

Problem Set 1

Starting From Scratch

Learning Goals

- design of algorithms
- basic programming concepts
 - iteration, conditional execution, lists
- drawing, variables, random #s, real-time interaction

Optional Reading

Majed Marji textbook; scratch.mit.edu website

Timeline

<u>Part</u>	<u>Due Date</u>	<u>Notes</u>
Part A	Thursday, June 22, Noon EDT	Questions 1 through 9
Part B	Saturday, June 24, 3 PM EDT	Problems 10 and 11 should be answered directly on <i>Gradescope</i>
Part C	Saturday, June 24, 3 PM EDT	Upload a .zip file containing solutions to problems 12 through 19 using <i>Gradescope</i>
Part D	Tuesday, June 27, Noon EDT	Required for graduate-credit students, and is "extra credit" for others. Upload using <i>Gradescope</i>

Your submissions for Parts C and D are required to be a “.zip” archive of all your files, so that you will ultimately upload just a single (compressed) file via *Gradescope* for all of Part C and a separate one for Part D. The *Appendix* describes how to accomplish this.

Collaboration Policy Reminder

Warning: *PLEASE DON'T COPY ANY OTHER SOLUTIONS!* It is fine to discuss algorithmic approaches with fellow students and talk about how to solve a problem. It is not OK to look at any other solution before writing or modifying yours, including solutions from the internet.

Depending on the severity, solutions that are inappropriately similar to other students' homework or other source may have their scores divided in two, or may receive a zero score, or may cause the student to be referred to the Harvard College Honor Council.

***P*art A: Preliminaries + Encouragement (just 4 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.

Of course, there *is* one important difference between using 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 few days. The teaching assistants will hold regular office hours both in-person and on Zoom,

and will also be available by appointment, via email and the *Ed* discussion board (which you connect to via our course website).

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!*

1. (1 point)

First, create your **HarvardKEY**, if you have not already done so (most of you probably have it already). You can go through this process via the web at: <http://accounts.fas.harvard.edu> Click the link that says "Claim Your HarvardKey". You will need to know your Harvard ID number in order to complete this process. **Don't forget the password you create** — you will need it to access our course website and other Harvard resources.

2. (2 points)

Before you do anything else, be sure you have already filled out the following short, online survey and section form at:

<http://tinyurl.com/cscis7-2023>

This survey is also available from the main page of the CSCI S-7 course website: <https://canvas.harvard.edu/courses/119412>

Next, click the [Student Locations link](#) on the lefthand side of our course website to give us some idea of where you are located geographically.

3. (0 points)

If not already installed on the computer you're using, download the *Scratch Desktop* software from <https://scratch.mit.edu/download>

If you are using an older computer that does not support *Scratch version 3*, download both *Adobe AIR* and the *Scratch Offline Editor* from the URL: <https://scratch.mit.edu/download/scratch2> Related instructions appear on our course website (navigate to the "Scratch Programming Resources" page using the link on the lefthand side of the website).

4. (0 points)

Whether you are on a Mac or a Windows computer, you should next download the **ScratchLectureExamples** from the [Scratch Programming Resources](#) section of our course website.

After you "uncompress" this *ScratchLectureExamples.zip* file by double-clicking it, you should end up with a *folder* named **ScratchLectureExamples** on your computer's "desktop" or in your Downloads, though the .zip file may decompress automatically.

Feel free to move or copy this **ScratchLectureExamples** folder to a more convenient location on your computer's hard drive.

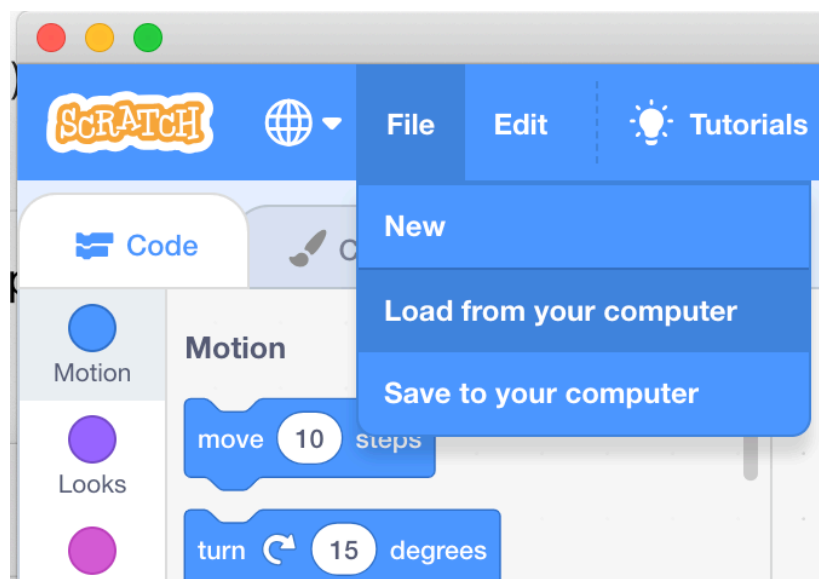
5. (0 points)

