Senior Coding for Good, Badge 1 requirements:

Self-portrait

1. Use functions to create a self-portrait.
2. Write code to create a portrait.

Quiz

1. Learn about computer logic
2. Explore **"If"** statements.
3. Use computer logic to create a quiz show.

Be sure to review the "Senior Coding for Good" overview/guide for the badge requirements and badge steps provided by your leader and the Girl Scouts. It also includes relevant vocabulary and interesting background information to spark your interest in coding. This lesson will allow you to earn **Badge 1: Coding Basics**, using your TI-84 Plus family graphing calculator.

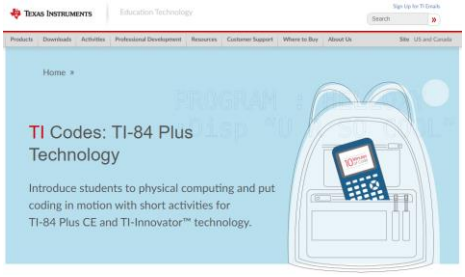
Introduction

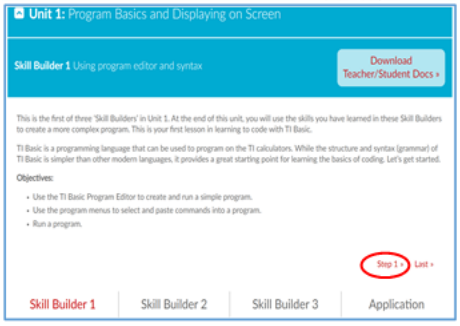
The programming language **BASIC** (stands for **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) was developed in the 1960s as an easy system for teaching computer programming. TI-Basic is similar to other flavors of BASIC, but you must select the programming words and commands from the onboard menus, as you will soon see.

The Senior Level, Badge 1: Coding Basics consists of two separate programming projects:

- A graphics program to produce a self-portrait, and
- A quiz program to help someone learn or review facts or a skill

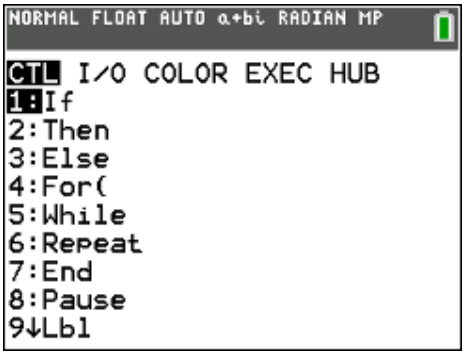
1. For an introduction to programming on the TI-84 see the TI Codes lessons at **education.ti.com > Activities > TI Codes > TI-Basic**. Units 1 through 5 should be enough to get started. Then return to this lesson for the **Senior Coding, Badge 1** project. Go to 10 Minutes of Code (for TI-84 Plus technology).





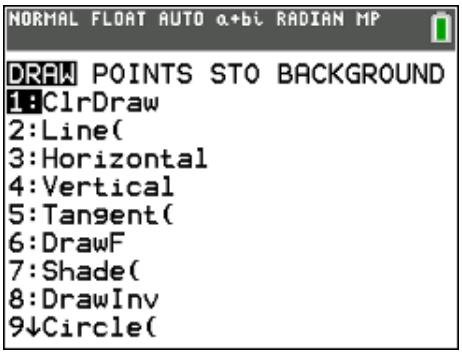
How to navigate TI Codes: Be sure to notice that each unit has three Skill Builders (SB) and one Application activity, as you can see here. You will navigate through each Skill Builder, by clicking on the "Step 1" in the right bottom corner, which will take you through the content. This screenshot shows you are currently in Skill Builder 1 > Step 1. After you complete all the steps in SB 1, you will then move to SB 2, and so on, until you complete all of Unit 1. You will then move on to Units 2, 3, 4 … and do the same thing.

