# TI Math Games
## Lesson Plan

**Cube Fellow:** Deric Miller
**Teacher Mentor:** Dale Adkins
**Grade/Class:** 8th Grade Math
**KY Standards:**
**MA-08-1.1.3, MA-08-4.1.1, MA-08-4.1.4, MA-08-5.3.1**
**Objectives:**

Students will gain an introductory familiarity with programming theory.

Students will learn about and generate flow diagrams.

Students will learn how to enter programs into their classroom calculators.

Students will enter and use a math quiz drill program. They will both acquire full understanding of every line of code comprising the program, and use the program to hone their arithmetic skills.

## Resources/Materials needed:

One TI calculator (TI-73 or better) for each class member or group

A copy of the handout, included below, for every student

## Motivation:

I find that many students simultaneously claim to dislike math, but enjoy games of many various types. This lesson seeks to emphasize that games represent mathematical constructs. I hope that by offering an introduction to computer programming, I can inspire students to generate their own programs both for entertainment and practical applications in the future. The lesson also serves to install effective arithmetic drill instructional software on every calculator in the classroom.

## Prior Knowledge:

The lesson assumes that students have been using TI calculators in the classroom, but assumes no prior programming experience whatsoever.

## Lesson Source:

This lesson plan, including the programs and flow diagrams contained herein, is all original work.

## Mode of Instruction:

Interactive lecture, followed by group work on a worksheet, and competitive gaming

## Estimated Time:

This lesson fills one class period, eighty minutes in length.

## Date Submitted:

5/13/9

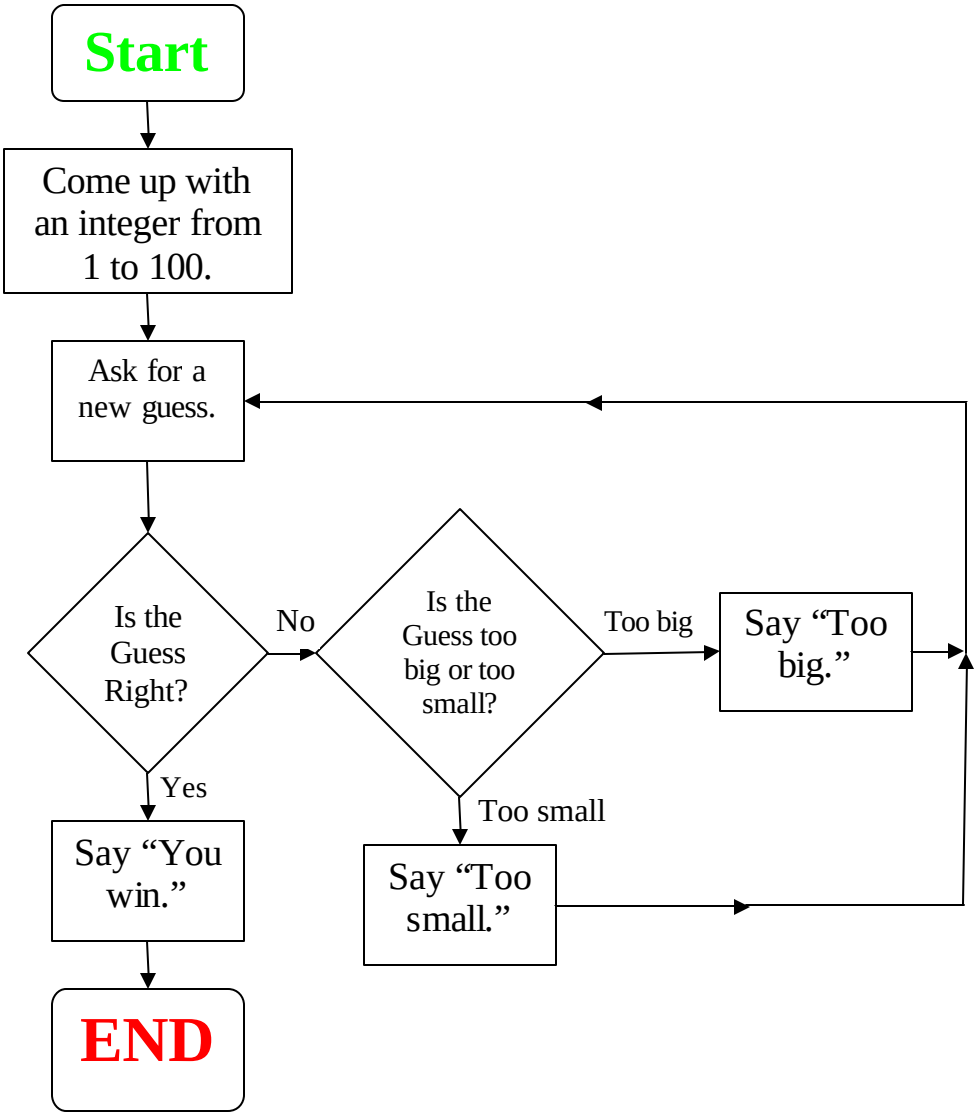
# Lesson Plan:

*In order to avoid distractions during the theoretical portion of the lesson, do not allow the students to have the calculators until the programming section of the lesson.*

Today we're going to learn how to program our calculators to play games. Because we only have one day for this, the games that we program today will be simple, but hopefully this will help you get started, and you'll be able to use these skills to write more complex games in the future. So, I need a volunteer to come play a game with me. We'll play the game that we're going to teach the calculator.

*Bring up a student and play with them the "pick a number between 1 and 100" game, with them guessing your secret number, and you giving "too high" and "too low" guidance.*

*Talk about the best strategy to win the game, and draw the following simple flowchart on board for giving the "too high"/"too low" responses, explaining that programmers use flowcharts to think about how to get their programs to work before they actually type them into a computer/calculator. Start drawing with the Start box, and continue in the order of the program flow, explaining each new step as you go. When you come to the first choice, "Is the guess right?" explain that diamonds indicate choices, and which arrow to follow depends on the answer to the choice question.*




*Play again with a new volunteer, going through the flowchart as you go to show that the flowchart covers all possible situations.*

*Pass out the calculators. Give every student one if possible; otherwise get the students to work in pairs, sharing a calculator with their partner. As you go over what to do, use the TI emulator on the smart board so that the students can see you doing everything things as they go.*

So we're going to write a computer program to do everything that I just did in playing the game. It will pick a number, take guesses, and give feedback on whether the guesses were too high or to low, or correct.

A computer, or calculator, program is series of commands executed by calculator in order. The computer will do exactly what you tell it to do, so you have to be very precise in what you tell it to do.

So first, we make a new program. From the home screen, press the PRGM button. Then press the right arrow twice, so that you've highlighted the command New, then press the ENTER

button. Hit 2nd, then Text, then use the arrows and the ENTER button to enter in a name for your new program. Call this program "HIGHLOW"; select DONE and press enter after entering your name.

The first thing we have to tell the calculator to do according to the flowchart is tell the calculator to pick a random number from 1 to 100. On the program editing screen, press the Math button, then use the right arrow button to select PRB, then the down arrow button to select randInt(. The randInt() command outputs a random integer between two integers that you give it. For instance randInt(1,6) behaves like a standard six-sided die, giving you a number from 1 to 6. We want a number from 1 to 100, so, after hitting ENTER on randInt( in the menu, type "1,100)" so that the line reads:

**:randInt(1,100)**

Now, it's not enough to just make the random number, we need to put it somewhere so that later we can compare it to the guesses. So, we're going to make a variable, and put our random number in it. With your cursor just after the ")" on the line, press the "STO? " button, then press 2nd, TEXT, and then enter a letter for the variable name, select DONE, then press ENTER. Let's call the variable N for number. The full line should read:

**:randInt(1,100)** ? N

Press ENTER to begin a new line.

This generates a random number from 1 to 100, then stores that number in the variable N. The next step in our flowchart tells us to ask for a guess. To output text to the screen, we use the Disp command. Press PRGM, then use the arrow buttons, first to select the I/O heading, then, under I/O, Disp. Press ENTER. Then press 2nd, TEXT, and use the arrow buttons and the ENTER button to type "GUESS" (including the quotation marks). Select DONE, then press ENTER.

Now that we've set up the program to ask the player for their guess, we need to get that guess from them and keep it in a variable of its own. Press PRGM, then use the arrow buttons to select Input from the I/O menu, and press ENTER. Now we name the variable where we will store the player's guess G for Guess. Press 2nd, TEXT, use the arrows to select G, then press ENTER, select DONE, and press ENTER again.

The full program should now read:

**:randInt(1,100)** ? N
**:Disp "Guess"**
**:Input G**

We have now coded the first three boxes of our flowchart into the calculator. The next box represents our first choice, "Is the guess right?" What we do next depends on whether or not the player guessed the secret number correctly. To find out if they guessed correctly, we compare our variable storing their guess, G to our variable storing the secret number, N, as follows. Press the PRGM button, then select If from the CTL menu, and press ENTER. Press 2nd, TEXT, and then use the arrow buttons and the enter key to enter G=N, then select DONE and press ENTER.