Go ahead and launch *Scratch* by double-clicking the application. The icon for version 3 of the software should look like this:



Scratch Desktop

Once the software has started up, click the “**File**” menu, located near the top of the Scratch window, and select the “**Load from your computer**” item as shown:



In the window that appears, you should be able to navigate to the place on your hard drive where you located the *ScratchLectureExamples* folder. Open the file *Oscartime* within the “*ScratchLectureExamples*” folder.

Enter “Presentation Mode” by clicking the rectangular icon with 4 arrows. It appears at the the top righthand side of the the Scratch software window:



Oscartime should fill most of your screen. Click the green flag toward the screen’s top-right corner. Yes, 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!

6. (0 points)

Go ahead and open a few more 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. If you are new to the world of programming, from a web browser you should go to https://en.scratch-wiki.info/wiki/Getting_Started_with_Scratch and try out some of the exercises.

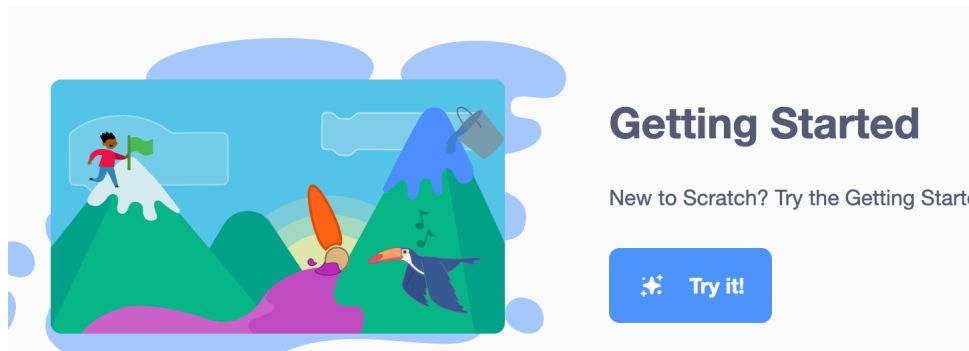
7. (1 point)

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

8. (0 points)

You might also try out the step-by-step intro that is shown near the middle of <http://scratch.mit.edu/help/> or try some of the exercises in

<https://sip.scratch.mit.edu/wp-content/uploads/2020/06/Getting-Started-With-Scratch-3.0.pdf>



9. (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

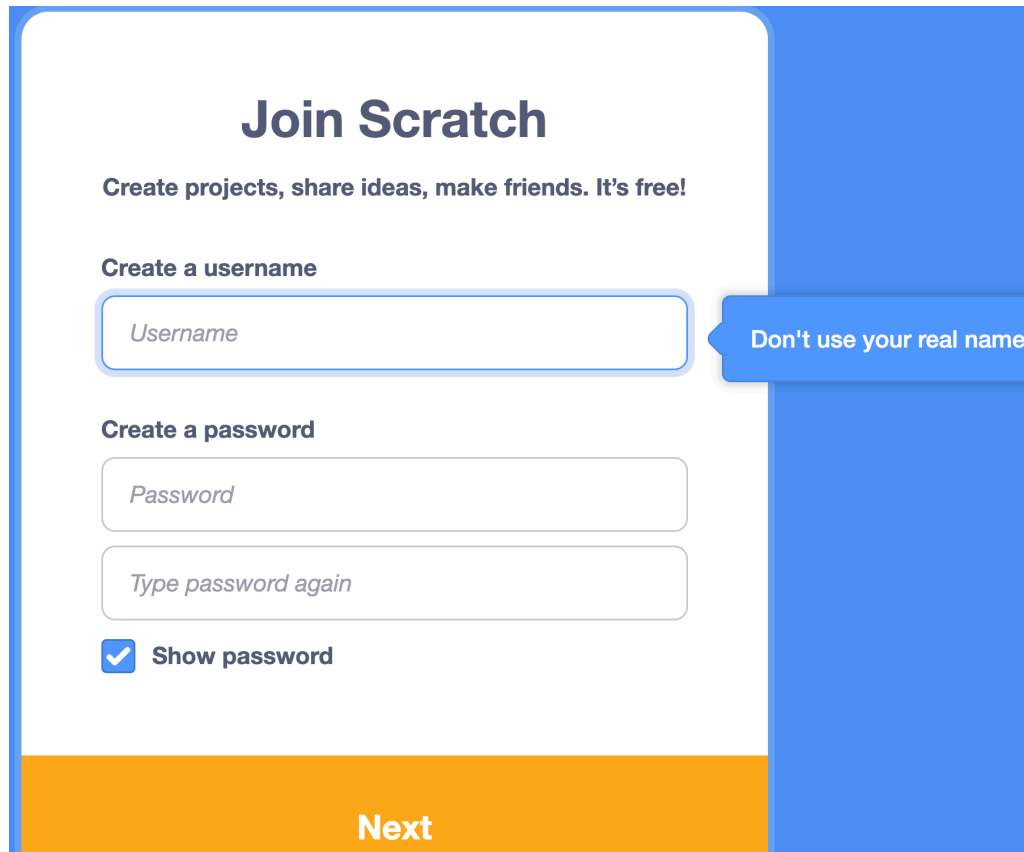
As an alternative to using the **Scratch 2 (or Scratch 3) Offline Editor**, you can instead use *Scratch* via the web, as described below.

