Starting From Scratch

Part A: A Little Encouragement

You can't learn to drive by reading about it or even by watching other people do it — and the same holds true for learning to program a computer! To develop any real facility you must become familiar with an actual programming language and use it to write your own programs.

A programming environment, like a car or anything else one is trained to use, takes some getting used to. This assignment has been designed to "put you in the driver's seat," and to show you where the "controls" are. Specifically, the following pages will get you started with *Scratch*; we sincerely hope you will quickly discover that programming can be an enormously exciting (and fun) intellectual activity.

Of course, there *is* one important difference between programming a computer and driving an auto: Using a computer is NOT dangerous, and you can't do any damage to the computer, yourself, or anyone else — unless, of course, you fling your laptop out the window or you become an *uber geek* who neglects to eat and sleep!

Some advice: don't be afraid to try something, even if you don't know exactly what will happen. *And please don't be shy about asking for help* — especially during the next couple of weeks. The teaching fellows will hold regular office hours on Zoom, and will also be available by appointment and through email.

So don't delay even one more hour! Go to it … **note the due date for the various parts of this assignment** … and … *Good Luck!*

Part B: A Gentle Introduction to Scratch (3 points)

This part should be completed prior to 5 pm on Sunday, January 31.

1. (2 points)

   Before you do anything else, please fill out the online section form at: https://tinyurl.com/e7-2021

   The survey is also available from the main page of the CS1 course website: https://canvas.harvard.edu/courses/81756

2. (0 points)

   Follow the instructions on the *Scratch* page of our course website: https://canvas.harvard.edu/courses/81756/pages/programming-in-scratch in order to download and test out *Scratch* on your computer. It doesn't matter if you use version 2 of Scratch or the newer version 3.

3. (0 points)

   Go ahead and open a few more *Scratch* projects, even some of those that you already saw in lecture. For each project of interest to you, run it to *see* how it works; then, look over its scripts to *understand* how it works. Feel free to make changes to scripts and observe the effects. Once you can say to yourself, "Okay, I think I get this," you're ready to proceed to the next problem.

   Though you are probably "itching to program," it's probably a good idea to first familiarize yourself with various aspects of Scratch in a systematic way. From a web browser, go to https://resources.scratch.mit.edu/www/guides/en/Getting-Started-Guide-Scratch2.pdf

   The first page will look like this:

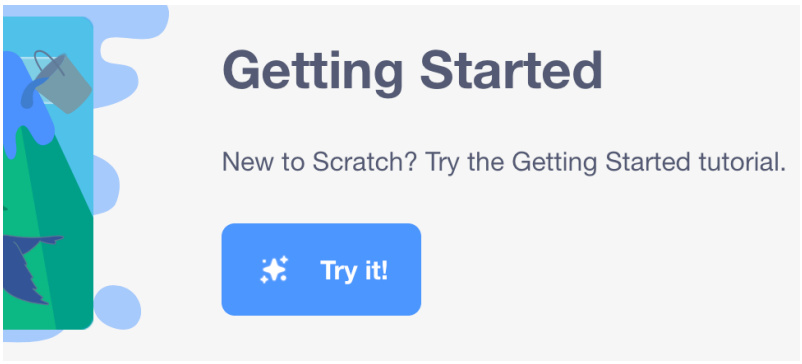
Solve at least the first 5 exercises that are described in this PDF.

4. (1 point)

Here's a point just for making it through so many 0-point questions!

5. (0 points)

You might also try out the step-by-step interactive "Getting Started" tutorial that appears near the top of https://scratch.mit.edu/ideas



6. (0 points)

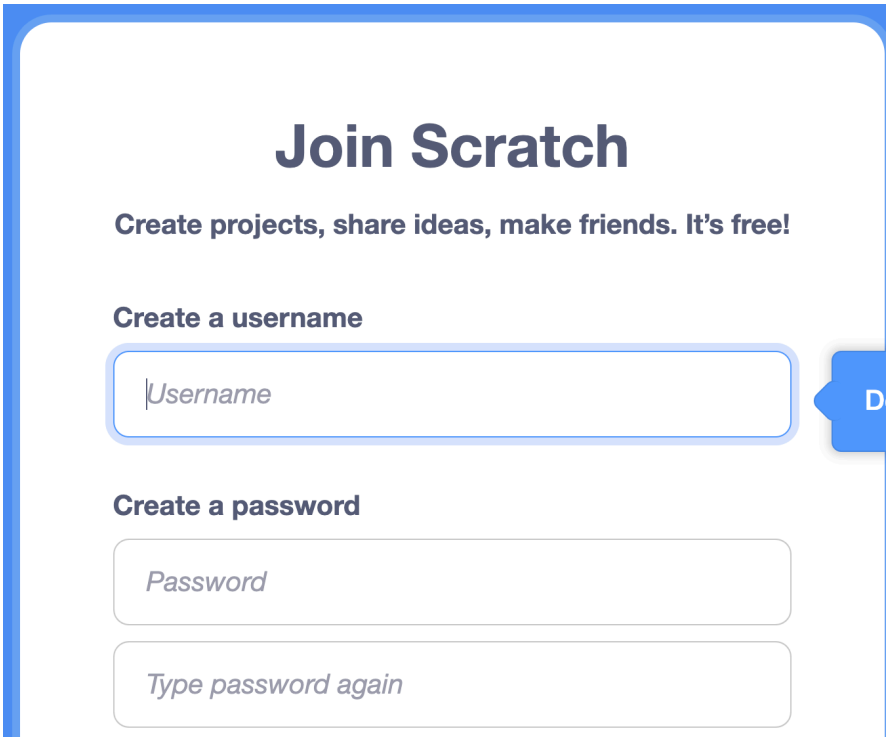
Use the **say** command and the appropriate blocks from the *Operators* palette to calculate the following:

- the square root of 37
- the cosine of 60 degrees
- the result of rounding 99.459

As an alternative to using the **Scratch 3 (or Scratch 2) Offline Editor**, you can instead use *Scratch* via the web, as described below.

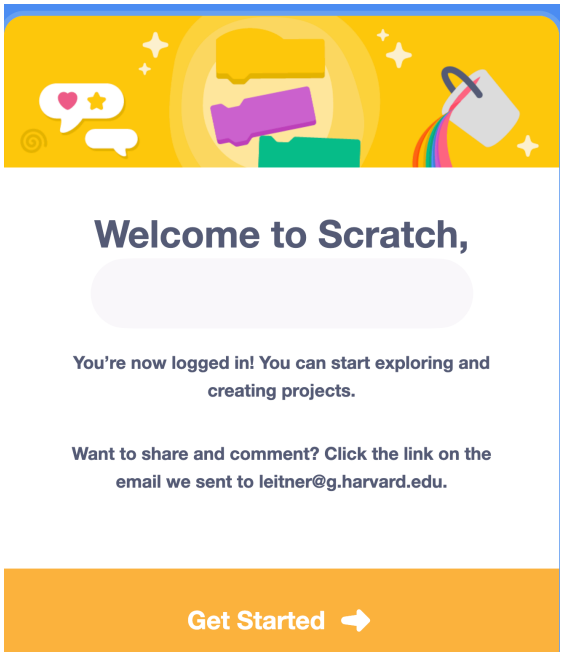
Using a web browser, go to the website: **http://scratch.mit.edu**
Near the top of the page, you will see a menu bar that looks like this:

Click the link that says "Join Scratch" on the righthand side, and fill out the simple form that appears on your screen:



Ideally, pick a *Scratch Username* that's identical to your Harvard email username.

Click the **Next** button, and keep clicking this button on all subsequent screens that appear. Eventually, you will get a window appearing that looks like this:



You will receive an email to confirm this account.

You must submit your solution to all questions in the rest of this problem set via "upload" to the Computer Science 1 course website, https://canvas.harvard.edu/courses/81756

To do this, first click the Gradescope button, available on the lefthand side of the website's main page.

You will need to upload a ".zip" file. Instructions for creating this file appear at the end of part D of this problem set.

Parts C and D are both due prior to 5 PM on Friday, February 5. Part E (which is required for graduate-credit students and optional for everyone else) may be completed prior to 10 PM on Sunday, February 7.

Part C: Simple Exercises (25 points total)

7. (4 points)

Suppose variable **x** has the value **2** and variable **y** has the value **4**.

What is the value of each of the following Scratch expressions?

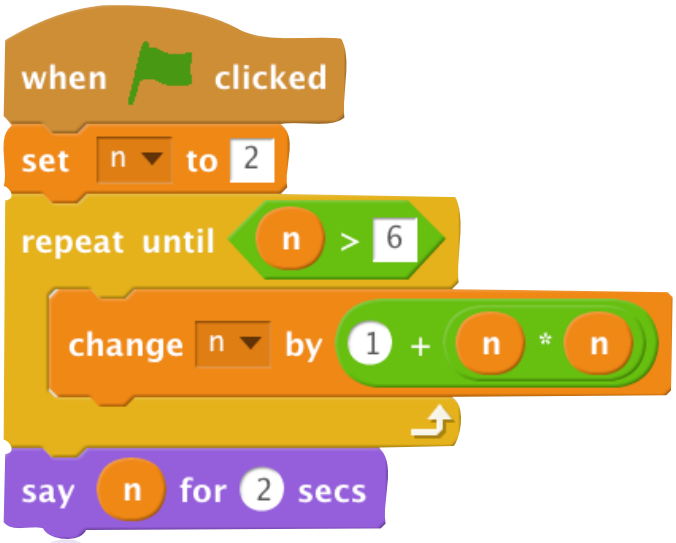
Submit your responses to these questions in the file
problem7.yourLASTname.txt

Part (a) 2 points

(b) 2 points

8. (7 points)

What will the following Scratch program cause the sprite to "say" at the end?