The full program should now read:

**:randInt(1,100)** ? N
**:Disp "Guess"**
**:Input G**
**:If G=N**

Press ENTER to start a new line, then press PRGM, and use the arrow buttons to select Goto from the CTL menu. Then go to the text entry screen and add a B, such that the full program now reads:

**:randInt(1,100)** ? N
**:Disp "GUESS"**
**:Input G**
**:If G=N**
**:Goto B**

The if command only executes the next line if the test is true. If G is equal to N, then the program will execute the next line. If not, it will skip that line and move on. Goto B will make the program jump to Lbl B, which we will put in the program later. The program only goes to Lbl B if G=N, so Lbl B will include everything from the flowchart from the Yes line out of the "Is the guess right?" diamond down through the end. But before we get to that, we have to write code for what happens if G does not equal N. The No line out of the "Is the guess right?" diamond takes us to the choice "Is the guess too big?" To code this line, we check to see if G is greater than N, and, if it is, we tell the user that their guess is too big. These lines, coded, follow:

**:If G>N**
**:Disp "TOO BIG"**

A similar pair of lines tells the user if their guess was too small:

**:If G<N**
**:Disp "TOO SMALL"**

Now we need for the program to loop back to a point just before we asked for a new guess, and after the point where we picked our random number. We can do this by adding the line

**:Goto A**

To the program, but for this to work, we need to go back up and insert the line

**:Lbl A**

just after picking the random number. To do this, use the arrow buttons to put the cursor at the end of the first line of the program, then press 2nd, INS, ENTER. This makes a new blank line. Now press the PRGM button, use the arrow buttons to select Lbl from the CTL menu, press ENTER. Now press 2nd, TEXT, then use the arrow keys to select the letter A. Press Enter, then select DONE and press enter. Your entire program should now read:

**:randInt(1,100)** ? N
**:Lbl A**
**:Disp "GUESS"**
**:Input G**
**:If G=N**
**:Goto B**
**:If G>N**
**:Disp "TOO BIG"**
**:If G<N**
**:Disp "TOO SMALL"**
**:Goto A**

The only part of the flow diagram that we have not yet incorporated into our program is the portion for when the user guesses correctly. If the guess G is equal to the secret number N, the program already has a command to Goto B, but we now need to add a Lbl B for it to go to, at the bottom of the program.

Use the arrow buttons to return to the end of the last line of the program. As before, press ENTER to get a new line, then press the PRGM button and use the arrow keys to select Lbl from the CTL menu, and press ENTER. Now press 2nd, TEXT, then use the arrow keys to select the letter A. Press Enter, then select DONE and press enter. This should add the line

**:Lbl B**

to the program. Finally, we must tell the user that they've won. Press PRGM, and then use the arrow buttons, first to select the I/O heading, then, under I/O, select Disp. Press ENTER. Then press 2nd, TEXT, and use the arrow buttons and the ENTER button to type "YOU WIN" (including the quotation marks). Select DONE, then press ENTER. The completed program should read:

**:randInt(1,100)** ? N
**:Lbl A**
**:Disp "GUESS"**
**:Input G**
**:If G=N**
**:Goto B**
**:If G>N**
**:Disp "TOO BIG"**
**:If G<N**
**:Disp "TOO SMALL"**
**:Goto A**
**:Lbl B**
**:Disp "YOU WIN"**

The program will simply end when it runs out of commands after the last line.

Now we should test our program. Press 2nd, QUIT to get back to the home screen, then press PRGM. Use the arrow buttons to highlight HIGHLOW under EXEC, then press ENTER. Press ENTER again, and then play the game. Type a number to guess, then press ENTER. Improve your guesses based on the program's responses until you guess correctly.

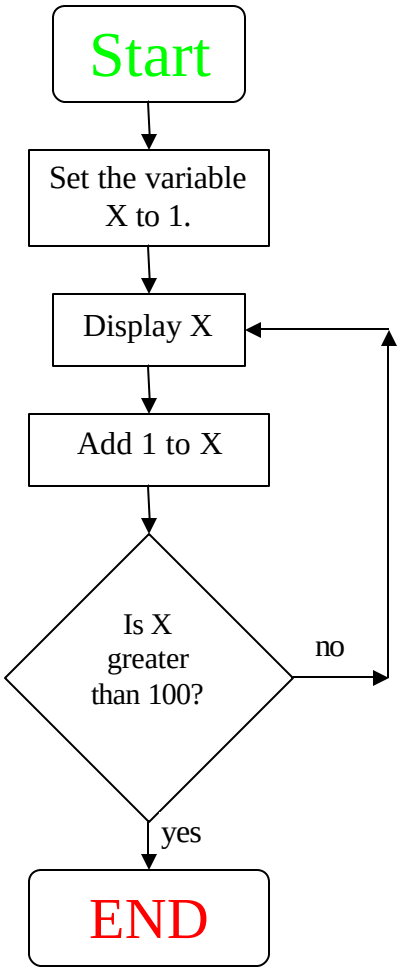
You will notice that the screen can get littered up with old information. One way to change this is to insert a new line after **:Input G** and put the command ClrHome on it. Now, every time the program loops, it clears away the clutter, but not until after the user has had a chance to see the results of the last guess and enter their new guess.