2. After you have completed the TI Codes units, you know about:
   - The programming process (planning, coding, testing and debugging)
   - Using the TI-84 Basic Program Editor and its menus
   - Running a program
   - Some important pieces of TI-Basic code, including: Input, Disp, variables and assignment statements, **If…Then…Else** structures and loops, like **For** and **While** (some of these statements are shown to the right)
   - Drawing commands



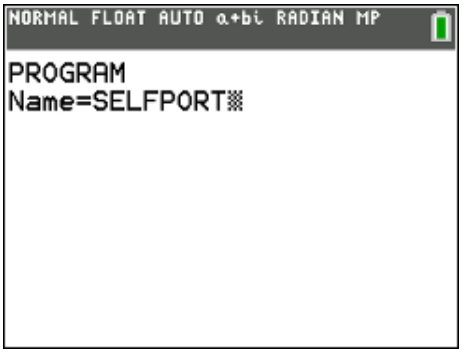
3. **Part 1: Self-portrait.** Plan your project on paper first!

   There are several **functions** on the [**draw**] menu shown to the right that let you design interesting graphics. These are covered in Unit 5: Graphics of the TI Codes lessons you completed. These functions produce points, lines, circles, text and more.



   Tip for the TI-84 Plus CE: The **BACKGROUND** *menu lets you add a background image to the graph screen. Background images (.jpegs) can be loaded onto your calculator using the free* TI Connect™ CE software application *for your computer.*

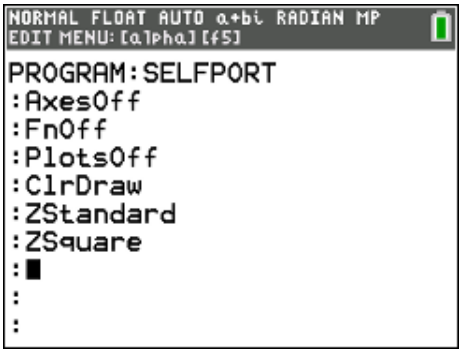
   a. Start a new program (press **[prgm] > NEW**), and name it **SELFPORT**.



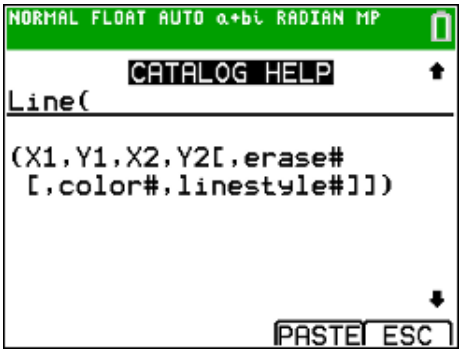
   b. When designing a screen for graphics display, we will use some setup commands that are found on several menus: **[draw]**, **[format]**, **[statplot]**, **[vars]** and **[zoom]**.

      The commands shown are to ensure that the program begins with a blank graph screen: no axes, no functions being graphed, no stat plots showing, and a *friendly* window where circles appear round.

      Remember that these, and **all**, calculator commands can be found on the **[catalog]** key (**[2nd] [0]**). Once in the catalog, press the first letter of the command you want, scroll down until you find the command you want, and press **[enter]** to add it to your program.



c. Also, while pointing to a command on a menu, pressing the **[+]** key will show the CATALOG HELP screen. Help for the **Line(** statement is shown here. When you select a function, you will get an informational screen. You can complete the command at the top of the screen and then press **[PASTE]** (use the **[zoom]** key) to paste the entire completed statement into your program.



d. To draw a line segment, use the **Line**( ) function from the **[draw]** menu which requires four (or more) arguments.

Line(0, 0, 4, 5)

draws a line *segment* between the points (0,0) and 4,5) using the current window setting.

