Programming Problem List
CSci 127: Introduction to Computer Science
Hunter College, City University of New York
Fall 2021

General Notes

- Every program should begin with a comment that includes your name, email and a brief description.

```
#Name: Thomas Hunter
#Email: thomas.hunter1870@myhunter.cuny.edu
#Date: September 2, 2021
#This program prints: Hello, World!

print("Hello, World!")
```

- You are strongly encouraged to submit your assignments before the due date. Below lists the last date on which each assignment will be accepted. Assignments must be **submitted before 6pm** on the date due.
- For flexibility (to allow you to catch up should you be unable to submit work on a particular day/week), due dates are staggered with one assignment due each day starting the second week of classes. **To succeed in this course you should work on assignments the week of the corresponding lab, don't work on an assignment the day it is due!** You don't know how long it will take you to debug a program and you will likely miss deadlines.
- **No late assignments will be accepted.**
- Before submitting an assignment you are free to ask for help in Blackboard drop-in tutoring, Blackboard discussion board and by e-mail (**`cs127uta AT hunter.cuny.edu`**). Please take advantage of this, we offer a lot of support but you must be proactive about it.
- **You must always test your code locally (on your computer) before submitting to Gradescope.** When you run your program on your computer, if there is an error you will get compiler messages that will help you debug your code. You will not get that kind of feedback from Gradescope. When your program runs correctly on your computer, then you can submit.
- For more information on using gradescope, see the Lab 1 .
- While you are encouraged to work with others, **all work submitted must be your own.** As a rule of thumb, **you must do your own typing**. If it is not from the book or class webpage and you did not type it, it is plagiarism. For the first incident, your grade will be a 0 for the assignment (even for cases where you typed the program but others submitted it as their own). For the second incident of cheating or plagiarism, your grade will be a 0 for the homework component of the grade (35% of your overall grade). For the third incident, you will fail the class. We report all incidents to the Office of Student Affairs.

- Learning programming is like learning a foreign language: you will learn more (with less work) if you practice every day. Some of the programs below are easy; some will take more time. We suggest you study the week's Lecture notes and the Lab, then set aside a block of time to work on programming. If you complete the programs for the week you are encouraged to work ahead on next week's batch.
- If Gradescope gives you a "Test Failed: list index out of range" error or "The autograder failed to execute..." it means your program does not run. **Please check that your program runs correctly on your local machine before submitting to Gradescope.**

Submit the following programs via Gradescope:

Lecture / Lab 1

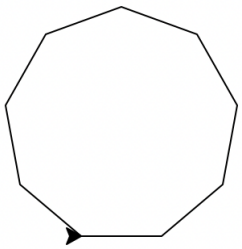
1. **Due Date: 13 September** **Reading: Think CS: Chapters 1 & 2 & Lab1**

   Write a program that prints "Hello, World!" to the screen.

   Hint: See the Lab 1.

2. **Due Date: 14 September** **Reading: Think CS: Chapter 4 & Lab1**

   Write a program that draws a nonagon.



   *Note: Whenever submitting a turtle program, choose a name for your file that is **not** `turtle.py`. When executing the "import turtle" statement, the computer first looks in the folder where the file is saved for the turtle module and then in the libraries (and other places on the path). So, it thinks the module is itself, causing all kinds of errors. To avoid this, name your program something like "myTurtle.py" or "program2.py".*

   Hint: See the Lab 1.

3. **Due Date: 17 September** **Reading: Think CS: Chapter 4 & Lab1**

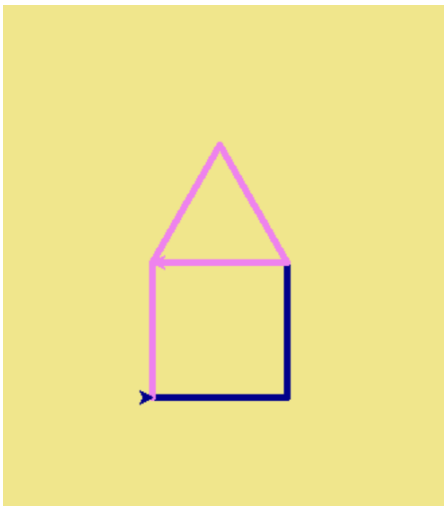
   Copy the program from Section 4.3 into a file on your computer and modify the program

as described below:

```
- change the background to be 'khaki',
- change the turtle 'tess' color to "darkblue",
- set the turtle 'alex' color to "violet" and set its pensize to also be 5

- next, make tess draw a square instead of alex, and have the square be 10

alex should then:
- turn left 90 degrees,
- move forward 100.

Now alex is at the top of the square drawn by tess, and at this point
alex should draw an equilateral triangle which will become the roof of
our house. The triangle should have sides of length 100 to match with
the square drawn by tess.
```

Important: Make sure that your turtle turns the proper number of times, and no more. For example, the instructions the turtle should follow must be the same as the provided square in the assignment, and for the triangle, alex should first turn and then move, so there should be no extra turns after the triangle has been drawn. Gradescope will not test to see if you draw a house. Gradescope will test to see how precisely you followed the above instructions. If it looks like a house and does not get full credit, you may have departed from these instructions. Note the overlapping colors in the image, and make sure yours looks identical!

The result should look as follows:



Note: Make only the modifications indicated here, all other behaviors should stay the same. (i.e. distance forward, pen size etc.)

In programming, it is important to learn to follow instructions precisely. A small difference to you may result in a program that runs incorrectly. This is part of what you are learning here: **follow specifications to the letter!**

4. **Due Date: 20 September** **Reading: Think CS: Chapter 4 & Lab1**

Write a program that implements the pseudocode ("informal high-level description of the operating principle of a computer program or other algorithm") below:

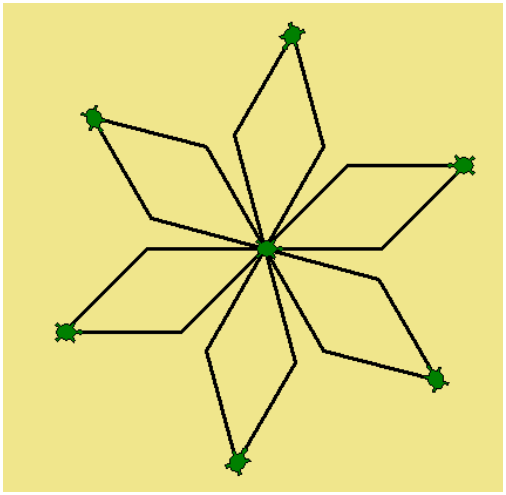
```
import turtle
create a turtle

set the turtle shape to be 'arrow'
set the turtle color to 'purple'
set the turtle pensize to 3

walk forward 30 steps

repeat 3 times:
  walk forward 50 steps
  turn right 120 degrees

walk backward 30 steps
turn right 90 degrees
walk forward 30 steps

repeat 3 times:
  walk forward 50 steps
  turn right 120 degrees

walk backward 30 steps
turn right 90 degrees
walk forward 30 steps

repeat 3 times:
  walk forward 50 steps
  turn right 120 degrees

walk backward 30 steps
turn right 90 degrees
walk forward 30 steps

repeat 3 times:
  walk forward 50 steps
  turn right 120 degrees

walk backward 30 steps
```

Your output should look similar to:

Here is a complete list of turtle commands for your reference.