Suppose we change the number **6** to **10** and run the program again? What will the sprite "say" at the end? What if we change the **6** to **20** and run the program? Submit your responses to these questions in the file **problem8.yourLASTname.txt**

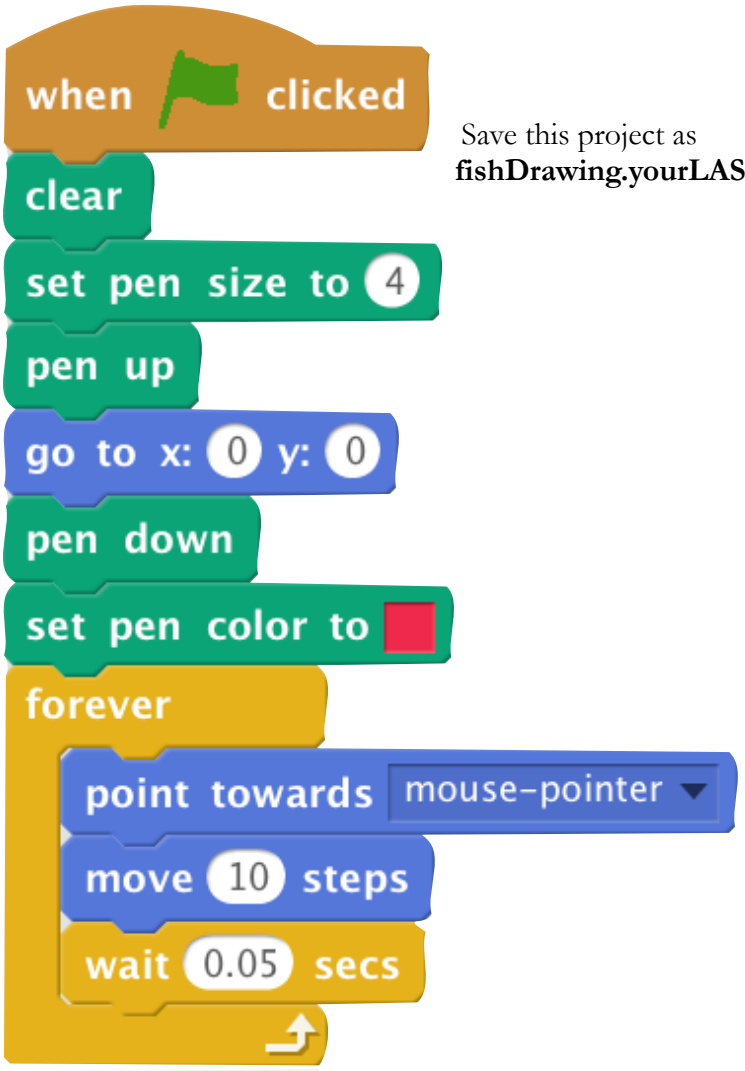
9. (5 points)

Create a new *Scratch* project. Change the standard cat sprite into a "fish" (one of the standard sprites that comes with the *Scratch* software) and then construct the following two *Scratch* scripts (see the next page) for that sprite.

Run the program by clicking the green flag. What happens when you move your mouse around on the stage while the program is running? While the program is executing, you should occasionally press the *space* key and watch what happens.[1]

Now add some code to this project so that every time you press the letter "C" on the keyboard the stage will entirely get erased before the sprite starts drawing again.

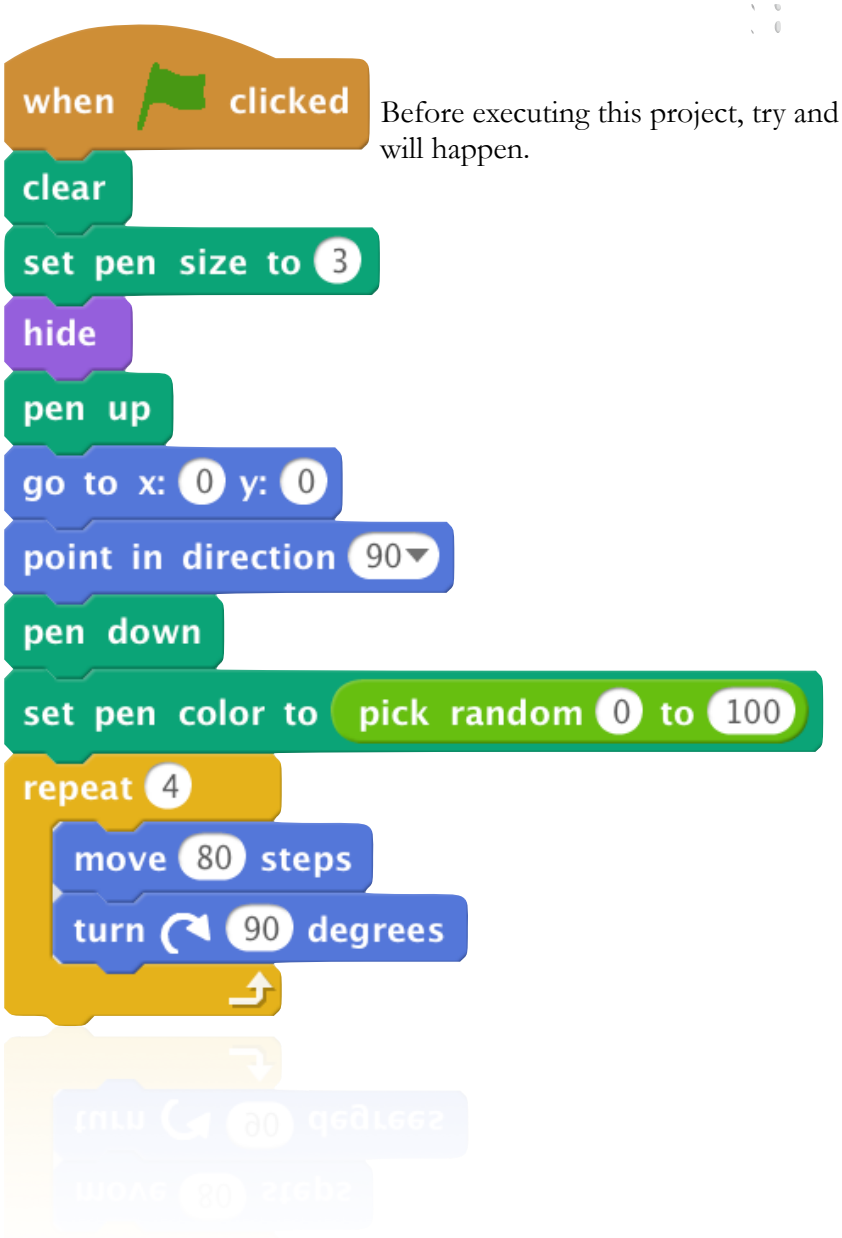
[1] Note the color may not change that perceptibly unless you hold down the spacebar for a while. The color range is 0-200.



