

Problem Set 1

Starting From Scratch

Part A of this assignment should be completed prior to Noon on Thursday, June 27. There is nothing you have to turn in for Part A.

The rest of this assignment (Parts B and C) must be submitted prior to 3 PM on Tuesday, July 2. You must submit your solution to all questions in parts B and C via upload to the “electronic dropbox” on the Computer Science S-1 course website, <http://www.fas.harvard.edu/~libs1>

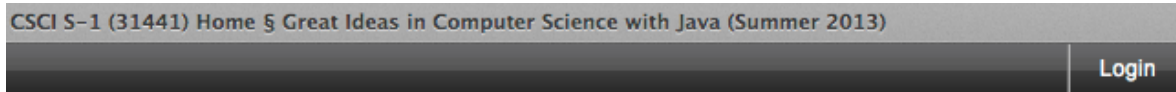
It would be best if you created a “.zip” archive of all your files, so that you will ultimately upload just a single (compressed) file to our website for all of Part B and a separate one for Part C. The *Appendix* describes how to accomplish this.

Part A: Preliminaries (worth 2 points)

You cannot 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 language, 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 the *Scratch* environment; we sincerely hope you will soon discover that programming can be an enormously exciting intellectual activity.

Before doing anything else, be sure to fill out the short, online survey that appears on the front page of our course website: <http://www.fas.harvard.edu/~libs1> If you don't see the *Quick Survey* showing up in your web browser, you probably forgot to login to the site. Click the **Login** button that appears on the upper righthand part of the window:



You will need to know your Harvard ID number and your PIN in order to *Login* to the course website.

1. (1 point)

Next, CREATE YOUR FAS ACCOUNT, if you don't already have one. You can go through this process via the web at: <http://accounts.fas.harvard.edu> Click the link that says “Create a New Account”. You will need to know your Harvard ID number and your PIN in order to complete this process.

2. (0 points)

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:

Join Scratch x


It's easy (and free!) to sign up for a Scratch account.

Choose a Scratch Username

Don't use your real name

Choose a Password

Confirm Password



1

2

3

Next

Ideally, pick a *Scratch Username* that's identical to your FAS account 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:

Join Scratch x

Thanks for joining Scratch!
You're now logged in.

Scratch is a community of all ages, from all over the world. Make sure your projects and comments are respectful and friendly.

Would you like to:
[Learn how to make a project](#)
[Choose a starter project](#)
[Connect with a Scratcher](#)



1

2

3

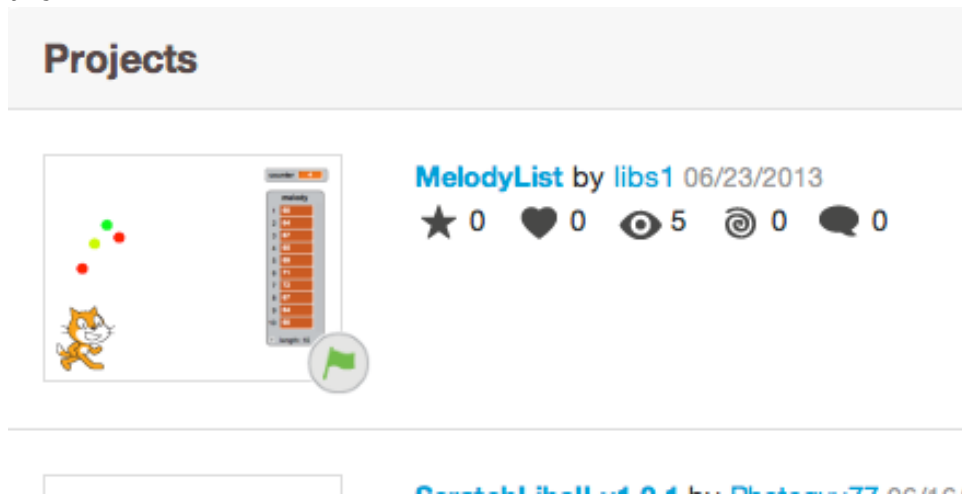
OK Lets Go!