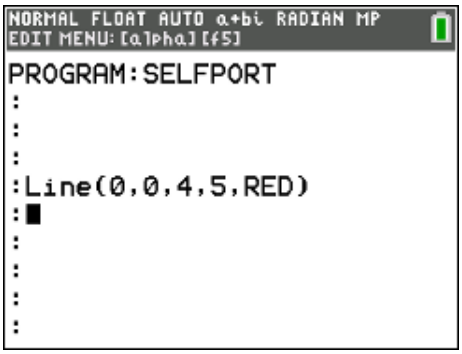
For the TI-84 Plus CE: Line*(0,0,4,5,* RED*) makes a red line.*

Get the variable RED *from* **[vars] COLOR**. You *cannot* simply type the letters R-E-D.

Reminder: ***[2nd] [PRGM]*** *accesses the DRAW menu.*

Tip 1: Remember to always close the parenthesis! It's a good habit.

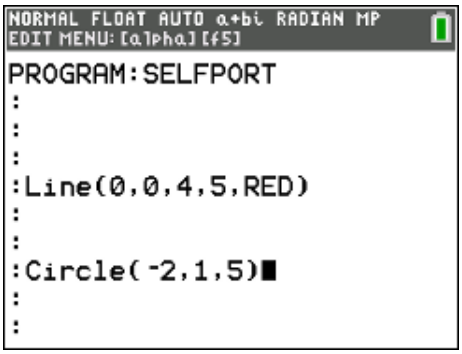
Tip 2: See **[alpha] [graph]** *(the* **F5** *key) for special editing commands, especially "Undo clear" which restores an accidentally cleared line of code.*



e. To draw a circle:

Circle(-2, 1, 5)

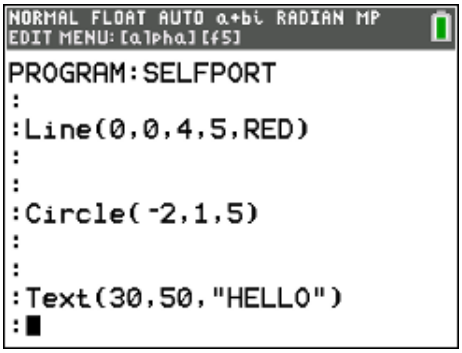
draws a circle with center at (-2,1) and radius 5 using the current window setting.



f. To draw text on the screen:

Text(30,50,"HELLO")

Draws the word HELLO at the pixel on *row* 30, *column* 50 of the screen *regardless of the window setting!*



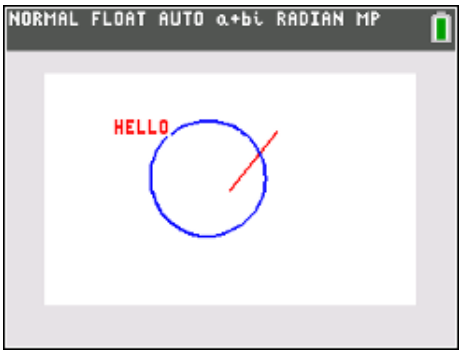
Reminder: " is on the [alpha] [+].

Note: The TI-84 Plus graph screen is 64 rows by 96 columns. The TI-84 Plus CE graph screen is 164 rows by 264 columns.

Run your program. Press **([2nd] [MODE])** to Quit and return to the HOME screen. Then **[prgm]** and under the **EXEC** menu, select your program.

g. The **Line**, **Circl**, and **Text** commands should produce the screen shown to the right.

Circle(-2, 1, 5, BLUE)
Line(0, 0, 4, 5, RED)
Text(30,50,"HELLO")



On a TI-84 Plus, the image is black and white.