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:

The image shows the 'Join Scratch' registration form. It has a white background with a blue border on the left and a blue vertical bar on the right. At the top, the text 'Join Scratch' is in large, bold, dark blue font. Below it, a smaller line of text says 'Create projects, share ideas, make friends. It's free!'. The form is divided into sections: 'Create a username' with a text input field containing the placeholder 'Username' and a blue callout bubble to its right that says 'Don't use your real name'; 'Create a password' with two text input fields, the first containing 'Password' and the second containing 'Type password again'; and a checkbox labeled 'Show password' which is checked. At the bottom, there is a large orange button labeled 'Next'.

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 something like this:

Join Scratch**Thanks for joining Scratch!**

You're now logged in.

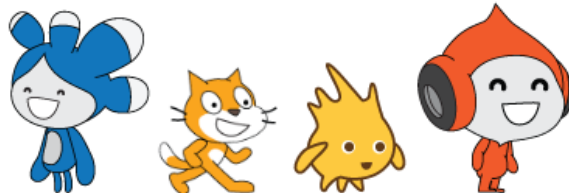
Scratch is a community of all ages, from all over the world. Make sure your projects and comments are respectful and friendly.

Would you like to:

Learn how to make a project

Choose a starter project

Connect with a Scratcher

**OK Lets Go!**

You must submit your solution to all questions in the rest of this problem set via “upload ” to the Computer Science S-7 course website, <https://canvas.harvard.edu/courses/119412>

To do this, you will click the **Gradescope** button, available on the lefthand side of the website’s main page:



CSCI S-7 Section 1 (34533)

2023 Summer

Home

Course Syllabus

Scratch Programming
Resources

Ed Discussion

Gradescope

Course Handouts:
Lecture Slides and
Problem Sets**Recent Announcements****Summer 2023 Vaccination Policy for On-Cam Participation**

Dear Students, If you plan to attend any portic

Introduction to Computer Science

Greetings everyone! In case you have not already done so available in the lefthand navigation part of this website.

Alternatively, you can go directly to *Gradescope* from this link: <https://www.gradescope.com/courses/551624>

You can revise your answers any time before the due date! Go back to your submission and a “Resubmit” button will appear in the bottom-right corner. This will open the problem set up for editing again. We’ll grade the last submission we receive.

A dark teal button with white text that reads "Submit & View Submission" followed by a right-pointing chevron.

Note: 2 points of *extra credit* on this problem set (and all future problem sets) will be awarded if you answer or ask a question in "good faith" using the Ed discussion tool.

Part B: 8 points in total

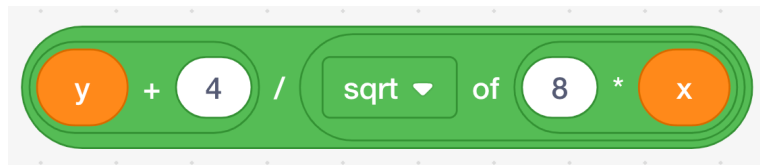
Problems 10 and 11 should be answered directly on Gradescope.

10. (4 points)

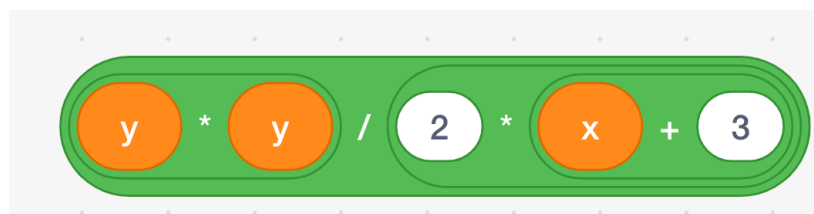
Suppose variable **x** has the value **2** and variable **y** has the value **4**.