Save this project as
fishDrawing.yourLASTname.sb3

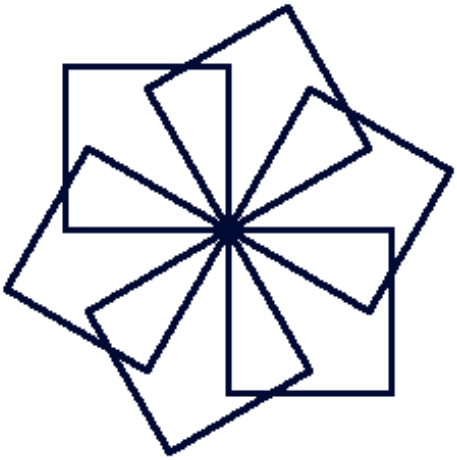
10. (9 points)

Create a new *Scratch* project, and construct the following program.



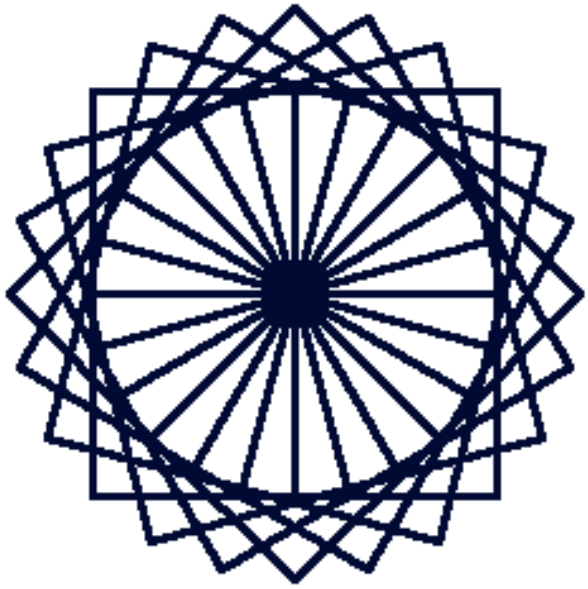
Before executing this project, try and predict what will happen.

Now modify this program so it can produce output that looks like this:



If you look carefully at this figure, the simple square shape that the original program would draw just once is now being drawn *six* times (each one rotated a bit more than the previous one), so we get a circular overall shape. The modification you need to make should be really simple: it will consist of adding a **repeat** loop which will enclose the existing **repeat** loop (in other words, you'll end up with a repeat loop inside of another one). In addition, you will need to add a "turn" command.

Here is what the modified program would produce if we simply increase the number of repetitions in the new **repeat** loop and correspondingly decrease the number of degrees that the sprite rotates before drawing another square:



(Hint: the number of repetitions multiplied by the number of degrees the sprite turns before the next repetition should equal 360 in order to accomplish the overall circular shape.) Save this project as **rotatingSquares.yourLASTname.sb3**

Part D: More Complicated Programming Problems (75-80 points)

Before you continue with this part of the problem set, it is strongly suggested that you attend one of the section meetings. The schedule of sections is posted on our course website.

11. (15 points total)

Go back to the original Scratch project featured in problem #10. Modify this project so that instead of drawing a single square, it will instead draw a row of 6 squares, spaced apart as shown below. You'll need to adjust the starting (x,y) coordinate so that the pen begins drawing toward the top left of the stage.