3. (0 points)

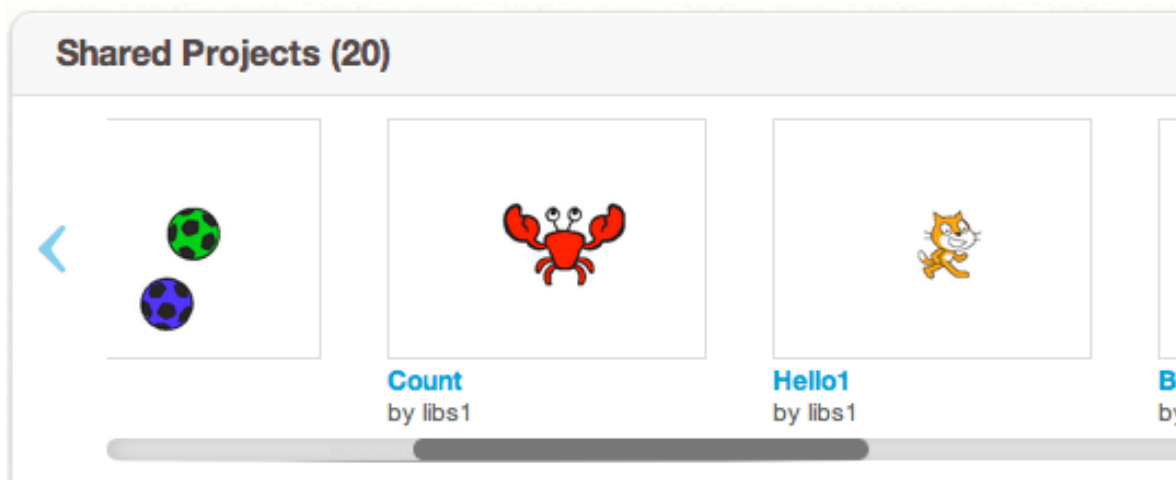
Once your *Scratch* account has been set up and you can *Sign in*, type **libs1** into the search box near the top of the window:



Next, hit the “**return**” key, and you will get a list that looks something like this:



To see all of the **libs1** projects, click any of the blue hyperlinks that say **libs1**, and you will see a scrollable list, similar to the following:



Locate and select *Oscartime*:



Enter “Presentation Mode” by clicking the icon at the far top left of the above window:



Oscartime should fill most of your screen. Click the green flag toward the screen's top-right corner if the program doesn't start playing music right away. Now go ahead and play *Oscartime*. As many times as you'd like, in fact. When done procrastinating, stop the game, if necessary, by clicking the red button toward the screen's top-right corner. To exit “Presentation Mode,” hit **Esc** on your keyboard.

Be sure you scored at least 5 points in this game before quitting!

4. (0 points)

Go ahead and open a few more projects from **libs1**, 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 Scratch’s “Help Screens,” available at http://info.scratch.mit.edu/Support/Help_Screens (You can also click the ? button when creating Scratch programs to get tips on using the different types of block.)

5. (1 point)

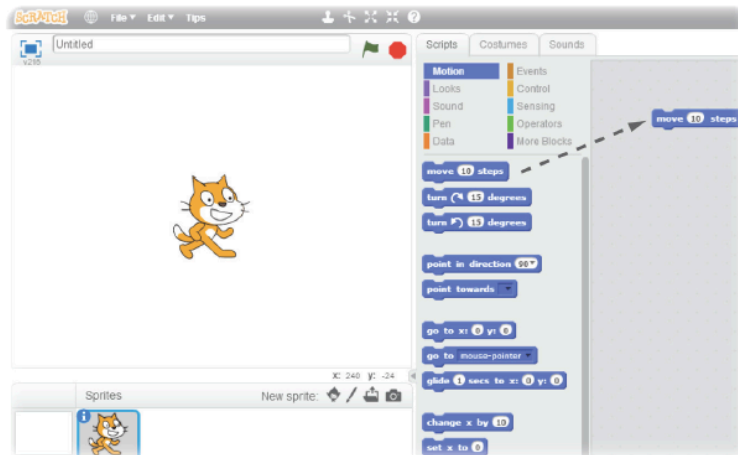
Here’s a point just for making it through so many 0-point questions!

6. (0 points)

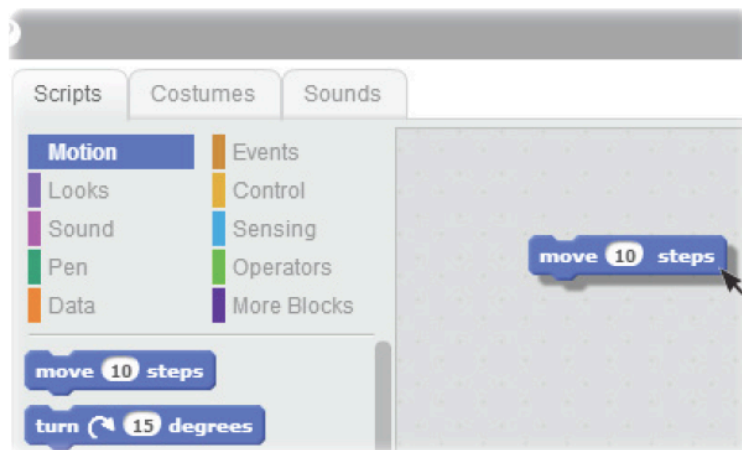
Try the following exercises (all of which are taken from the “Getting Started” document at the MIT Scratch website:

http://scratch.mit.edu/scratchr2/static/_1371843055_//pdfs/help/ScratchGetStarted_beta_draft_Jan2013.pdf