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

Part (a) 2 points

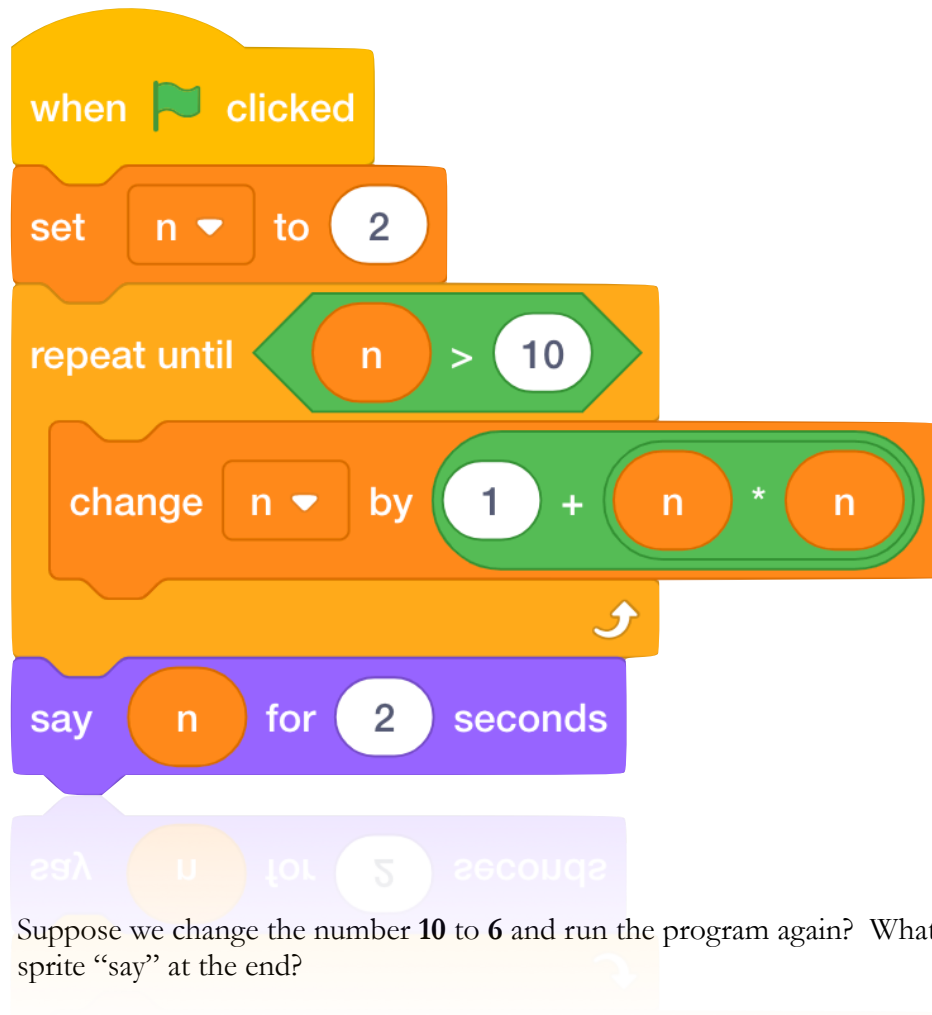


Part (b) 2 points



11. (4 points)

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



Suppose we change the number **10** to **6** 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 **problem11.txt**



Important: For all of the remaining problems, you will lose a point if you name your files differently from what is requested.


Part C: 72 points in total + 2 "extra credit" points


12. (5 points)

Use file name **fishDrawing.sb3**

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. (Note the “erase all” block is named “clear” in Scratch version 2.)

when  clicked



 erase all

 set pen size to 4

 pen up

go to x: 0 y: 0

 pen down

 set pen color to 

forever

point towards mouse-pointer ▼

move 10 steps

wait 0.05 seconds

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

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.

Note that if you do not see the PEN category of blocks, click the bottom lefthand icon that looks like this:



... and then you will discover that you can choose “an extension” that allows you to draw with your sprites!