5. **Due Date: 21 September** **Reading: Think CS: Chapter 4 & Lab1**

Write a program that uses turtle graphics to generate the image below:



Hints: Create the same image shown above. Turtle should have a **pen size of 3** and **move forward 100 steps in the direction it's facing initially before making a turn**.
This assignment aims to challenge you to analyze the image to deduce the process (given a description of the output, write the program to produce it), which is how software developers often approach problems.
So what should you do? Try to puzzle out the image, break down what the turtle does, then apply what you have learned so far about definite loops (for loops) and turtle graphics to reproduce that process. Here some guiding questions:

- How would you create one rhombus first if the wide angle is 45?
- How many turtles do you see? How many repetitions? How many rhombuses do you need to draw?
- What does the turtle do in each repetition?
- What direction is the turtle facing before it stamps? Does it stamp before or after it turns?
- Once you figured out the answer to the previous two questions, play around with the angle to produce the image.
- Look up the turtle functions `pencolor()` and `fillcolor()` to produce the turtle colors?
- You have used that background color before, what color was it?

Hope this helps!!!

Here is a complete list of turtle commands for your reference.

Lecture / Lab 2

6. **Due Date: 22 September** **Reading: Think CS: Chapters 2 & 9 & Lab 2**

Using the string commands introduced in Lab 2, write a Python program that **prompts the user for a message and a letter**, and then prints **that message**, the message in upper case

letters and the message in lower case letters. Finally, it prints out how many times the **given letter** is found in the message.

A sample run of your program should look like:>

```
Enter a message:  Mihi cura futuri
Enter a letter: i
Mihi cura futuri
MIHI CURA FUTURI
mihi cura futuri
3
```

Another sample run:

```
Enter a message:  I love Python!
Enter a letter: s
I love Python!
I LOVE PYTHON!
i love python!
0
```

Hint: Your program should be able to take any phrase and letter the user enters. To do that, you need to store the phrase and the letter in a variable, then print variations of the stored variable as well as how many times the letter appears in the phrase.

7. **Due Date: 23 September** **Reading: Think CS: Chapters 2 & 9 & Lab 2**

Combining `ord()` and `chr()` write a program that:

- Prompts the user to enter a message
- Prints each character in the message, and the character corresponding to the unicode + 5.

A sample run of your program should look like:

```
Enter a message:  abc
a shifted by 5 characters is: f
b shifted by 5 characters is: g
c shifted by 5 characters is: h
```

Another run:

```
Enter a message: Hello
H shifted by 5 characters is: M
e shifted by 5 characters is: j
l shifted by 5 characters is: q
l shifted by 5 characters is: q
o shifted by 5 characters is: t
```

8. **Due Date: 24 September** **Reading: Think CS: Section 4.7 & Lab2**

Write a program that prints out the numbers 2-20 incrementing by 2. Next to each number, print out the number + 10, separated by a '|' symbol and formatted as follows:

The output of your program should be:

```
current value: 2 | value+10: 12
```

```
current value: 4 | value+10: 14
current value: 6 | value+10: 16
current value: 8 | value+10: 18
current value: 10 | value+10: 20
current value: 12 | value+10: 22
current value: 14 | value+10: 24
current value: 16 | value+10: 26
current value: 18 | value+10: 28
current value: 20 | value+10: 30
```

Hint: *Use the* `start, stop, step` *options in the* `range()` *function* and print the loop variable.

9. **Due Date: 27 September** **Reading: Think CS: chapter 9, Section 10.24 & Lab 2**

   Write a program that does the following:

   1. Prompt the user for a phrase.
   2. Reverse the phrase and print it.
   3. Make the phrase upper case.
   4. Split the phrase into a list of words.
   5. Take the **last letter** of each word (by looping through the list of words) and print it on the same line.

   A sample run of your program should look like:

```
Enter a phrase:  City University New York
Reversed phrase:  kroY weN ytisrevinU ytiC
Last letters of reversed words:  YNUC
```

   Another run:

```
Enter a phrase:  Hunter College
Reversed phrase: egelloC retnuH
Last letters of reversed words: CH
```

10. **Due Date: 28 September** **Reading: Think CS: Chapters 2 & 9 & Lab 2**

(The cipher disk above shifts 'A' to 'N', 'B' to 'O', ... 'Z' to 'M', or a shift of 13. From secretcodebreaker.com.)

Write a program that prompts the user to enter a word and then prints out the word with each letter **shifted right by 13** characters.

Assume that all inputted words are in **upper case letters**: 'A',...,'Z'.

A sample run of your program should look like:

```
Enter a word: ZEBRA
Your word in code is MROEN
```

Hint: *See the example programs from Lecture to find out how to wrap around (Lecture slides can be found on the course outline).*

Lecture / Lab 3

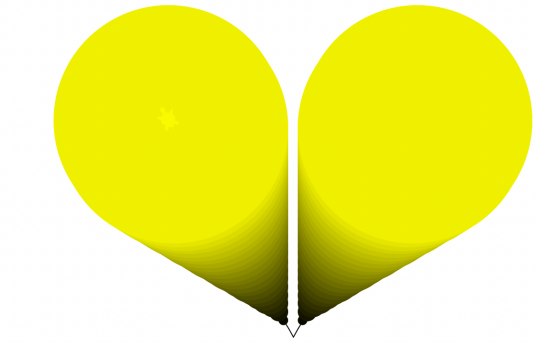
11. **Due Date: 29 September** **Reading: Think CS: Chapter 4 & Lab 3**

    Modify the program from Lab 3 to show the shades of yellow in the shape of a pseudo-heart. The turtle draws on a 60 degree angle and then a 120 degree angle respectively from the start. Once you draw one side, go back to where you started, without drawing, to draw the other side. Remember to use `penup()` and `pendown()`.

    Follow the sequence of actions:

    - Turn left
    - Move forward while changing color and pen size
    - Lift the pen up
    - Change the pen size and color back to 0
    - Move backward 260
    - Turn left
    - Put the pen down
    - Move forward while changing color and pen size

    Your output should look similar to:

    Note: when we say "Modify the program..." only make the requested modifications, everything else is assumed to be keept as is.

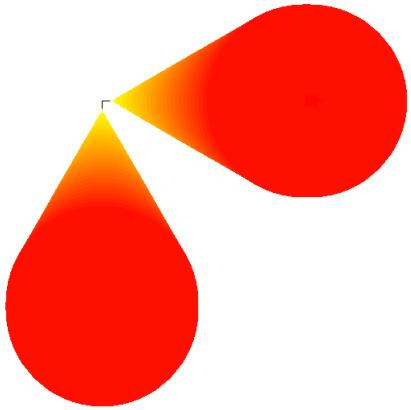
12. **Due Date: 30 September** **Reading: Think CS: Chapter 4 & Lab 3**

Now modify the program from Lab 3 to create two transitions from shades of yellow to red by moving forward 10 steps each.

Follow the sequence of actions:

- Move forward while changing color and pen size
- Move backward with pen up
- Turn right
- Move forward while changing color and pen size

Your output should look similar to:



Hint: *decrease the amount of green using the loop variable while keeping the amount red the same.*

13. **Due Date: 1 October** **Reading: Think CS: Section 10.24, Lab 2 & Lecture 3**

Write a program that prompts the user to enter a list of books and their authors. Each book with its respective author is separated from the next by a semicolon followed by a space ("; "). The names are entered `firstName lastName,` (i.e. separated by space " "). The title of the book is separated from the the author by a hyphen surrounded by spaces (-). Your program should then print out the name of the book, one per line, followed by the string "by", followed by the the first name initial and last name initial of the author both followed by "."