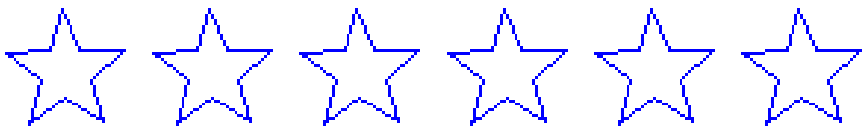
Save this project under the file named **sixSquares.yourLASTname.sb3**

Now modify your program so that instead of 6 squares, you get 6 equilateral *triangles* (i.e., each triangle consists of sides that are equal in length). Each triangle will contain three 60-degree angles. Save this project under the file named **sixTriangles.yourLASTname.sb3**

Now modify your program again so your program draws a row of objects containing some other number of sides (e.g., hexagons or pentagons). Save this project under the file named **sixOtherShapes.yourLASTname.sb3**

12. (20 points total) + 5 points of "extra credit"

You are to modify your program yet again, so that it produces a row of six 5-point "stars" like this:

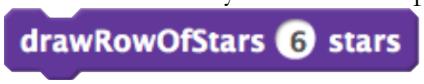


Remember that the Scratch stage is 480 units wide and 360 units tall. So you may need to do some experimentation (and some basic math) to figure out the size of the stars and the spacing.[2] Do not worry about the background color or filling in the stars. Save this project under the file named **sixStars.***yourLASTname***.sb3**

[2] You may Google the angles for a pentagram.

For extra credit: Now that you can draw a row of stars, you need to draw a field of 9 rows in order to produce an image that resembles the upper-left of an American flag, as shown below.

Don't worry if you can't figure out how to produce white stars on a blue background; blue stars on a white stage is good enough! Consider defining a two "new blocks": one named **draw1Star** (no parameters), and another one named **drawRowOfStars** (this one accepting a number parameter, **n**, to indicate how many stars should appear in a single row). For example,



You might consider having this block accept another parameter, that supplies the amount of blank space that needs to appear between each star.

Note: 5 of the rows have 6 stars, and 4 of the rows have 5 stars. You could write nine complete chunks of code, one chunk for each row. If you do that, you get 1 of the five points for this section. A more concise, and more flexible solution, is to look for repetition in the pattern. Then, use loops to produce that repetition. You might use a **repeat-until** or just a regular fixed **repeat** puzzle-piece. There are lots of ways of doing this. You get full credit if the stars line up in the nice diagonals you see in the picture above, but you only lose one point if the right number of stars appear with roughly the correct spacing. Save this project as **flagStars.yourLASTname.sb3**

Note: There is a setting called "turbo mode" under settings that will make the drawing go much faster.

13. (7 points)

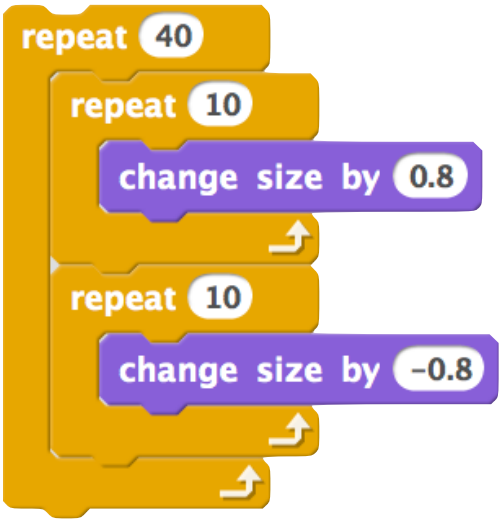
Write a program that graphically displays a die (one of

a pair of dice) every time you click the green flag. The die should randomly show between 1 and 6 white dots, which roughly corresponds to tossing the die. You will need to create 6 different "costumes," one for each of the 6 possible values. For example:

(You may use images found online for the die faces.) Save your solution in a file named **randomDie.yourLASTname.sb3**

14. (8 points)

Consider the following sequence of blocks. When you double-click the outer block, the current sprite (e.g., the cat) will appear to "pulsate." Try it out.



Now "Make a Block" named **pulsate** that contains the above sequence. Edit the definition of **pulsate** so that it accepts 3 numeric parameters:

- The first parameter should take the value of the number of iterations in the outer repeat block (40 in the above example)
- The second parameter should take the value of the number of iterations used by the two inner repeat blocks (10 in the above example)
- The third parameter should take the value of the amount by which the size is changed (0.8) in the above example.

Thus when you use the new **pulsate** block, you will be required to supply 3 actual numeric arguments. Test **pulsate** out on a variety of input values, and note the effect of making the 3 arguments larger or smaller than what is used in the above example. The 3 actual arguments can be "hard coded," as opposed to being input from the keyboard. Save the project as a file named

pulsate.yourLASTname.sb3

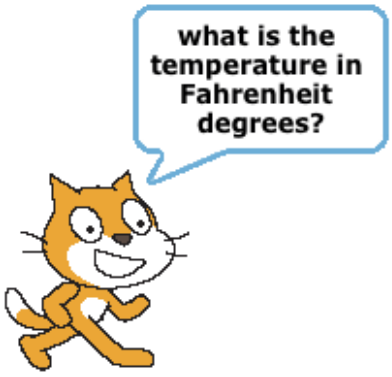
15. (10 points)