when space ▼ key pressed

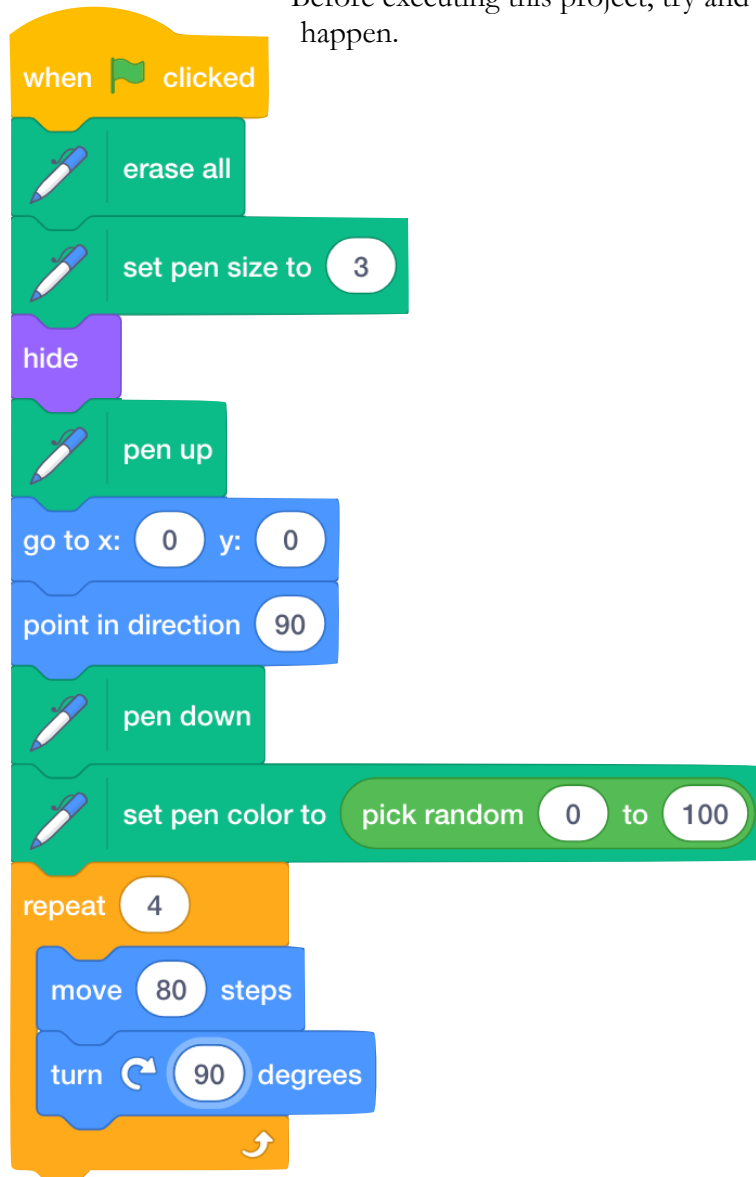
 change pen color ▼ by pick random -10 to 10

13. (10 points) Use file name **rotatingSquares.sb3**

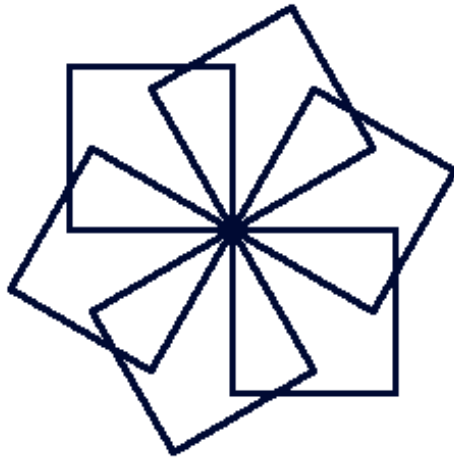
Create a new *Scratch* project, and construct the following program.

BEWARE: There are *two* different "*set pen color to*" blocks: you want to use the second one that takes numbers for input.

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

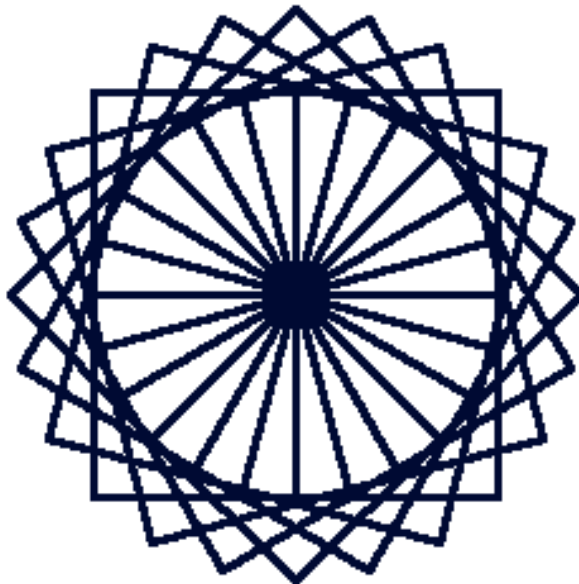


Now modify this program so it can produce output that looks like this:



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.) The program you submit should output precisely 36 squares.