1 Start Moving

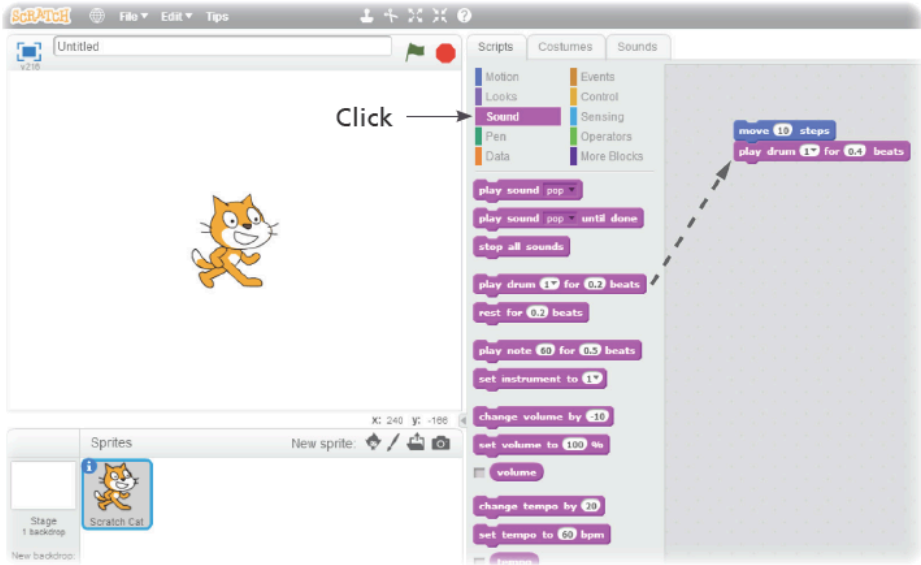


Drag a **MOVE** block into the Scripts area.




Click on the block to make the cat move.

2 Add a Sound



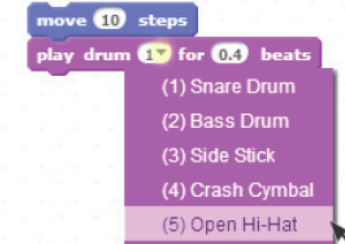
Click →

Drag out a **PLAY DRUM** and snap it onto the **MOVE** block.



Click and listen.

If you can't hear it, check that the sound on your computer is on.



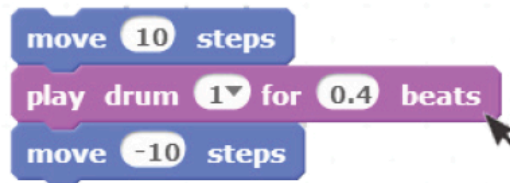
You can choose different drums from the pull-down menu.

Note that in Scratch version 2.0, the **play drum** block looks a little different -- it now has values starting with **1**, not **48**

3 Start a Dance



Add another **MOVE** block. Click inside the block and type in a minus sign.

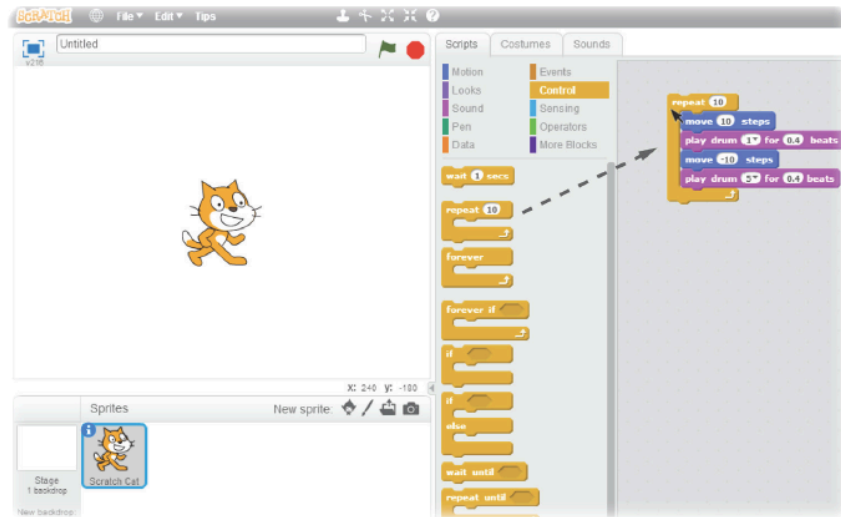


Click on any of the blocks to run the stack.