A sample run of your program should look like:

```
Enter a list of books and their authors:Frankenstein - Mary Shelley; Lucy -
Frankenstein by M.S.
Lucy by J.K.
Beloved by T.M.
Annabel by K.W.

Thank you for using my book organizer!
```

Hint: See Section 10.24 for a quick overview of `split()`*. Do this problem in parts: first, split the list by each book and author (what should the delimiter be?). After that, split the name of the book from the author. Then, split each person's name into first and last name*

(what should the delimiter be here?). If you have a string `str`*, how do you index to extract the first character?*

14. **Due Date: 4 October** **Reading: Section 10.24, Lab 2 & Lab 3**

Write a program that asks the user for a 6-digit hex number and uses it as the hex code to stamp 4 turtles of that color into a square
Let the turtle walk forward 100 steps between stamps and use the sequence turn/forward /stamp.

A sample run of the program:

```
Please enter a 6-digit Hexadecimal number: 00FFFF
```

which produces an output:

Hint: *don't forget to add the '#' at the beginning of the hex number to use it as a hex encoding for color! For string concatenation see Lab 2. You may assume that the user always inputs a valid 6-digit Hex number.*

15. **Due Date: 5 October**
Reading: Think CS: Section 8.10 & Datacamp Numpy Tutorial & Lab 3

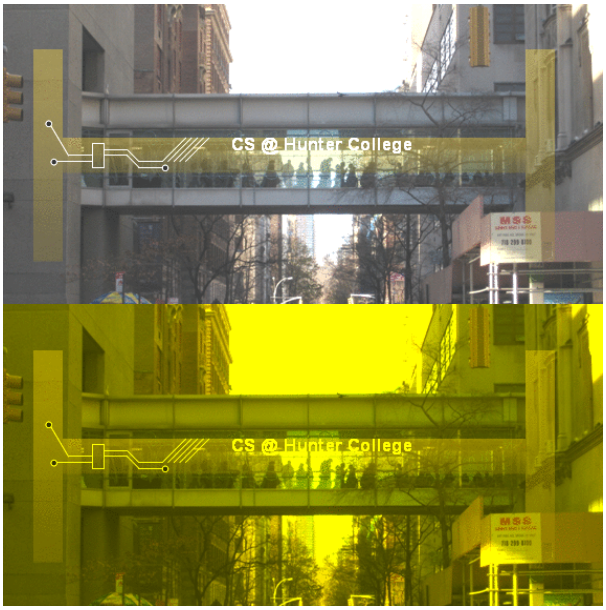
Write a program that asks the user for the name of an image `.png` file and the name of an output file. Your program should create a new image that has only the green and red channels of the original image (that is, no blue channel).

To test your program you can download csBridge.png.

A sample run of your program should look like:

```
Enter name of the input file:  csBridge.png
Enter name of the output file:  noblue.png
```

Sample input and resulting output files:



IMPORTANT: before submitting your program for grading, **remove the commands that show the image** (i.e. the ones that pop up the graphics window with the image). The program is graded on a server on the cloud and does not have a graphics window, so, the `plt.show()` and `plt.imshow()` commands will give an error. Instead, the files your program produces are compared pixel-by-pixel to the answer to check for correctness.

Hint: See Lab 3.

Lecture / Lab 4

16. **Due Date: 6 October** **Reading: Think CS: Chapter 2 & Lab 4**

    The parsec is a unit of length used to measure the large distances to astronomical objects outside the Solar System, approximately equal to 3.26 light-years, i.e. 30.9 trillion kilometres (19.2 trillion miles).

    For this program, you'll be doing a very common conversion: parsecs to light-years, and light-years to parsecs. You will use the formula: *1 parsec = 3.26156 light-years*.

    With this information, write a program that does the following:

    1. Asks the user which conversion they want to do:

    ```
    enter 'a' for Light-Year to Parsec,
    enter 'b' for Parsec to Light-Year.
    ```

    2. If the user enters the character 'a', the program will prompt for the number of lightyears to be converted and computes light-years to parsecs; if the user enters the character 'b' instead, the program will prompt for the number of parsecs to be converted and computes parsecs to light-years.

    3. Then the program will print the appropriate result.

A sample run of your program should look like:

```
Please enter the conversion you want to do:
'a' for Light-Year to Parsec conversion
'b' for Parsec to Light-Year conversion
Your selection: a
Please enter a number of Light-Years: 3
3.0 Light-Years is equal to 0.919803 Parsecs.

### Another run of the program:

Please enter the conversion you want to do:
'a' for Light-Year to Parsec conversion
'b' for Parsec to Light-Year conversion
Your selection: b
Please Enter a number of Parsecs: 1
1.0 Parsecs is equal to 3.26156 Light-Years.
```

Hint *Don't forget to convert the input into a numeral for calculation.*

17. **Due Date: 7 October Reading: Think CS: Chapter 4 and Chapter 7.4, Lab1 & Lab 4**

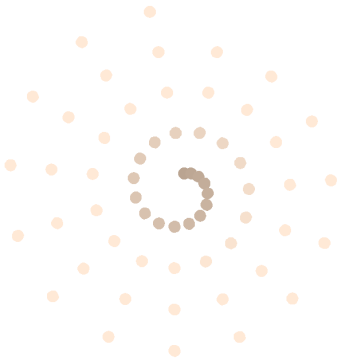
Write a program that implements the following pseudocode using turtles:

```
ask the user for number of stamps
create a turtle
set colormode to 255
set the turtle shape to 'circle'
lift the turtle pen up
set steps to 10, red to 186, green to 164, and blue to 145
set turtle color to (red, green, blue)

repeat stamps times:
    stamp
    increment steps by 2
    if adding 3 to red, green, and blue does not exceed 255:
      add 3 to red, green, and blue
    set turtle to new color
    have turtle move forward by steps
    have turtle turn right 24 degrees
```

A sample run of your program should look like:

```
Enter number of stamps the turtle will print: 50
```

and the output should look similar to:



For your reference, here is a complete list of turtle commands.

Suggested: Set turtle speed to 0.

18. **Due Date: 8 October**

Reading: Think CS: Datacamp Numpy Tutorial, Lab 3 & Lecture 4

In Lecture 4, we designed a program to make a Hunter logo 'H' on a 10x10 grid. Write a program that creates a pink 'P' logo for Python on a 30x30 grid.

The grading script is expecting:

- The file to be saved as: `logo.png`.
- The grid to be 30 x 30.
- The 'P' to be 100% red, 0% green, and 100% blue. It is best to sketch this before starting to code. The body and the head thickness of the "P" is 1/5 of the grid.
- The lower portion of the head of the P (lower horizontal P-region) begins at the lower 1/2 of the image (row 15).
- The remaining pixels in the image should be black (0% red, 0% green, and 0% blue).

Note: before submitting your program for grading, **remove the commands that show the image** (i.e. the ones that pop up the graphics window with the image). The program is

graded on a server on the cloud and does not have a graphics window, so, the `plt.show()` and `plt.imshow()` commands will give an error. Instead, the files your program produces are compared pixel-by-pixel to the answer to check for correctness.