14. (10 points total)

Use file names as mentioned below (3 different ones)

Go back to the original Scratch project featured in problem #13. 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.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.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 as a file named **sixOtherShapes.sb3**.

15. (13 points total) + 2 points of “extra credit”

Use file names as mentioned below

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. You may *Google* the angles for a pentagram. Note also, there is a setting called “turbo mode” under settings that will make the drawing go much faster.

Do not worry about the background color or filling in the stars. Save this project under the file named **sixStars.sb3**.

For extra credit: With the 4th of July holiday coming up soon, try the following. 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 1 point if the right number of stars appear with roughly the correct spacing. Save this project as **flagStars.sb3**.

16. (7 points)

Use file name **dice.sb3**

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 see the die to the right. (We will demonstrate in class how to incorporate different costumes in a Scratch project.)

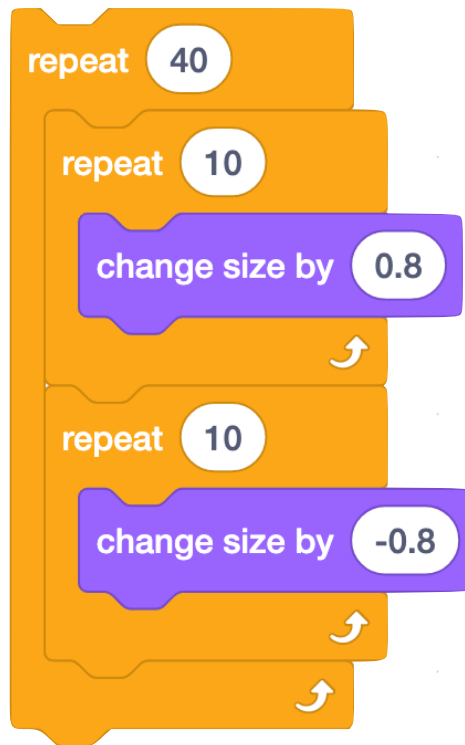


You may use images found online for the die faces.

17. (8 points)

Use file name **pulsate.sb3**

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.

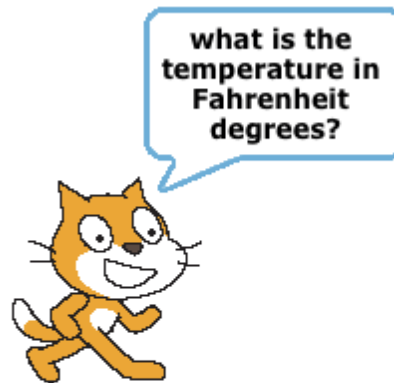
Test **pulsate** out on a variety of input values, and note the effect of making the 3 arguments larger or smaller than what is used in the above example. The 3 actual arguments can be “hard coded,” as opposed to being input from the keyboard.

18. (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:



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



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

19. (10 points) Use file name **weight.sb3**



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

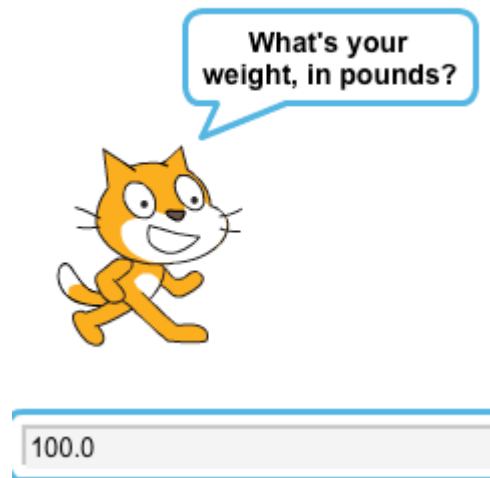
$$\frac{\text{weight (in pounds)}}{\text{height (in inches)}^2} \times 703$$

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

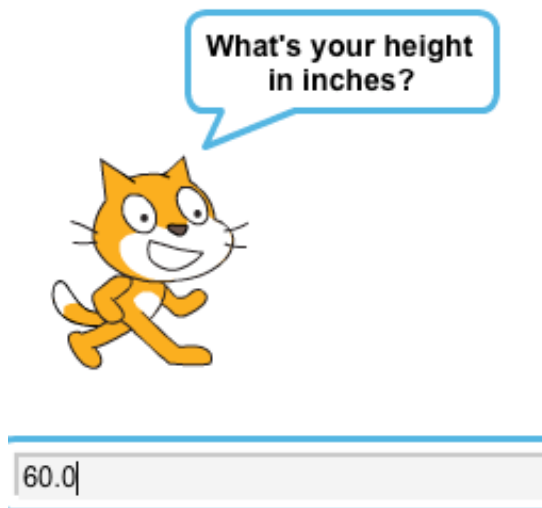
BMI	Weight Status
Below 18.5	underweight
18.5 to 24.9	normal
25.0 to 29.9	overweight
30.0 and above	obese

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:



cases.