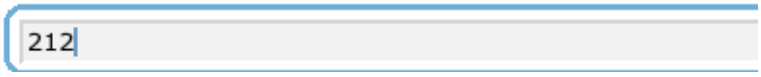
Save the following project as a file named **temperature.yourLASTname.sb3**

To convert a temperature from Fahrenheit degrees to Kelvin (absolute), the following relationship holds:

$$°K = \frac{5}{9}(°F - 32) + 273.16$$

Write a Scratch program that will print the Kelvin equivalent of any Fahrenheit temperature, based upon the current value that gets input by the user at the keyboard. In other words, when you start your program, the sprite will "ask" the following:





If the user responded by typing **212** (as in the above example), the response that gets output would be this:

16. (15 points)

According to the U.S. Center for Disease Control (CDC), a person's "body mass index" (BMI) is a number that is easily calculated from a person's weight and height. BMI provides a reliable indicator of body fatness for most people, and is used to screen for weight categories that may lead to health problems. The formula used for BMI is simply

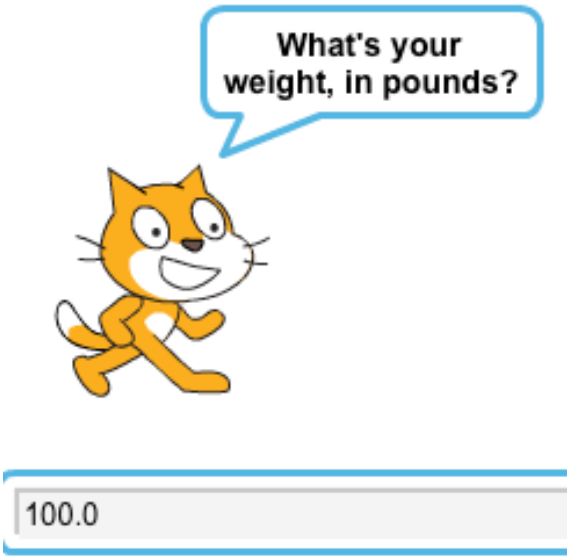
$$\frac{weight\ (in\ pounds)}{height\ (in\ inches)^2} \times 703$$

Knowing a person's BMI allows one to express a person's "weight status," according to the following table:

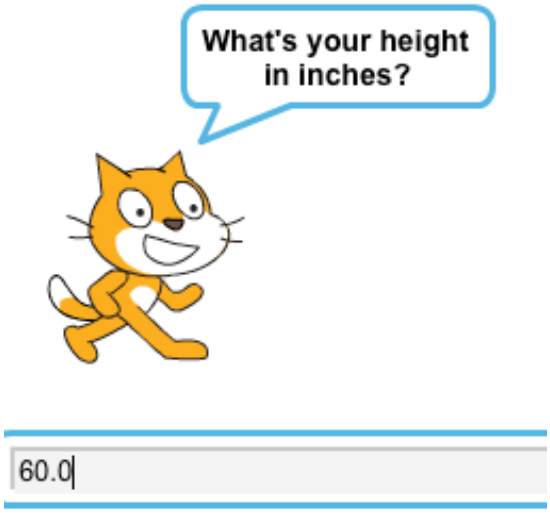
BMI	Weight Status
Below 18.5	**underweight**
18.5 to 24.9	**normal**
25.0 to 29.9	**overweight**
30.0 and above	**obese**

Write a *Scratch* program that accepts two values (a real number that expresses a person's weight in pounds, and another real number that expresses that person's height in inches), typed in by the user. Your program should calculate the person's BMI, and then output the appropriate "Weight Status."

Here is an illustration of what your program might look like in action:



The sprite then says you need to input a *height*:



And then the output appears:

Save the project as a file named

weight.yourLASTname.sb3

Note that the CDC's weight status table has some gaps. For example, what if a person has a height of 73.5 inches and a weight of 230 pounds? Their BMI would be computed as 29.9301217, which isn't between 25.0 and 29.9. It seems clear the CDC probably means that for a BMI of 18.5 up through 24.999999 (repeating), the status is "Normal", but having a repeating decimal would look a bit odd in a table. Your program should correctly perform such computations, without any "gaps."

You will now submit the projects you created in parts C and D of this assignment in a so-called *.zip* file named *yourLASTname.zip* that you will upload using Gradescope in the CS-1 course website.