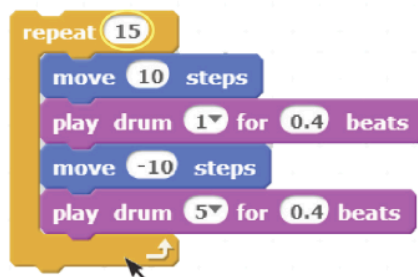
Add another **PLAY DRUM** block, then choose a drum from the menu. Click to run.

4 Again and Again



Drag out a **REPEAT** block and drop it on top of the stack.
You want the mouth of the **REPEAT** to wrap around the other blocks.

To drag a stack, pick it up from the top block.

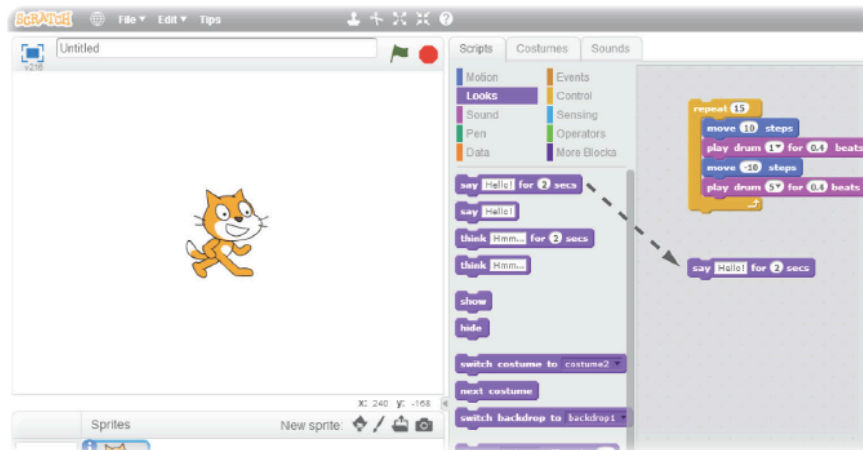


You can change how many times it repeats.

Click to run.

You can click on any block to run a stack.

S Say Something



Click the **LOOKS** category and drag out a **SAY** block.



Click inside the **SAY** block and type to change the words. Click to try it.



Then snap the **SAY** block on the top.

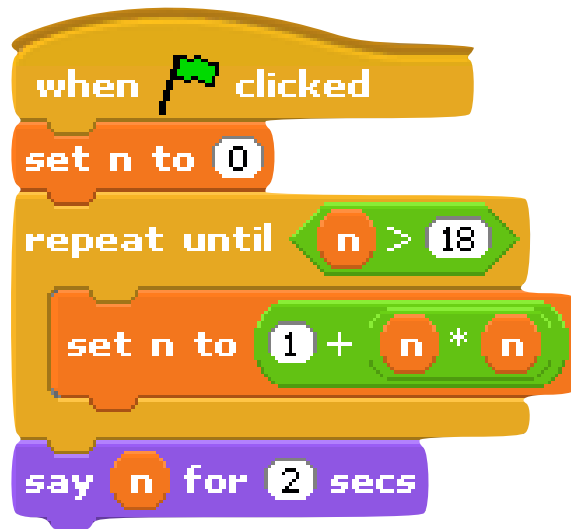
You might consider doing additional exercises at http://scratch.mit.edu/scratchr2/static/_1371843055_//pdfs/help/ScratchGetStarted_beta_draft_Jan2013.pdf

You must submit your solution to all questions in the rest of this problem set via upload to the “electronic dropbox” on the course website. If you don’t see the Dropbox, then you probably forgot to Login to the course website (look at the top righthand corner of the main page of <http://www.fas.harvard.edu/~libs1>)

Part B: Simple Exercises (85 points total)

7. (5 points)