Now, one of the most powerful tools in programming is the for loop. It repeats the same set of commands, each time changing the value of a variable. To explain this tool, we'll make a simple program. Make a new program, and name it SANDBOX. Then enter the following code (the For( command is under CTL after you hit the PRGM button) :

**:For(X,1,100)**
**:Disp X**
**:End**

Now return to the home screen by pressing 2nd, QUIT, and run the program by pressing PRGM, then selecting SANDBOX under EXEC and pressing ENTER twice. The program should quickly count from 1 to 100, by ones, on the right side of the screen. A

flow diagram of this program might look like this:




While this simple example has little practical value, the For Loop will play a critical role in our next game.

You need to know one final fact before you start work on the handout. The calculator function abs() returns the absolute value of whatever expression you put in the parentheses.

*Pass out a handout sheet to every student. The last two pages of this document contain the student version of the handout. The preceding pages contain the answer key. Only a small percentage of the students should be able to finish the handout without help. Put them in groups of 4, and walk around answering questions as they work. Leave the flowchart above on the smart board as they work so they can use it as a reference. Encourage the students to compete at the game, particularly by racing to a perfect score. After all the groups finish, go over the worksheet in class.*

*TI-Math Game Handout: Teacher's Edition*

You have already participated in a group exercise translating a flowchart into code for a calculator game. Your class work starts with code, and works from there. The following code generates a math game.

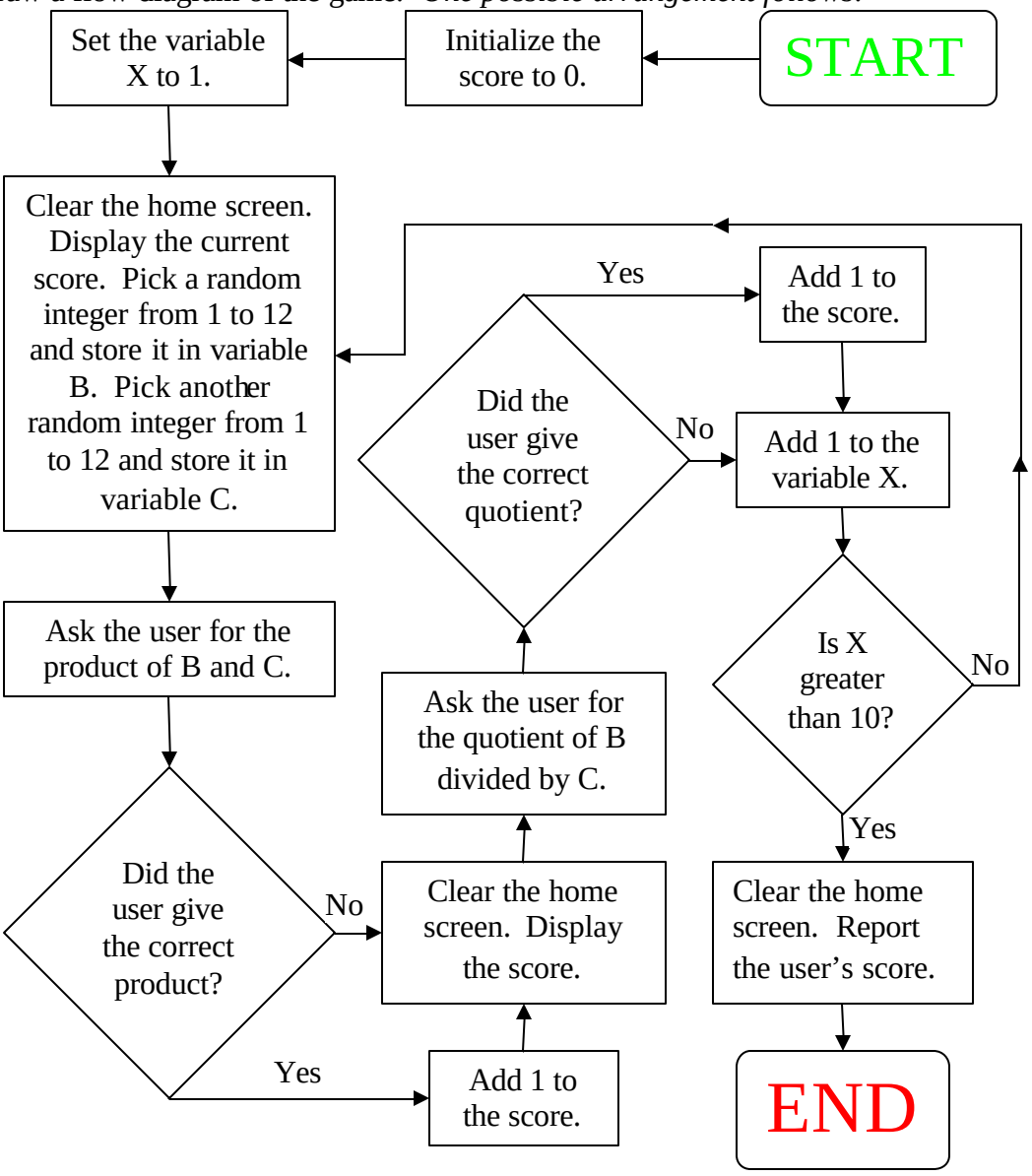
```
:0? S
:For(X,1,10)
:ClrHome
:Disp S
:randInt(1,12) ? B

:randInt(1,12) ? C
:Disp B,"TIMES",C
:Input D
:If D=(B*C)
:S+1? S

:ClrHome
:Disp S
:Disp B,"OVER",C
:Input E
:If abs(E-(B/C))<.01

:S+1? S
:End
:ClrHome
:Disp "YOU GOT A", (S/20)*100,"PERCENT"
```