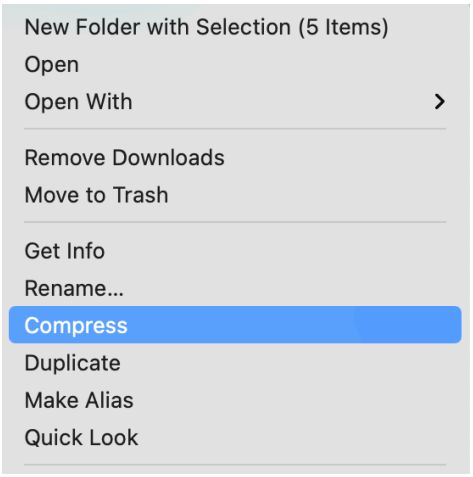
Here are some details about how to create the .zip file:

You should create a compressed, ".zip" archive of the files you created in solving problems 8 through 16. Name this compressed file ***yourLASTname.zip***

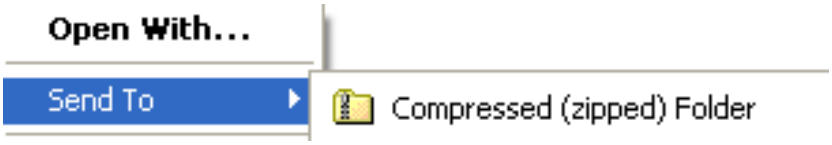
To create a .zip file is easy. On a Mac, select the 3 files with your mouse so they are all highlighted at the same time. Then hold down the "Control" key while clicking these files, and the following contextual menu will appear. Select the item that says "Compress 3 Items", and a file named Archive.zip will be created. Rename it to

yourLASTname.zip

See the following screen shot (from the latest version of Mac OS X, Big Sur):



It's just as easy to create the compressed file on Windows. Right-click the files after selecting them all, and the following contextual menu will appear:



Choose the "Send To" option, and from that select "Compressed (zipped) Folder". You will have to rename the created folder to
yourLASTname.zip

NOW you can submit this file to the electronic dropbox on the CS1 course website via Gradescope.

Part E: An Original Project (up to 20 points)

This part of the assignment is required only for graduate-credit students, and is optional (for extra credit) for undergraduate-credit students.

17. (up to 14 points)

And now for a real adventure! Your task for this problem is, quite simply, to have fun with *Scratch* and implement a project of your choice (be it a game, an animation, or something else), subject only to the following requirements.

i. Your project's filename must be **`student.sb3`**, where **`student`** is your last name and first name (joined together).

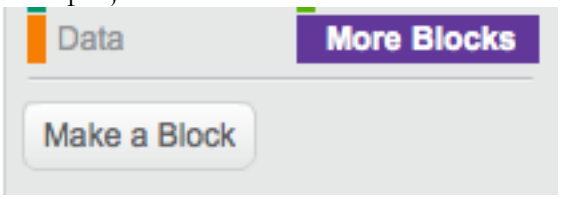
ii. Your project must have at least two sprites, neither of which may be a cat.

iii. Your project must contain at least 4 *scripts* in total (*i.e.*, not necessarily per sprite).

iv. Your project must use at least one condition, one loop, and one variable.

v. Your project must use at least one sound.

vi. Your project must define at least one new block, using



You can decide whether the new block requires parameters.

vii. Your project should be more complex than the simple, short examples that are described in the lecture notes (and appear also in your **`ScratchExamples`** folder) but ought to be less complex than *Oscartime*. Its complexity should be more on par with the other projects that come with Scratch. Thus your project should probably use a few dozen puzzle pieces overall.

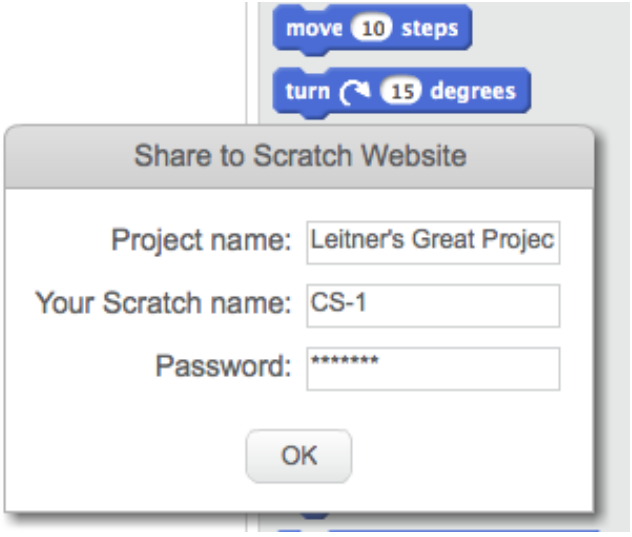
Feel free to look through the projects that come with Scratch for inspiration, but your own project should not be terribly similar to any of them. Try to think of an idea on your own, and then set out to implement it. If, along the way, you find it too difficult to implement some feature, try not to fret: alter your design or work around the problem. If you set out to implement an idea you find fun, you should not find it hard to satisfy this problem's requirements.

If you suspect your program might fall short of our expectations, feel free to ask for our opinion prior to submitting. *All right, off you go.* Impress us! A non-trivial prize shall be awarded for the best two or three programs.

18. (0 points) *This question is optional, but highly recommended!*

Consider sharing your project!

Within Scratch itself, click the **File** menu and select "Share to website." A dialog box will appear, in which you can type the name of your project. We have set up a



special *Scratch* account for this purpose, so please type **CS-1** as "Your Scratch name," and type **scratch** in the box where it asks for your Password:

Your project will shortly be uploaded to the **CS-1** account. (Note that the password, when you type it, will show up as asterisks on screen.)