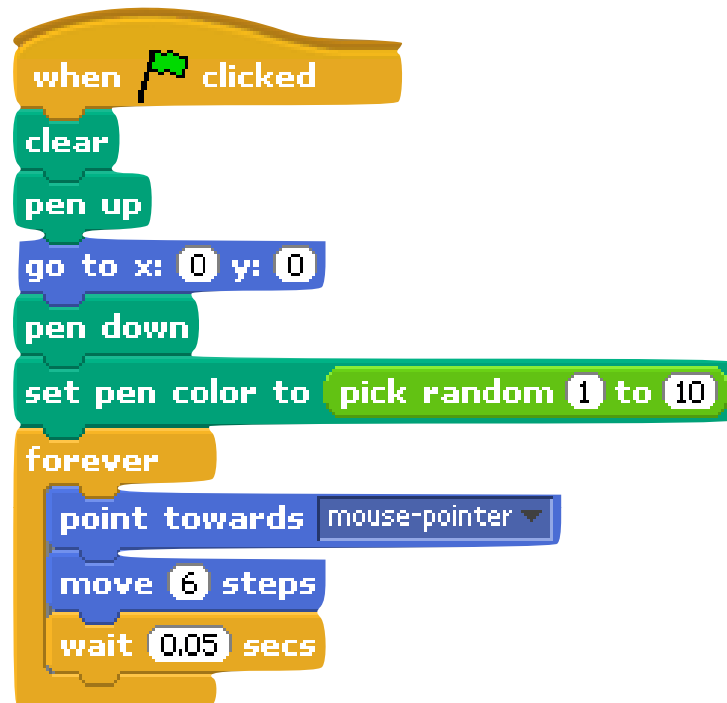
What will the following Scratch program cause the sprite to “say” at the end?



Suppose we change the number 18 to 25 and run the program again? What will the sprite “say” at the end? What if we change the 18 to 100 and run the program? Submit your responses to these questions in the file **problem7.yourFASusername.txt**

8. (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 for that sprite.

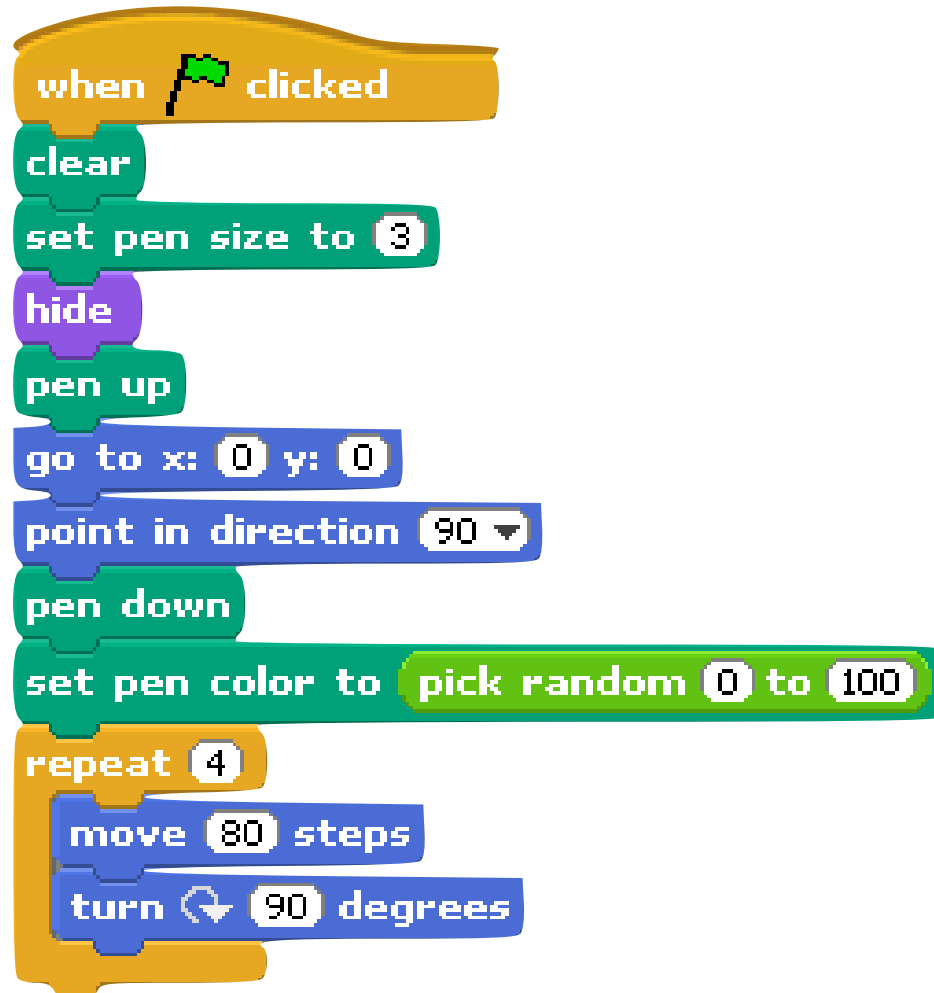


Run the program by clicking the green flag. What happens when you move your mouse around on the stage while the program is running? While the program is executing, you should occasionally press the *space* key and watch what happens.

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. Save this project as **fishDrawing.yourFASusername.sb2**.

