one doesn’t spend a great deal of time inserting error traps in word definitionst. There are much more interesting things to do with the mathematics of the situation. For example, suppose we wanted to change the inputs to allow for changing the 40, the amount the length exceeds the width, that occurs in the problem. Then simply copy down the definition into a new input cell and make the appropriate changes. Something like this will work:

```
restart;
```

```
swim := proc (p,excess)
local eq1, eq2, sol;
eq1 := l = p*w;
eq2 := l = w+excess;
sol := solve({eq1, eq2},{l, w})
end;
```

$$
\begin{aligned}
&swim := \mathbf{proc}(p,\ excess)\\
&\quad \mathbf{local}\, eq1,\ eq2,\ sol;\\
&\quad\quad eq1\ := l = p \times w\ ;\ eq2\ := l = w + excess\ ;\ sol := \mathrm{solve}(\{eq1,\ eq2\},\ \{l,\ w\})\\
&\quad \mathbf{end}
\end{aligned}
$$

```
swim(3,40);
```

$$\{w = 20,\ l = 60\}$$

```
swim(3,ex);
```

$$\{l = \frac{3}{2}\, ex,\ w = \frac{1}{2}\, ex\}$$

Notice that you can ’read off’ the solution in words if you put in a variable for the excess. So, if the pool is 3 times as long as it is wide, then the width must be 1/2 of the excess and the length is 3/2 of the excess.

Visual Checking of answers.

You can also define a word to draw a picture of the pool. These types of words are very useful to perform a visual check of computations. Sometimes, you have solved a problem incorrectly, but do not discover that until you have performed a visual check on the solution you have obtained.

To develop a visual check of our swimming pool problem, we can first draw one picture. Get the dimensions of a pool

```
dims := swim(3,40);
```

$$dims := \{w = 20, l = 60\}$$

Now we want to draw a rectangle which is 20 by 60. We can set up a general rectangle with one corner at the origin and the opposite corner at [l,w]

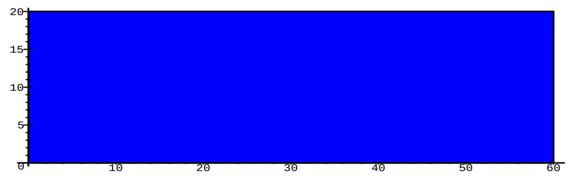
```
rect := [[0,0],[l,0],[l,w],[0,w]];
```

$$rect := [[0, 0], [l, 0], [l, w], [0, w]]$$

```
pool := subs(dims,rect);
```

$$pool := [[0, 0], [60, 0], [60, 20], [0, 20]]$$

```
plots[polygonplot](pool,
color=blue,style=patch, scaling = constrained);
```



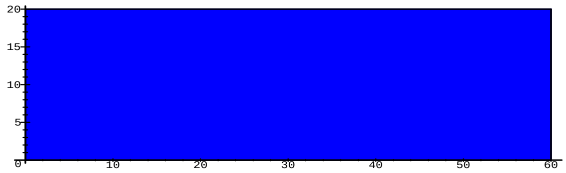
Now we can define the word. Copy down and join the appropriate input cells. Then insert a proc line at the top (deciding on what the inputs will be), and an end line at the bottom of the new input cell.

```
drawpool := proc(p,excess)
dims := swim(p,excess);
rect := [[0,0],[l,0],[l,w],[0,w]];

pool := subs(dims,rect);

plots[polygonplot](pool,
color=blue,style=patch, scaling = constrained);
end;
```

Warning, ‘dims‘ is implicitly declared local
Warning, ‘rect‘ is implicitly declared local
Warning, ‘pool‘ is implicitly declared local

$$
\begin{array}{l}
\boldsymbol{drawpool} := \mathbf{proc}(\boldsymbol{p},\ \boldsymbol{excess}) \\
\quad \mathbf{local}\boldsymbol{dims},\ \boldsymbol{rect},\ \boldsymbol{pool}; \\
\quad\quad \boldsymbol{dims} := \mathbf{swim}(\boldsymbol{p},\ \boldsymbol{excess})\,; \\
\quad\quad \boldsymbol{rect} := [[0,\ 0],\ [\boldsymbol{l},\ 0],\ [\boldsymbol{l},\ \boldsymbol{w}],\ [0,\ \boldsymbol{w}]]\,; \\
\quad\quad \boldsymbol{pool} := \mathbf{subs}(\boldsymbol{dims},\ \boldsymbol{rect})\,; \\
\quad\quad \boldsymbol{plots}_{\boldsymbol{polygonplot}}(\boldsymbol{pool},\ \boldsymbol{color} = \boldsymbol{blue},\ \boldsymbol{style} = \boldsymbol{patch},\ \boldsymbol{scaling} = \boldsymbol{constrained}) \\
\quad \mathbf{end}
\end{array}
$$

```
drawpool(3,40);
```



Now we can make an animation of the how the pool dimensions change as we vary the excess and/or the ratio of length to width. Make a list of the outputs from drawpool (call it movie, say) and then use plots[display] with the option `insequence = true.`

```
movie := [seq(drawpool(3,10*i),i=1..8)]:
plots[display](movie,insequence=true);
```

This movie merely shows the dimensions of the pool increasing linearly with the excess, as we already knew, so not much is gained from observing the animation. Often, however, one can learn a lot from animations, if some care is taken with their construction.

3.10 Problems

3.10.1 *Exercise: Make a movie of the change in the pool as the ratio p changes from 1 to 4 by increments of 1/2.*

3.10.2 *Exercise: What is wrong with taking the original word swim and changing to proc line to read swim := proc(p,w) ?*

3.10.3 *Exercise: Modify drawpool so that if the ratio p is greater than 2, the color of the pool is blue, otherwise the color is red.*

3.10.4 *Exercise: Define a word quadform to take three numbers a, b, c and return the roots of the quadratic equation* $\boldsymbol{a\,x^2 + b\,x + c = 0}$*, if they are real, and the message 'no real roots' if they are complex. Don't just use solve here .*

3.10.5 *Exercise: Modify the word swim so that it simply returns the point [l,w] rather than the solution to the equations.*

3.10.6 *Exercise: Define a word which takes a function and tabulates its value over a given interval with a given increment between x values.*

Index