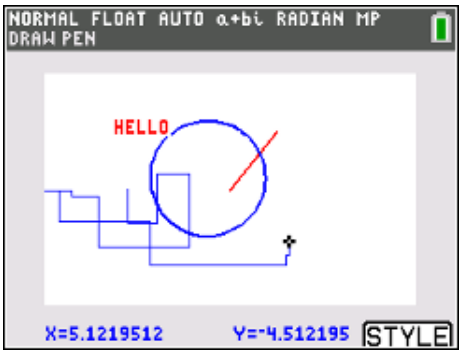
Tip for the TI-84 Plus CE: Use the separate **TextColor( )** *function on the* **[draw]** *menu to set the text color* ***before*** *drawing text. If you don't see the RED, you probably typed in the word, instead of using the menu choice. You* ***cannot*** *just type the color in, you must use the menu.*

As a reminder, to get back to your program code to edit further:

- Press [PRGM]
- Choose the **EDIT** menu using the arrow keys
- Select your program and press [ENTER]

h. **Freehand drawing and coding.**

From the Home screen or Graph screen, select **[draw] Pen**, a freehand drawing tool. Move the cursor around on the graph screen, press **[enter]** to put the pen down (or up) and make a sketch. The screen to the right shows the pen drawing in progress.

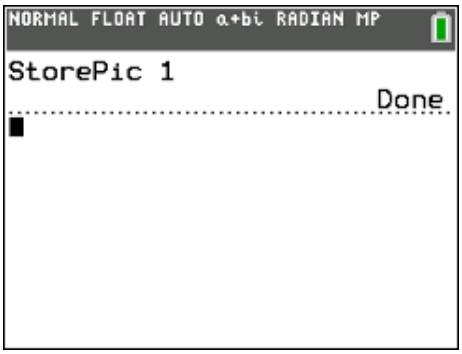


*Tip for the TI-84 Plus CE: The STYLE menu (***[graph]***) lets you change the pen color.*

Hold on … since pen drawing is not "programming," you must **now** *save* your drawing in a **Pic** file that can be incorporated into your code. If you modify the graph screen before saving, your drawing will be lost. See the next step on how to save the drawing.

i. When done drawing, select **[draw] STO StorePic** (the command is pasted onto the home screen) and store the picture into one of the 10 pic variables in the calculator *by typing a number from 0 to* 9. You can also press **[vars] Picture**) for the list of the 10 Pic variables: Pic0, Pic1, Pic2, …, Pic9). Select any one of those variables:

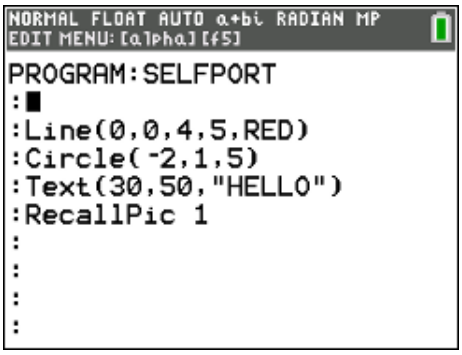
StorePic 1 or **StorePic Pic1**



It will have the same result.

After saving you will return to the drawing screen. You can press **[quit]** now to go to the home screen.

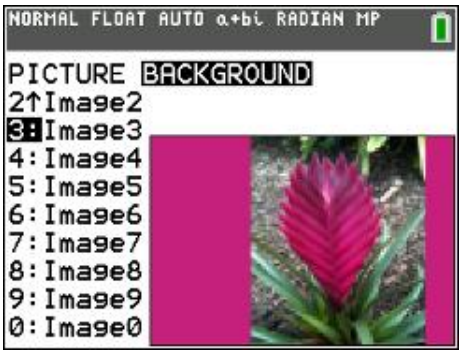
j. Now we need to add code to your program to include that drawing (Pic file). Use **[draw] STO RecallPic** # to add the stored pixels to your project. Any previous drawing effects remain on the screen.



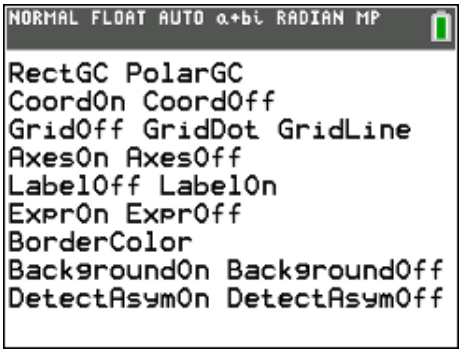
To insert blank lines in your program:

- Place the cursor at the insertion point (the beginning of a line)
- Press **[2nd] [del]** for **[insert]**
- Press **[enter]**
- Press **[uparrow]** to place the cursor on the new blank line

k. **Background images** (TI-84 Plus CE only): In addition to **Pic** files, the TI-84 Plus CE can store Image files for use as a background image on the Graph screen. It comes with some preloaded image frills from the factory. Press **[vars]** > Pictures and Background > Background and look through the images.