Hint: See notes from Lecture 4. (Lecture slides can be found on the course outline)

19. **Due Date: 12 October** **Reading: Think CS: Section 2.7 & Lab 4**

Write a program that implements the pseudocode below:

```
Ask the user for amount of social media followers they have.
Compute and output the their influencer tier as follows:
  - Error if followers less than 0
  - Hyper Influencer if number of followers less than 1K followers
  - Nano Influencer if number of followers 1K-10K followers (not including 1
  - Micro Influencer if number of followers 10K-100K followers (not includin
  - Mid-Tier Influencer if number of followers 100K-500K followers (not incl
  - Macro-Influencer if number of followers 500K-1M followers (not including
  - Celebrity Influencer if number of followers greater than 1M followers
```

Think about the order of your decisions: **elif/else clauses are considered only if the previous condition is false!**

A sample run of your program should look like:

```
Enter amount of social media followers: 150
Your influencer tier is: Hyper Influencer
```

and another sample run:

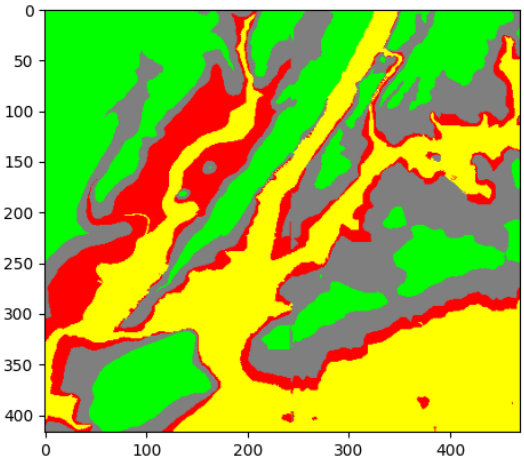
```
Enter amount of social media followers: -200
Your influencer tier is: Error
```

Hint: See Section 7.7.

20. **Due Date: 13 October**
Reading: Think CS: Chapters 2 & 8.10 & Datacamp Numpy Tutorial & Lab 4

Modify
the flood map of NYC from Lab 4 to color the region of the map with elevation greater than 5 feet and less than or equal 20 feet above sea level the color grey (50% red, 50% green, and 50% blue). Also, change the blue water to yellow

Your resulting map should look like:



and be saved to a file called `floodMap.png`.

Note: before submitting your program for grading, remove the commands that show the image (i.e. the ones that pop up the graphics window with the image). The program is graded on a server on the cloud and does not have a graphics window, so, the `plt.imshow()` and `plt.show()` commands will give an error. Instead, the files your program produces are compared pixel-by-pixel to the answer to check for correctness.

Lecture / Lab 5

21. **Due Date: 14 October**
Reading: Think CS: Section 8.10 & Datacamp Numpy Tutorial, Lab 3, Lab 4 & Lecture 4
REVIEW from Lecture/Lab 4

    Write a program that creates an image of khaki and rosy horizontal and vertical stripes.

    The color codes are below:

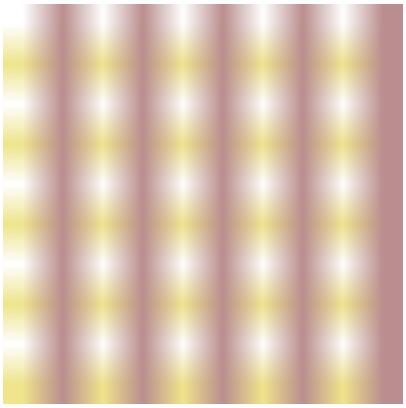
    ```
    RGB(188,143,143), #BC8F8F - Rosy Brown
    RGB(240, 230,140) #F0E68C - khaki
    ```

    You can use a color converter to get the RGB% from hex

    Your program should ask the user for the size of your image, the name of the output file, and create a `.png` file of stripes. For example, if the user enters 10, your program should create a 10x10 image, alternating between khaki and rosy brown stripes.

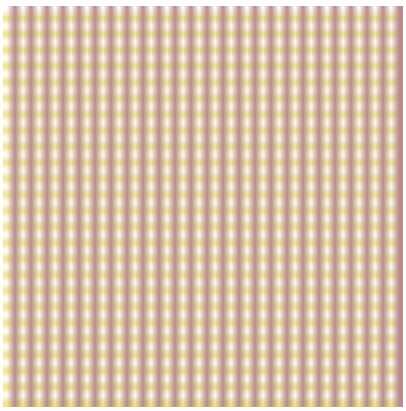
    A sample run of the program:

    ```
    Enter the size: 10
    Enter output file: 10stripes.png
    ```



Another sample run of the program:

```
Enter the size: 50
Enter output file: 50stripes.png
```



Note: before submitting your program for grading, remove the commands that show the image (i.e. the ones that pop up the graphics window with the image). The program is graded on a server on the cloud and does not have a graphics window, so, the `plt.show()` and `plt.imshow()` commands will give an error. Instead, the files your program produces are compared pixel-by-pixel to the answer to check for correctness.

Hint: See notes from Lecture 4.

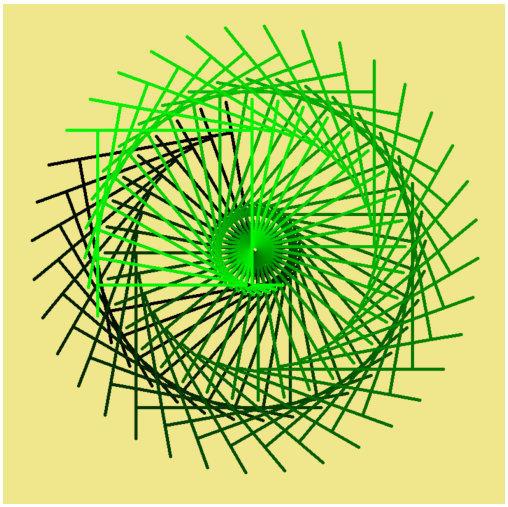
22. **Due Date: 15 October** **Reading: Think CS: Chapter 4, Lab1, Lab 2 & Lab 4**

Write a program that implements the pseudocode ("informal high-level description of the operating principle of a computer program or other algorithm") below for drawing a spiral of squares using turtles:

```
Set the turtle colormode to 255
(optional) set the turtle speed to 0 to make things faster
set the pensize to 5
set the turtle background color to "khaki" for aesthetic purposesof course

loop 36 times:
    change the pencolor to be 0 red, (i*7) green, and 0 blue
    move the turtle forward 10
    turn the turtle left 10
    loop 4 times:
```

```
turn left 90
move forward 300
move backward 50
```

Your output should look similar to:

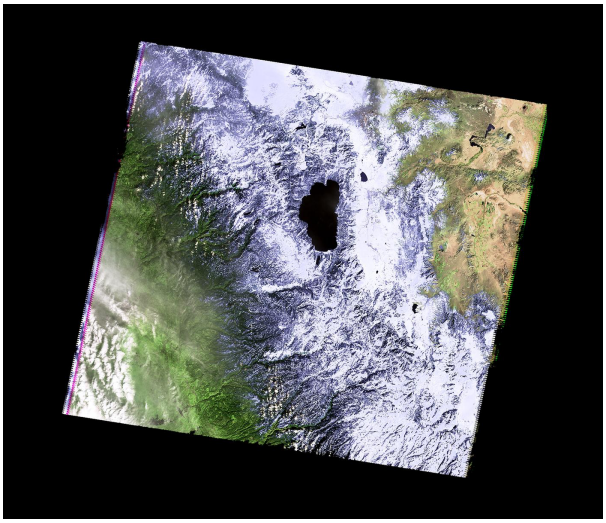


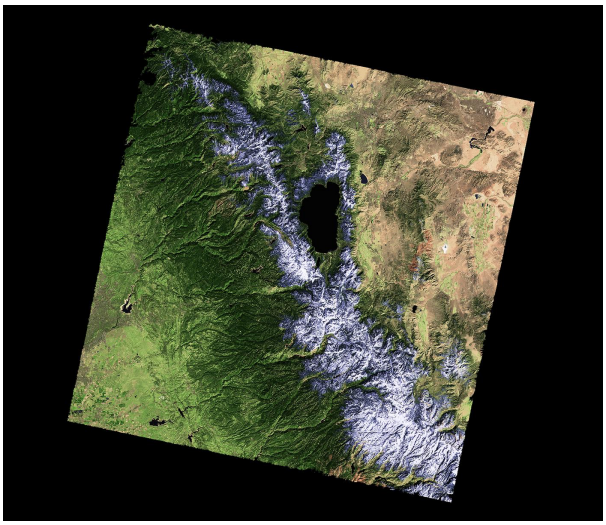
23. **Due Date: 18 October**

Reading: Think CS: Chapters 7 & 8.10 & Datacamp Numpy Tutorial & Lab 5

Following Lab 5, write a program that asks the user for the names of two `.png` files and a threshold for the white pixels. Then, print the number of pixels that are nearly white (the fraction of red, the fraction of green, and the fraction of blue above the threshold) for each image. Calculate the difference between the snow counts and print them out.

For example, if your images was of the snow pack in the Sierra Nevada mountains in California in February 2011 and in February 2014:



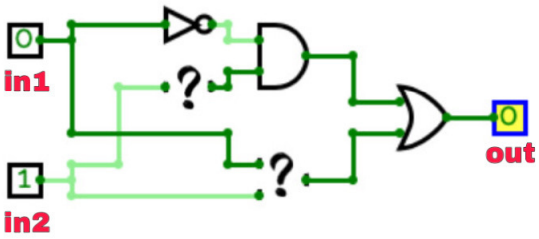


then a sample run would be:

```
Enter first image file name:  feb2011.png
Enter second image file: feb2014.png
Enter threshold of white pixels: 0.8
Snow count of first image: 228663
Snow count of second image: 31171
Difference between first and second image: 197492
```

24. **Due Date: 19 October** **Reading: Burch's Logic & Circuits & Lab 5**

Given the incomplete circuit and the truth table, provide a logical expression that matches the circuit and the table.



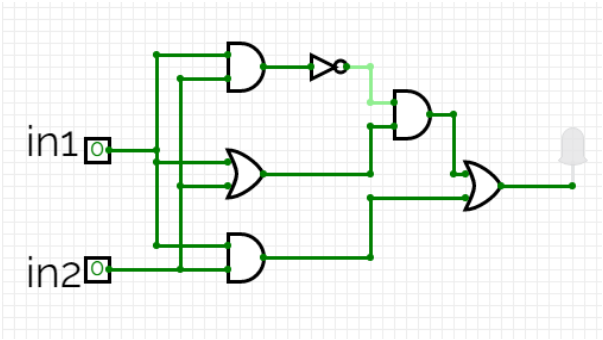
Here is the truth table for the `mystery circuit` operator:

in1	in2	in1 ___ in2
True	True	True
False	True	False
True	False	False
False	False	True

Save your expression to a text file. See Lab 5 for the format for submitting logical expressions to Gradescope.

25. **Due Date: 20 October** **Reading: Burch's Logic & Circuits**

Write a logical expression that is equivalent to circuit shown below. Use `in1`, `in2` for the inputs. An example is shown in Lab 5.



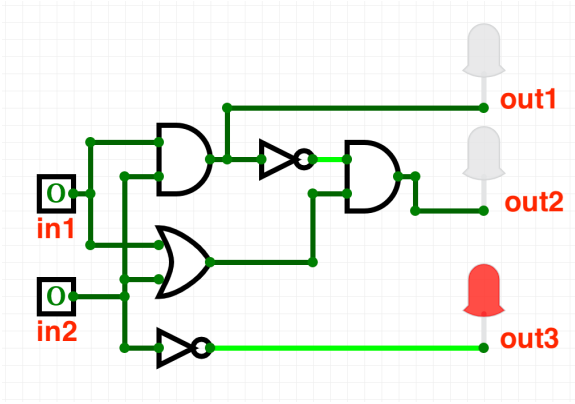
Save your expression to a text file. See Lab 5 for the format for submitting logical expressions to Gradescope.