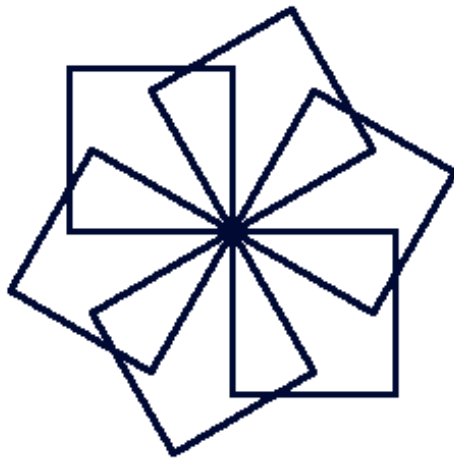
9. (10 points)

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



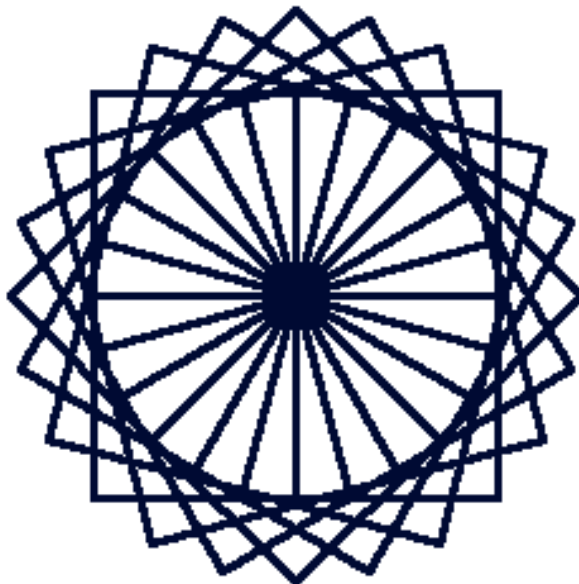
Before executing this project, try and predict what will happen.

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.yourFASusername.sb2** .

10. (10 points total)

Go back to the original Scratch project featured in problem #9. 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.



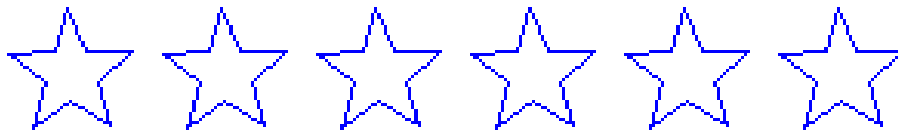
Save this project under the file named **sixSquares.yourFASusername.sb2**.

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.yourFASusername.sb2**.

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 as a file named **sixOtherShapes.yourFASusername.sb2**.

11. (13 points total) + 2 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. Do not worry about the background color or filling in the stars. Save this project under the file named **sixStars.yourFASusername.sb2**.

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,

drawRowOfStars 6 stars

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



Five of the rows have six stars, and four of the rows have five 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 will earn only point if the right number of stars appear with roughly the correct spacing. Save this project as **flagStars.yourFASusername.sb2**.

12. (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.



Save this project as a file named **dice.yourEASusername.sb2**.

13. (7 points)

Write a program that displays an image of a “healthy food” (such as bananas), an “unhealthy food” (such as *Cheetos*), and the value of a variable named **Good Nutrition Points** (whose value starts at zero).



Click on food to eat it.



Your program works as follows: whenever the user clicks the healthy food, the variable gets incremented by 1; whenever the user clicks the unhealthy food, the variable gets decremented by 1. You can use any 2 images you wish if you are not interested in the particular food items shown above.

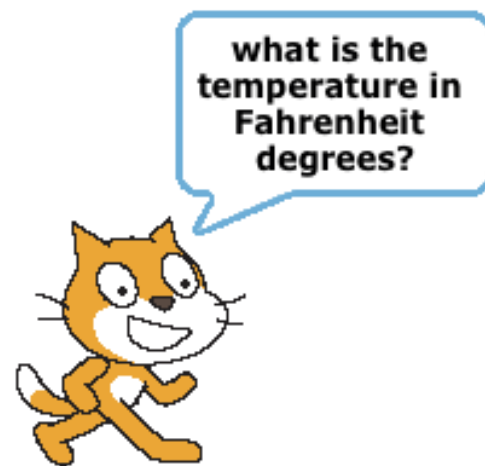
Save the project as a file named **foods.yourEASusername.sb2**

14. (9 points)

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

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

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:



212

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



Save the project as a file named **temperature.yourEASusername.sb2**

15. (9 points)



A man should weigh 106 pounds for the first 5 feet of height, plus 7 pounds for every inch above that; a woman should weigh 100 pounds for the first 5 feet of height, plus 6 pounds for every inch above that.