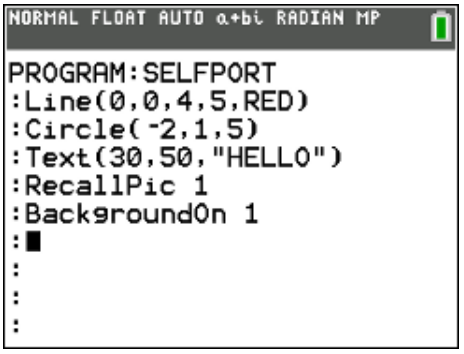


l. To use a background image in your program, use the **BackgroundOn** command followed by a number from 0 to 9. Within the Program Editor, find the **BackgroundOn/Off** commands on the **[format] ([2nd][Zoom)** screen shown.

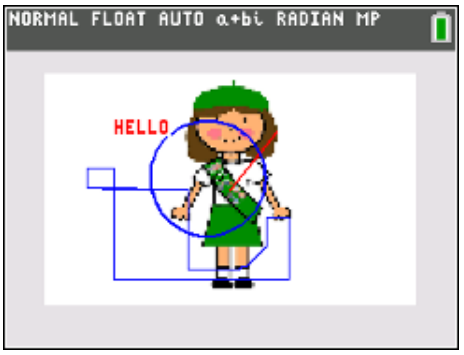


m. The statement **BackgroundOn 1** places image #1 in the background of your graph screen without disturbing any of your current drawing.

n. To use your own image file, use the free TI Connect™ CE software to transfer a picture from your computer to your calculator using a USB cable.

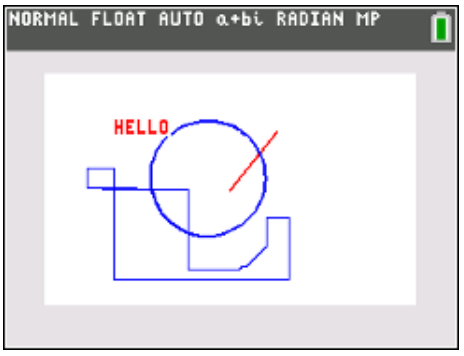


o. This is an example of a background image. This scout image is located in the download folder if you want to try and put it on your calculator.



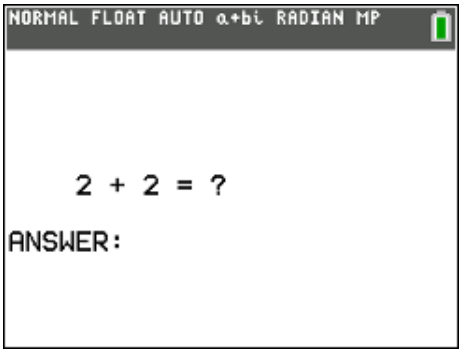
p. Now it's your turn! Try and use a combination of these graphics tools to make your own creative self-portrait program.

To share your project, you can take a picture of the screen for sharing with friends and family, or you can use TI-Connect™ CE software to capture a screenshot for sharing. If you share on social media, don't forget to tag it **@TICalculators**!