Lecture / Lab 6

26. **Due Date: 21 October** **Reading: Burch's Logic & Circuits & Lab 5**

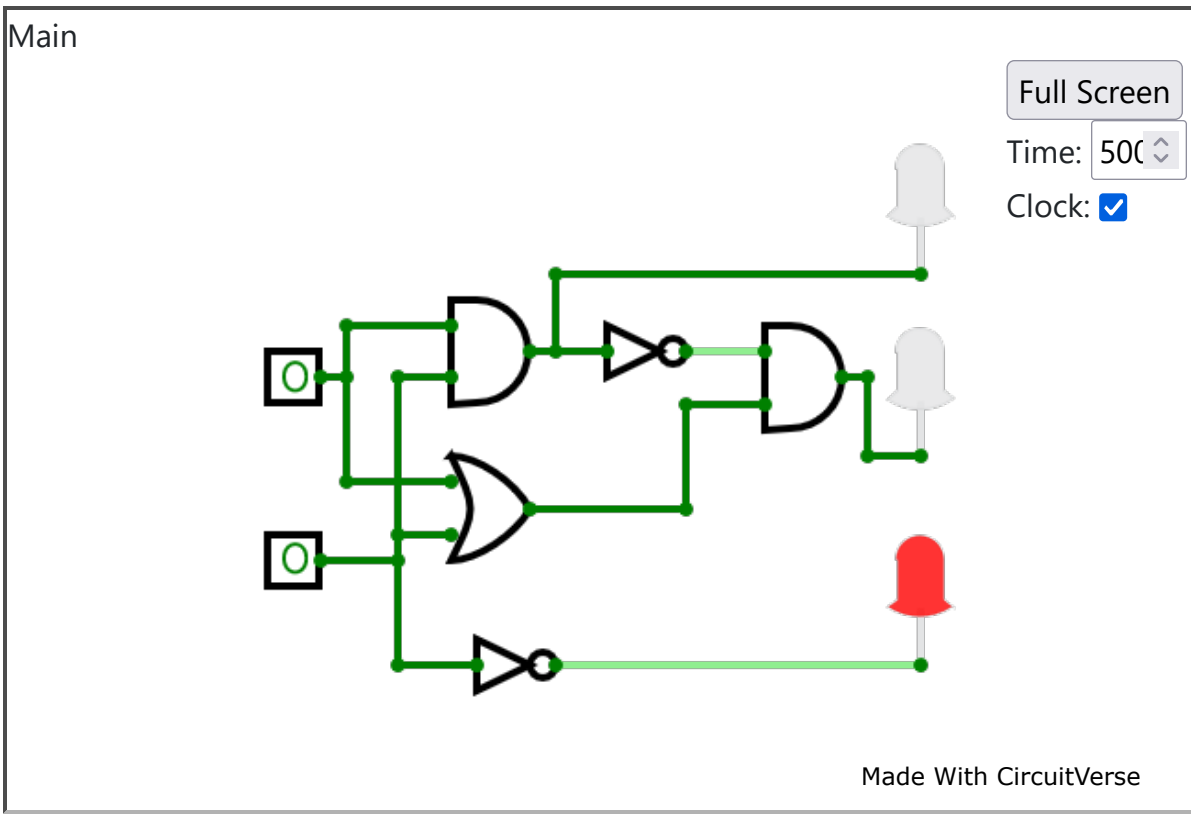
REVIEW from Lecture/Lab 5

Logical gates can be used to do arithmetic on binary numbers. For example, we can write a logical circuit whose output is one more than the inputted number. Our inputs are `in1` and `in2` and the outputs are stored in `out1`, `out2`, and `out3`.



Here is a table of the inputs and outputs:

Inputs			Outputs			
Decimal Number	**in1**	**in2**	**Decimal Number**	**out1**	**out2**	**out3**
0	0	0	1	0	0	1
1	0	1	2	0	1	0
2	1	0	3	0	1	1
3	1	1	4	1	0	0



Toggle the inputs, by clicking on the input boxes for `in1` and `in2`, and observe the output. Then translate the circuit into a logical expression.

Submit a **text file (.txt)** with each of the outputs on a separate line:

```
#Name:  YourNameHere
#Date:  November 2017
#Logical expressions for a 3-bit incrementer

out1 = ...
out2 = ...
out3 = ...
```

Where "..." is replaced by your logical expression (see Lab 5).

Note: here's a quick review of binary numbers.

27. **Due Date: 22 October** **Reading: 10-mins to Pandas, DataCamp Pandas & Lab 6**

Modify the program from Lab 6 that displays the NYC Covid-19 Cases. Your program should ask the user for the name of a borough and the name of the output file.
The program should compute the minimum, maximum, average, median and standard deviation of the covid-19 cases of the borough entered by the user and then display the fraction of the covid-19 daily cases in that borough over time and saves it using the file name entered by the user.

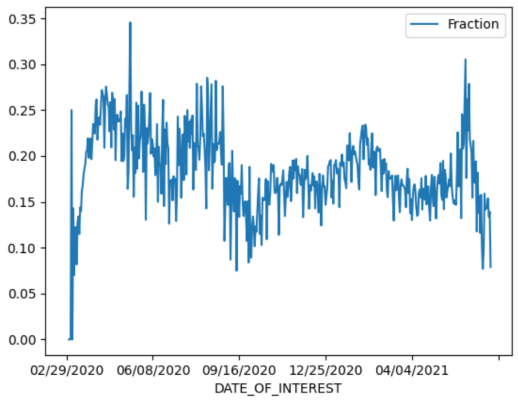
IMPORTANT: The grading script expects the new column to be called 'Fraction'

A sample run of the program:

```
Enter borough name: Bronx
```

```
Enter output name: bronxCovidFraction.png
Min: 0
Max: 1514
Mean: 309.1382113821138
Median: 136.5
Standard Deviation: 336.84985082264524
```

The file `bronxCovidFraction.png`:



Note: before submitting your program for grading, remove the commands that show the image (i.e. the ones that pop up the graphics window with the image). The program is graded on a server on the cloud and does not have a graphics window, so, the `plt.show()` and `plt.imshow()` commands will give an error. Instead, the files your program produces are compared pixel-by-pixel to the answer to check for correctness.

28. **Due Date: 25 October** **Reading: Github Guide & Lab 6**

In Lab 6, you created a github account. Submit a text file with the name of your account. The grading script is expecting a file with the format:

```
#Name:  Your name
#Date:  September 2020
#Account name for my github account

AccountNameGoesHere
```

Note: it takes a few minutes for a newly created github account to be visible. If you submit to gradescope and get a message that the account doesn't exist, wait a few minutes and try again.

29. **Due Date: 26 October** **Reading: Ubuntu Terminal Reference Sheet & Lab 6**

Write an Unix shell script that prints: `Hello CSCI127 Students!`
Then prints on a new line: `$USER is learning shells cripts!`
where `$USER` is a built-in variable that stores the name of the user.

A sample run of the program with user `tizianaligorio`:

```
Hello CSCI127 Students!
tizianaligorio is learning shell scripts!
```

Submit a single text file containing your shell commands. See Lab 6 for details.

Note: The output will not display a username on Gradescope since there is no login on the cloud image where the grading script runs.

30. **Due Date: 27 October** **Reading: 10-mins to Pandas, DataCamp Pandas & Lab 6**

Download the Ramen dataset, look at the data and, in particular, the column names.

Then, write a program that:

- Asks the user for the name of the input file.
- Prints the average rating, out of 5 stars, per serving style.
- Print the lowest rating for a ramen spot in Singapore, and the name of the place.

```
Enter file name:ramenRatings.csv

The average stars per serving style:
Style
Bar     5.000000
Bowl    3.937165
Box     4.291667
Can     3.500000
Cup     3.619075
Pack    3.863330
Tray    3.682203
Name: Stars, dtype: float32

KOKA has the lowest rating in Singapore with 2.5 stars.
```

Hint: You should use `groupby()` and `get_group()` as described in Lab 6 as well as `idxmin()` to find the row containing the restaurant with the lowest star rating in Singapore to get the name of the place (here is a useful resource from GeeksForGeeks).
IMPORTANT: After reading in the csv file, add the following line:

```
df["Stars"] = pd.to_numeric(df["Stars"], downcast="float")
```

Lecture / Lab 7

31. **Due Date: 28 October** **Reading: 10-mins to Pandas, DataCamp Pandas & Lab 7**

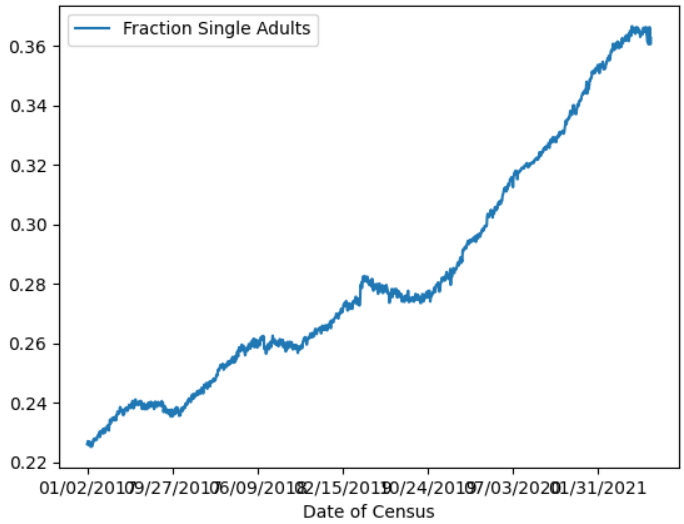
Modify the program from Lab 7 that displays shelter population over time to:

- ask the user to specify the input file,
- ask the user to specify the output file,
- make a plot of the fraction of total single adults in shelter over the total individuals in shelter
- store the plot in the output file the user specified.