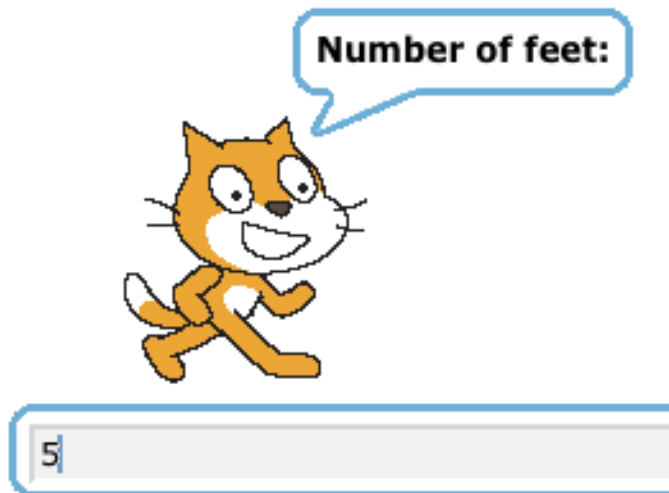
Write a program that determines how much an

individual person should weigh.

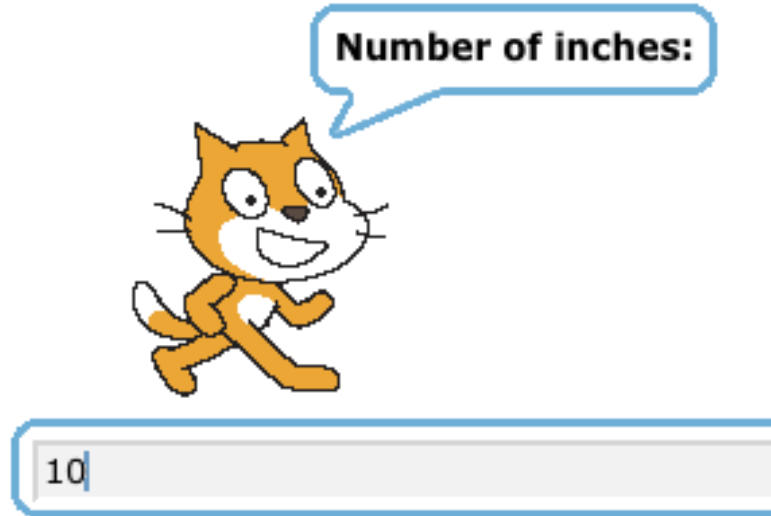
Here is an illustration of what your program might look like in action:



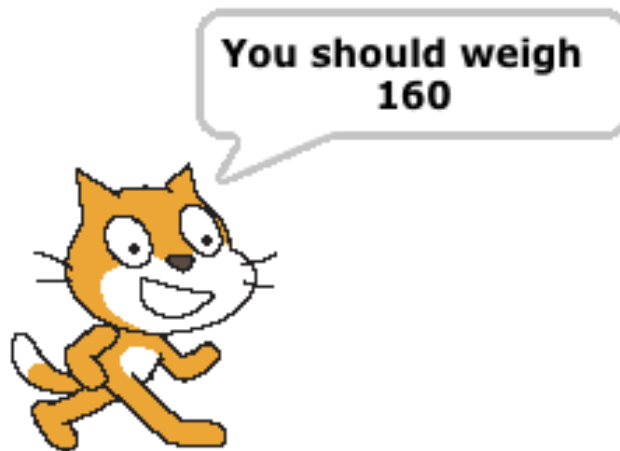
The sprite then says you need to input a *height*. First, in feet:



And then in inches:



And then the output appears:



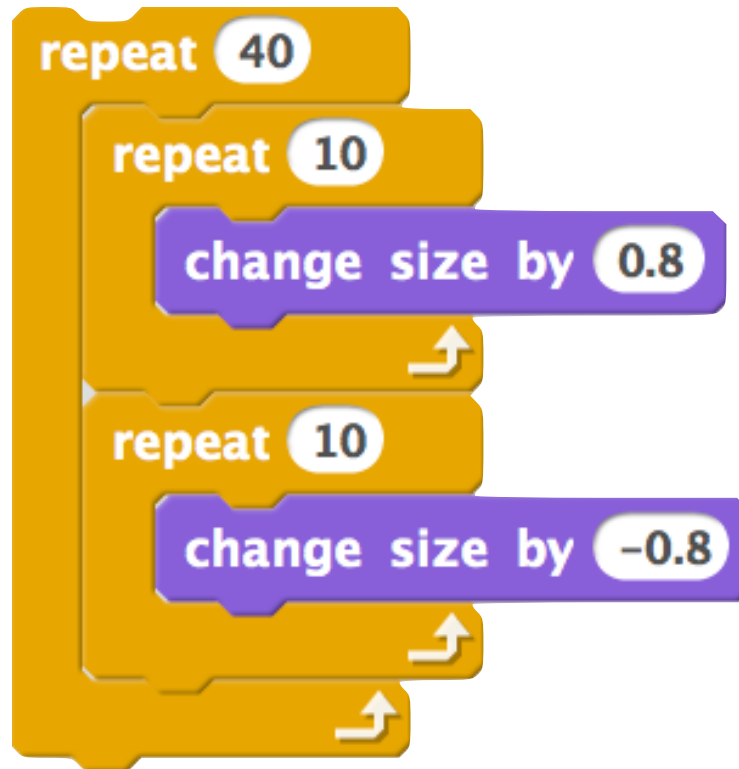
If instead the user indicated he was “male” and was exactly 6 feet tall, the output would be 190 pounds.