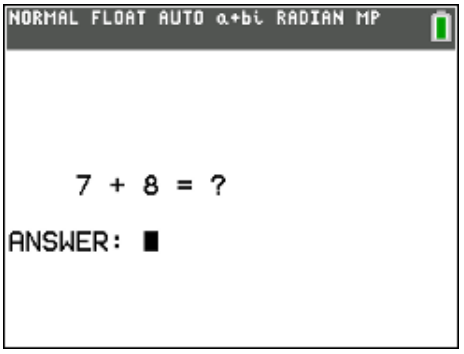
4. **Part 2: Quiz**

Calculator programming is a good way to develop a program that can allow a child to practice their arithmetic skills! You can easily expand on this demonstration program in several ways.

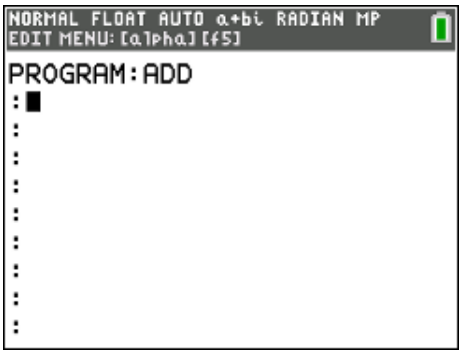


a. **The plan:**

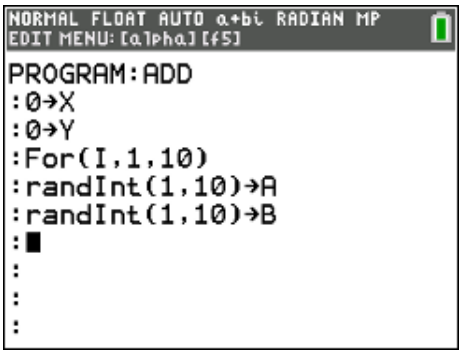
The program will ask 10 *addition* questions. The two numbers, called addends, are generated using **randInt( )**. The program keeps track of the number right and the number wrong and displays those results at the end. The screen is organized using **Output()** statements.



b. Begin a new program (press **[prgm] NEW [enter]**) and name it **ADD** and press **[enter]** again**.**



c. Begin your program with two variables that keep track of the number right and wrong (we use **X** and **Y**). Remember that the → is the **[sto]** key.

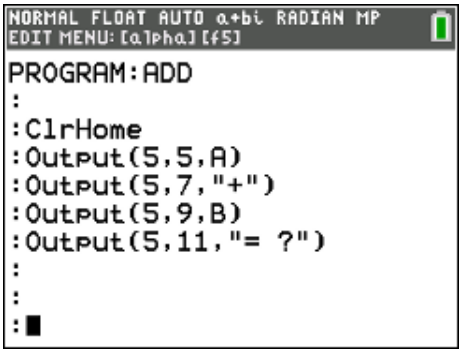


i. Add a **For** loop to control the number of problems (we are going to display 10 problems). Press **[prgm]** and locate the **For(** statement on the **CTL** menu, then type the rest of the statement.

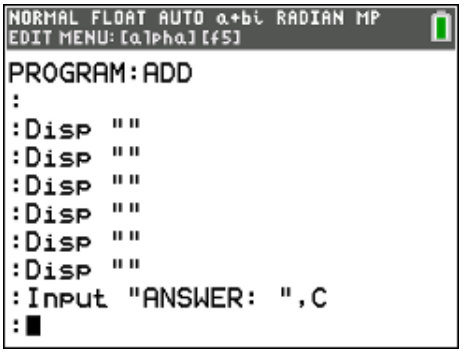
ii. Use two **randInt()** functions to pick two random number to add together (we're using random numbers from 1 to 10). **randInt(** is on the **[math] PROB** menu. Store (→) the **randInt**s in two variables, **A** and **B**.

d. Display the problem using **Output**( ) statements to position each part of the expression on the screen in the proper location. Use **ClrHome** (on **[prgm] I/O**) to clear the screen and find **Output(** on **[prgm] I/O** as well.

Output(uses the structure **Output(***line#*, *column#*, *item***)**. Item can be a variable or "literal string" (something in quotes). In the example screen, all output appears on line five of the display.