A sample run of the program:

```
Enter name of input file:  DHS_2015_2016.csv
Enter name of output file:  dhsPlot.png
```

which produces an output:



Note: The grading script is expecting that the name of your new column is "Fraction Single Adults".

32. **Due Date: 29 October**

Reading: 10-mins to Pandas, DataCamp Pandas, Lab 6 & Lab 7

Let's explore another dataset!

Consider the UFO dataset NUFORC data by Sigmond Axel.
You can obtain the csv file directly here.
First download and inspect the data by looking at the **column names** and type of data in each column.

Now modify the programs from Lab 6 that aggregates rainfall data and the program from Lab 7 that plots shelter population to write a new program that:

- Asks the user for the name of the input file.
- Asks the user for the name of the output file.
- Plots a bar plot of the 10 states in the dataframe with the longest average UFO sightings:
  - Use `groupby()` to aggregate the data by `'state'`, which in this dataset represents the state in which the UFO was seen, and average the `'duration (seconds)'` column.
  - Now sort the values with .sort_values(ascending=False)[:10] to get the top 10 states with the longest average UFO sighting (note, you'll notice that the averages are quite large, that is because this data contains some inconsistencies leading to a few records which report yearlong UFO sightings, etc.)
  - Make a bar plot of the data grouped this way by using `.plot.bar()` on the grouped data

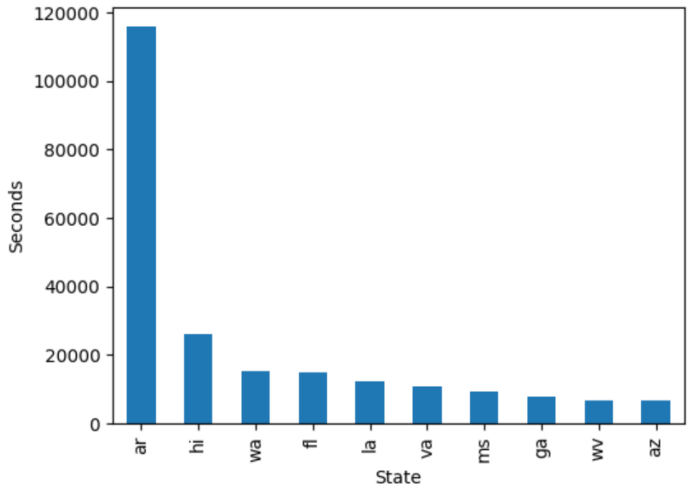
e.g. if you have
`grouped_data = df.groupby( ...`
then use `grouped_data.plot.bar()` to generate the bar plot

- Label the x-axis using `plt.xlabel('State')`
- Label the y-axis using `plt.ylabel('Seconds')`
- Store the plot in the output file the user specified.

A sample run of the program:

```
Enter name of input file:  UFO_sightings_US.csv
Enter name of output file:  UFO_sightings_barplot.png
```

which produces an output:



33. **Due Date: 1 November** **Reading: Think CS Section 6.7 & Lab 7**

Write a program using a function `main()` that asks the user for a number `n` and prints "Hello Functions!" `n` times. **The program should be executable as stand-alone program as well as included as a module in another program.** See Lab 7.

A sample run of the program:

```
Enter a number: 3
Hello, Functions!
Hello, Functions!
Hello, Functions!
```

34. **Due Date: 2 November** **Reading: 10-mins to Pandas, DataCamp Pandas & Lab 7**

In Lab 7, we worked through a program that displayed the homeless shelter occupancy over time. The same approach can be used for displaying any dataset where the date and time are stored.

For this program, use the csv file superbowl.csv. Export the file as CSV and save.

You can look at the program from Lab 7 that displays shelter population over time to create a program that:

- Asks the user to specify the input file,
- Asks the user to specify the output file,
- Converts the date column (which is stored as 'YYYYMMDD') to a datetime format recognized by pandas, for example if your dataframe is `df`, overwrite the 'Date' column to be:

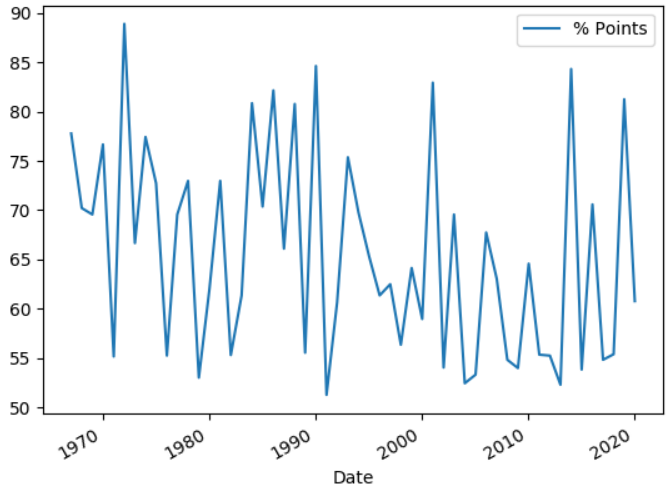
```
df["Date"] = pd.to_datetime(df["Date"].apply(str))
```

- Makes a plot of the percent of winning points out of total points over time from the data in the input file, and
- Stores the plot in the output file specified by the user.

A sample run of the program:

```
Enter name of input file:  Superbowl_Finals.csv
Enter name of output file:  results.png
```

which produces an output:



IMPORTANT: The grading script expects the name of your new column to be "% Points" and the percentage to be between 0 and 100 (Hints: how do you calculate the total points? What do you multiply by to convert a fraction into a percentage?).

35. **Due Date: 3 November** **Reading: Think CS: Chapter 7.4, Lab 4 & Lab 6**

Review from Lecture/Lab 4

You are a student attempting to take an Introduction to Computer Science course. Using what you have learned so far, you must finish this incomplete game titled `Maze Runner`. The objective of the game is, given the maze, the user must provide a string of commands that will navigate through the maze and reach the end. If the end is not reached, the game

will restart from the beginning. The program, `mazeRunner.py` (available at: https://github.com/HunterCSci127/CSci127) takes a string as input and uses that string to control where the user goes throughout the maze (inspired by code.org's graph paper programming).

Your task:
Implement the function `playGame()` to allow the user to use the following symbols:

- 'R': go Right of the current location
- 'L': go Left of the current location
- 'U': go Up from the current location
- 'D': go Down from the current location

`playGame()` must first check, utilizing the other functions in the program, if each command (i.e. each character in the input string) is a valid move. Then update the location of the user (through the given variables `'row'` and `'col'`). If the user attempts to go through walls or simply does not reach the end of the maze after all of the commands have been executed, `PlayGame()` returns false. If they reach the end (represented by an `'X'` at the location), it returns True. As per the display of the maze, each tile is represented by a 3x3 tile of special characters.