1. What sort of game does the code make?
*It makes a math quiz, that generates two random integers from 1 to 12, then asks the user to input their product and then their quotient. After 10 pairs of numbers it tells the user what percent of the questions they answered correctly.*
2. Line 10 reads **:S+1? S** which adds one to the score variable, S. For the program to execute this line, the condition in the If statement in line 9 must hold true. What must the user do to make that happen (and earn the point)?
*The user must enter the correct product of the two given numbers.*

3. What purpose does line 1 serve?
*It sets the players initial score to zero. Without this line, the player would begin the game with the cumulative score of previous games.*

4. Line 19 shows the user their score after they finish the game. Why does (S/20)*100 give their score in percent form?
*The quiz consists of 20 questions. The number of questions that the user answered correctly, S, divided by the number of total questions offers the decimal proportion of questions answered correctly. To convert that to a percent, we multiply by 100%.*

5. Draw a flow diagram of the game. *One possible arrangement follows:*




Extra Credit: What does line 15 do? Why? How?

*Line 15 determines whether or not the user entered the correct quotient of the given numbers, to within two decimal places. While any two integers always have an integer product, decimal quotients often occur. Without a line that allows some small difference between the entered answer and the actual quotient, the user would have to perform extensive long division to earn credit for answers for many problems. The line works by subtracting the actual answer from the user's answer. If the distance between them is smaller than .01, they get the point. The abs() function ensures that this works regardless of whether the user offered an answer greater or less than the precise answer.*

What changes could we make to the code to make the game easier? Harder?

*There are many answers to this question. One way would be to make the highest random number generated either smaller or bigger. 1 to 5 should be easy; 1 to 20 will be harder.*

*TI-Math Game Handout: Teacher's Edition*

You have already participated in a group exercise translating a flowchart into code for a calculator game. Your class work starts with code, and works from there. The following code generates a math game.

```
:0? S
:For(X,1,10)
:ClrHome
:Disp S
:randInt(1,12) ? B

:randInt(1,12) ? C
:Disp B,"TIMES",C
:Input D
:If D=(B*C)
:S+1? S

:ClrHome
:Disp S
:Disp B,"OVER",C
:Input E
:If abs(E-(B/C))<.01

:S+1? S
:End
:ClrHome
:Disp "YOU GOT A", (S/20)*100,"PERCENT"
```

1. What sort of game does the code make?

2. Line 10 reads **:S+1? S** which adds one to the score variable, S. For the program to execute this line, the condition in the If statement in line 9 must hold true. What must the user do to make that happen (and earn the point)?

3. What purpose does line 1 serve?

4. Line 19 shows the user their score after they finish the game. Why does (S/20)*100 give their score in percent form?

5. Draw a flow diagram of the game.

Extra Credit: What does line 15 do? Why? How?

What changes could we make to the code to make the game easier? Harder?