You will find it much easier to ask the user for all the information, storing the results in variables before using if conditions to handle the various

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."

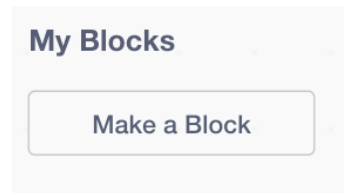
Part D: An Original Project (up to 20 points)

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

20. (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 *yourLastName.sb3*
- 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 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 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.

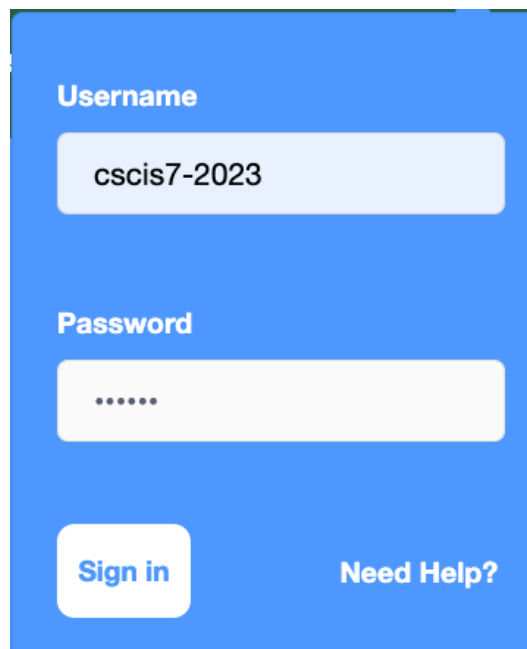
*Your responses to problems 21— 27 must be placed in a plain text file named **questions21To27.txt** and uploaded to our course website via Gradescope.*

21. (0 points)

Consider sharing your project!

Go to <https://scratch.mit.edu>

At the top right of the website, click "Sign in". A dialog box will appear, in which you can type a Username and Password, as shown below:

A screenshot of the Scratch Sign In dialog box. It has a blue background. At the top, it says "Username" in white. Below it is a white text input field containing "cscis7-2023". Below that, it says "Password" in white. Below it is a white text input field containing six asterisks "*****". At the bottom left is a white button with "Sign in" in blue. At the bottom right is the text "Need Help?" in white.

We have set up a special *Scratch* account for this purpose, so please type **cscis7-2023** as the *Username* and type **cscis7** in the box where it asks for your *Password*:

Then click the *Sign in* button.

Next, click the Create button at the top of the page, and you will be taken to the usual Scratch page. Here you can use the File menu to load a program you already created. At this point you can share this project by clicking



Your project will shortly be uploaded to the **cscis7** account. (Note that the password, when you type it, will show up as asterisks on screen.)

22. (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.

23. (1 point)

Roughly how much time did you spend implementing your **.sb3** project?

24. (1 point)

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

25. (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 **.sb3**?

26. (1 point)