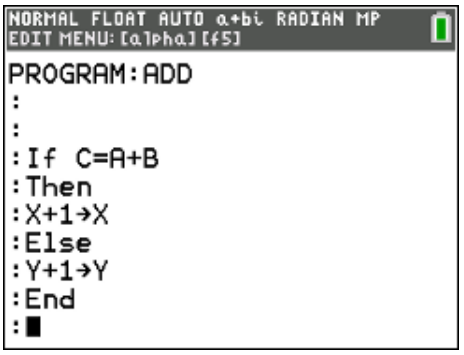


e. Use an **Input** statement to get the answer from the user. Since the Output statement does not affect the cursor position, use some **Disp** statements to move the cursor lower on the screen.



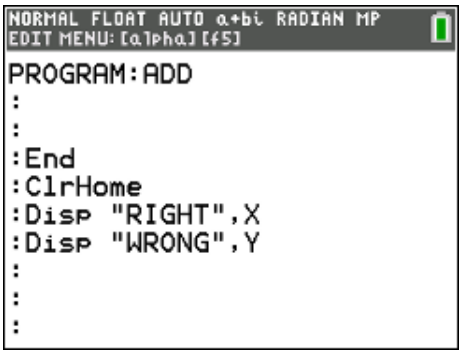
f. Use an **If** and an **Else** statement to determine if the users answer is right and update one of the two counters accordingly.

Remember that **If**, **Then**, **Else** and **End** are four separate commands, all on the **[prgm] CTL** menu, and that each belongs on a line of its own (with a condition after **If**).

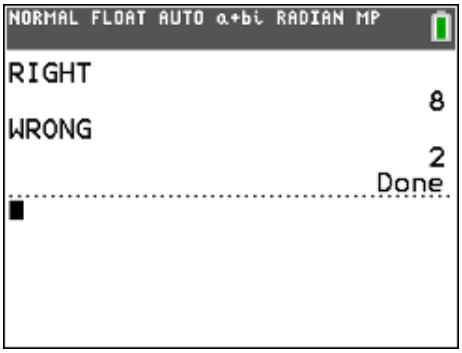


g. After the **End** of the **For** loop, report the results.

Remember that there will be two **End** statements in your program, one for the **If** structure and one for the **For** structure.



h. Using this program as a start, you can improve the program in many ways. Get creative! Some suggestions are listed below in the Extensions (optional).



Be sure to share your work on social media, and don't forget to tag it ***@TICalculators****!*

Congratulations!

You have completed the requirements for earning your **Senior Coding for Good, Badge 1: Coding Basics**. Now that you have completed this requirement, you are challenged to *give service* by *sharing* what you have learned about coding with others. Refer to the "Coding for Good" Girl Scout guide for suggestions on how to do so.

What's next? (Optional extensions)

Ready to try some additional practices with your new skills?

1. In the self-portrait graphics project, once your final image is complete, you can store the image in any one of the PICTURE variables (see **[vars] PICTURE** and **[draw] STO** menus). You can then have several picture variables stored with slightly different images, and then write a program that displays the images one at a time to create an animation:

```
:For K,1,5
:ClrDraw
:RecallPic K
:Wait .1
:End
```

2. In the **ADD** program:
   - Let the user enter the number of problems
   - Display (**Output**) the problem number at the top of the screen
   - Choose the operation (addition, subtraction, multiplication, division)*
   - Offer multiple tries ("Wrong, try again."); add points for getting it right on 1st try, 2nd try, etc.
   - Improve the results screen shown using Output statements
   - Display a percent score at the end

**For division, to ensure integer answers, choose two random numbers, determining their product. Display the product and one of the divisors. The user then determines the other divisor.*

3. See the activity "**Back in Time?**" found at Back In Time? Algebra I. Use the program MACHINE provided to develop a function guessing game of your own. You may need to do some research on the use of the Menu( ) feature of TI-Basic. As an extension to MACHINE, offer another multiple-choice question at the end of each function that shows three or four different function expressions and the student has to pick the correct one.