Your responses to problems 19— 24 must be placed in a plain text file named
`questions19to24.yourLASTname.txt`
and uploaded to our course website.

19. (1 point)

In a short paragraph, tell us what your project does (or how to use it). In one or more longer paragraphs, explain how your project works, noting the purpose of each sprite and script. Alternatively, use the "commenting" feature of Scratch and

incorporate typed comments into your actual scripts.

20. (1 point)
Roughly how much time did you spend implementing **student.sb3** for problem 18 above?

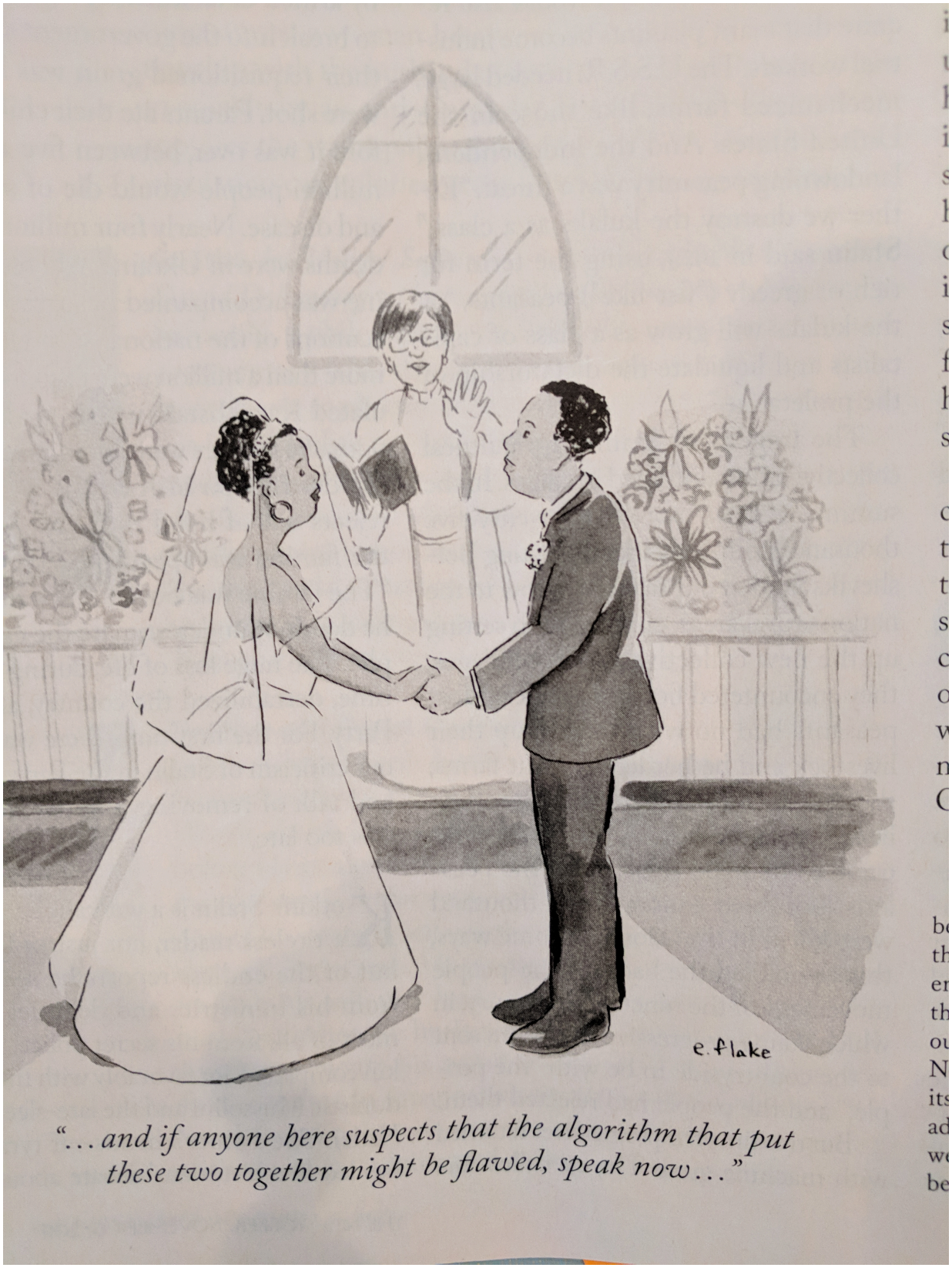
21. (1 point)
Did you base **student.sb3** on some project that came with *Scratch* or that was demonstrated in lecture? If so, which one?

22. (1 point)
In one or two short paragraphs, tell us what you think of *Scratch*. Do you like it? What's good about it? What's bad about it? Did you enjoy implementing **student.sb3**?

23. (1 point)
In a short paragraph, what (if anything) did you learn by using *Scratch*?

24. (1 point)
In implementing **student.sb3**, with what concepts or implementation details did you struggle? Why?

"On the Internet, nobody knows you're a dog."


e.flake
"... and if anyone here suspects that the algorithm that put these two together might be flawed, speak now..."