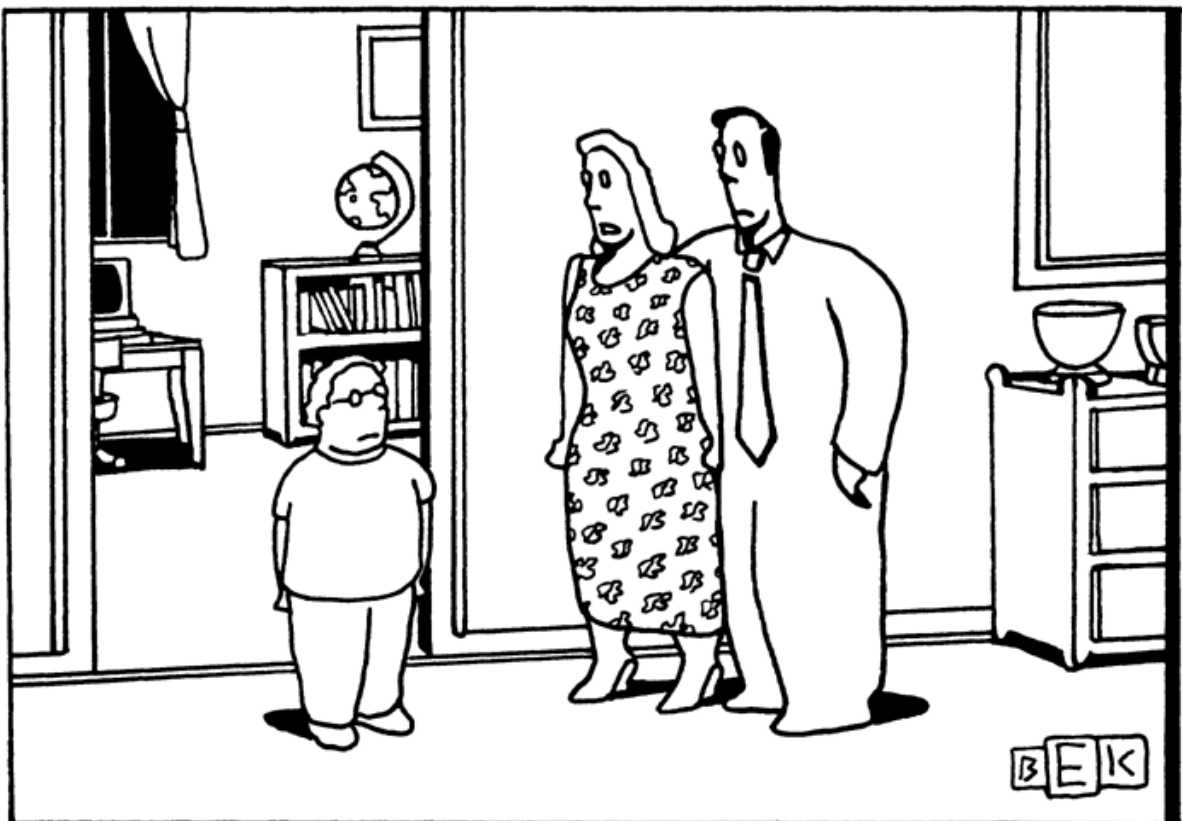
In a short paragraph, what (if anything) did you learn by using *Scratch*?

27. (1 point)

In implementing **.sb3**, with what concepts or implementation details did you struggle? Why?



"User name and password?"

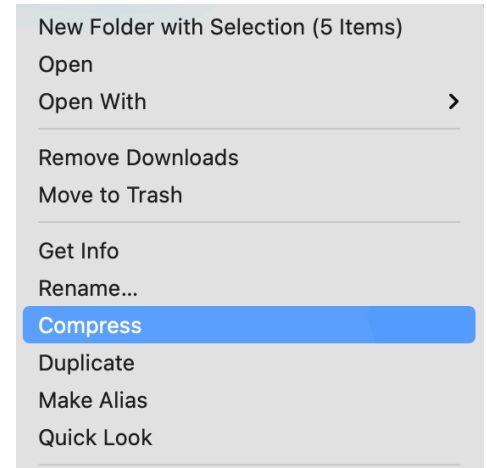


"We're neither software nor hardware. We're your parents."

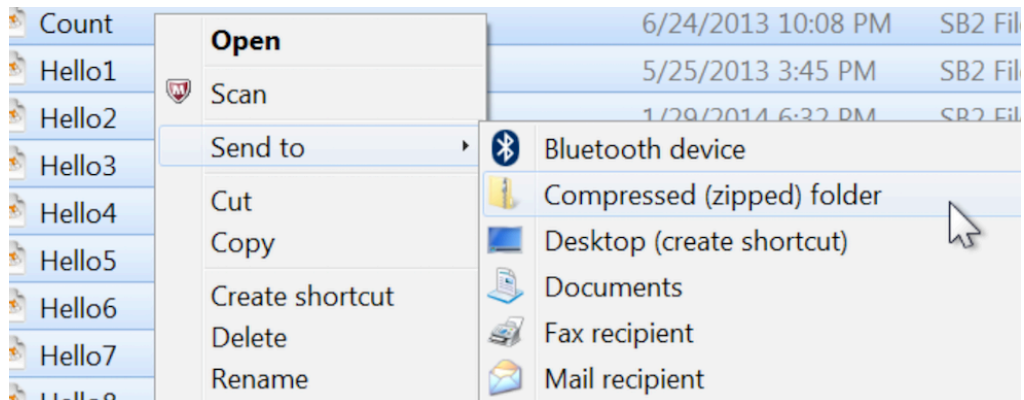
Appendix: Creating “Zip Files”

To create a **.zip** file is easy. In this example, on a Mac running the latest version of OS X (*Ventura*), I selected 3 Scratch 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 screen shot to the right.*



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 compressed file as **yourLASTnamePartC.zip** for submitting all the files in part C of this assignment by Saturday (June 24) at 3 PM, and **yourLASTnamePartD.zip** for Part D of this assignment (to be submitted by Tuesday at Noon). For example, I would name the compressed file **leitnerPartC.zip** if I were submitting the homework for part C.

Don't forget to use **Gradescope** — accessed via our course website — when you are ready to upload your work for Parts C and D.