If the height that gets input by the user is less than 5 feet, the sprite should complain that an illegal value has been input, and the program should stop at that point. (Remember, there’s a **stop** block in the *control* palette.)

Save the project as a file named **weight.yourFASusername.sb2**

16. (10 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.

Save the project as a file named **pulsate.yourFASusername.sb2**

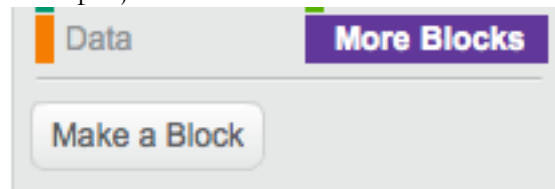
Part C: An Original Project (up to 20 points)

This part of the assignment is REQUIRED for all students who have enrolled for "graduate credit." It is optional for all other students, and may be completed for up to 20 points of "extra credit."

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 **username.sb2**, where **username** is your FAS username.
- ii. Your project must have at least two sprites, neither of which may be a cat.
- iii. Your project must contain a minimum of 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 the **libs1** Scratch account) but can be less complex than *Oscartime*. Probably the **Henry.sb2** boxing example is roughly of the right size. Thus your project should use a few dozen puzzle pieces overall.

Feel free to look through projects that are published on the *Scratch* website 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.

The following questions are optional, but highly recommended!

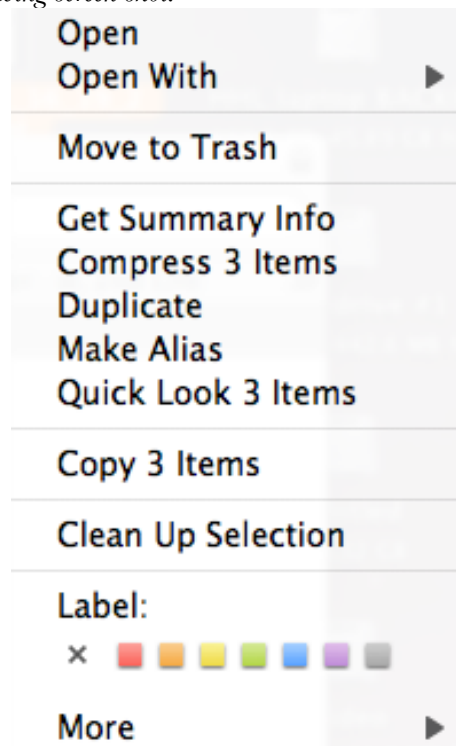
*Your responses to problems 18— 23 must be placed in a plain text file named **questions18To23.yourFASusername.txt** and uploaded to our course website.*

18. (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 new “commenting” feature of Scratch and incorporate typed comments into your actual scripts.
19. (1 point)
Roughly how much time did you spend implementing your **username.sb2** project?
20. (1 point)
Did you base **username.sb2** on some project that came with *Scratch* or that was demonstrated in lecture? If so, which one?
21. (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 **username.sb2**?
22. (1 point)
In a short paragraph, what (if anything) did you learn by using *Scratch*?
23. (1 point)
In implementing **username.sb2**, with what concepts or implementation details did you struggle? Why?

Appendix: Creating “Zip Files”

To create a .zip file is easy. In this example, on a Mac, I selected 3 files with my mouse so they were all highlighted at the same time. Then I held down the “Control” key while clicking these files, and the following contextual menu appeared. At this point, I select the item that says “Compress 3 Items”, and a file named Archive.zip is created. I can easily rename it to whatever I please.

See the following screen shot:



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 something more appropriate before submitting the file. We recommend naming the file yourFASusername.zip for submitting all the files in parts B, C, and D of this assignment. For example, I would name the file compressed file **leitner.zip** if I were submitting this homework.*