- '*': represents the wall tile
- ' ': represents a walkable tile
- 'S': represents the start tile
- 'X': represents the end tile
- 'v': represents the current location of the user

A sample run of the program using an example maze (which you can use for testing here: maze1.txt)

```
Hello Everyone! Welcome to Maze Runner!
To succeed in this Computer Science class, you must go through some trials a
So tonight, you must pass through our maze!
The game is not over until you have found your path...
And if you try and fail, you must restart from the beginning and try again!
Good luck!
Enter name of text file containing a maze: maze1.txt

Here is a picture of the maze, provided by some random person down the stree
*****************************
*****************************
*****************************
vvv   ***         ***      ***
vvv   ***         ***      ***
vvv   ***         ***      ***
***   ***   ***   ******   ***
***   ***   ***   ******   ***
***   ***   ***   ******   ***
***         ***   ***      ***
***         ***   ***      ***
***         ***   ***      ***
*********   *********   ******
*********   *********   ******
```

```
*********   *********   ******
******          ***          ***
******          ***          ***
******          ***          ***
************          ***    ***
************          ***    ***
************          ***    ***
***             ***    *********
***             ***    *********
***             ***    *********
***    ******   ***          ***
***    ******   ***          ***
***    ******   ***          ***
***XXX************************
***XXX************************
***XXX************************
Enter a string of commands: RDDRRR
moved Right, current location is:  1 1
moved Down, current location is:  2 1
moved Down, current location is:  3 1
moved Right, current location is:  3 2
moved Right, current location is:  3 3
******************************
******************************
******************************
SSS    ***          ***       ***
SSS    ***          ***       ***
SSS    ***          ***       ***
***    ***    ***    ******    ***
***    ***    ***    ******    ***
***    ***    ***    ******    ***
***       vvv***    ***       ***
***       vvv***    ***       ***
***       vvv***    ***       ***
*********   *********   ******
*********   *********   ******
*********   *********   ******
******          ***          ***
******          ***          ***
******          ***          ***
************          ***    ***
************          ***    ***
************          ***    ***
***             ***    *********
***             ***    *********
***             ***    *********
***    ******   ***          ***
***    ******   ***          ***
***    ******   ***          ***
***XXX************************
***XXX************************
***XXX************************
Cannot move to the Right! Try again :(

Here is a picture of the maze, provided by some random person down the stree
******************************
******************************
******************************
vvv    ***          ***       ***
vvv    ***          ***       ***
vvv    ***          ***       ***
***    ***    ***    ******    ***
***    ***    ***    ******    ***
***    ***    ***    ******    ***
***           ***    ***       ***
```

```
***          ***    ***       ***
***          ***    ***       ***
*********    *********    ******
*********    *********    ******
*********    *********    ******
******           ***          ***
******           ***          ***
******           ***          ***
************           ***    ***
************           ***    ***
************           ***    ***
***              ***    *********
***              ***    *********
***              ***    *********
***    ******    ***          ***
***    ******    ***          ***
***    ******    ***          ***
***XXX************************
***XXX************************
***XXX************************
Enter a string of commands: RDDRRDDRDDLLLDD
moved Right, current location is:  1 1
moved Down, current location is:  2 1
moved Down, current location is:  3 1
moved Right, current location is:  3 2
moved Right, current location is:  3 3
moved Down, current location is:  4 3
moved Down, current location is:  5 3
moved Right, current location is:  5 4
moved Down, current location is:  6 4
moved Down, current location is:  7 4
moved Left, current location is:  7 3
moved Left, current location is:  7 2
moved Left, current location is:  7 1
moved Down, current location is:  8 1
moved Down, current location is:  9 1
Congrats on escaping the maze! Please do join us again :)
```

Hint: See Lab 4

Lecture / Lab 8

36. **Due Date: 4 November** **Reading: Think CS: Chapter 6 & Lab 8**

    Write a function, `worldRecord()`, that takes two parameters: gender `(string)` and the event type `(int)`. The function should return a `float` for the Olympic world record for the event.

    - Men's Standard, event type 100 meters: record is 9.63 seconds.
    - Men's Standard, event type 200 meters: record is 19.30 seconds.
    - Men's Standard, event type 400 meters: record is 43.03 seconds.
    - Women's Standard, event type 100 meters: record is 10.62 seconds.
    - Women's Standard, event type 200 meters: record is 21.34 seconds.
    - Women's Standard, event type 400 meters: record is 48.25 seconds.

- Return -1 for any other value

A template program, `olympicRecords.py`, is available on the CSci 127 repo on github. The grading script does not run the whole program, but instead tests your function separately ('unit tests') to determine correctness. As such, the name of the function must match exactly (else, the scripts cannot find it).

A sample run:

```
Enter the gender: women
Enter the event distance: 400
The world record for women's 400 meters is 48.25 seconds.
```

And another:

```
Enter the gender: men
Enter the event distance: 100
The world record for men's 100 meters is 9.63 seconds.
```

Hint: *See Lab 8.*

37. **Due Date: 5 November**

Reading: 10-mins to Pandas, DataCamp Pandas & Lab 7 & Lab 8

Write a program that
asks the user for a CSV of Video Games Dataset

- A sample dataset `vgsales.csv` can be found on our github repo.

Your program should then print out:

- The total number of games
- The count of video games of each game genre.
- The top 3 game publishers

A sample run:

```
Enter file name: vgsales.csv
There are 16598 total games

The number of games in each genre is
Action          3316
Sports          2346
Misc            1739
Role-Playing    1488
Shooter         1310
Adventure       1286
Racing          1249
Platform         886
Simulation       867
Fighting         848
Strategy         681
Puzzle           582
Name: Genre, dtype: int64

The top 3 game publishers are
Electronic Arts       1351
Activision             975
Namco Bandai Games     932
```

```
Name: Publisher, dtype: int64
```

This assignment uses data freely available on kaggle.com, and can be found at:

https://www.kaggle.com/gregorut/videogamesales.

Hint: *See Lab 8 for counting values in structured data.*

38. **Due Date: 8 November** **Reading: Think CS Chapter 6 & Lab 8**

Write three functions, `setUp()`, `fractalRight()`, and `fractalLeft()`.

Provided in the class Github is a template program, `'turtleTree.py'`. Starting from this template and using the pseudocode below (also included as comments in the template), implement your program as follows:

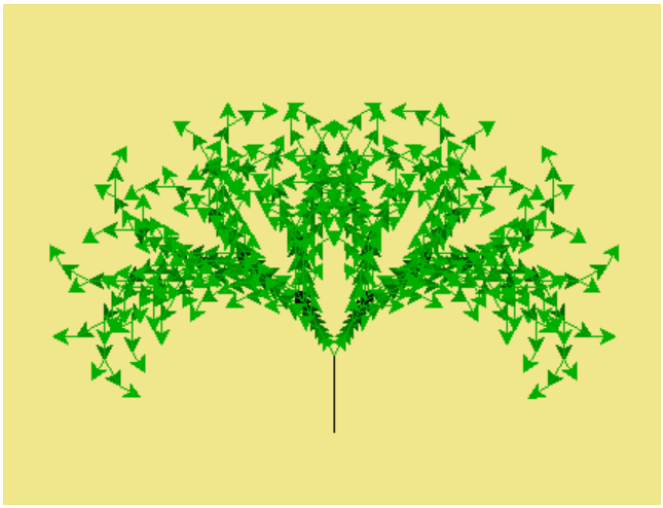
```
setUp():
  Takes no parameters, and simply sets up the turtle, returning to c
  Here you should:
      1. Instiantiate the turtle,
      2. optionally hide the turtle (use .hideturtle())
      3. set the heading to 90 (which will face the turtle upwards)
      4. set the speed (t.speed(speed=0), you will want this)
      5. return the turtle to the calling function.

fractalRight():
  Takes three parameters: a turtle t, distance 'distance', and i, th
  color value. This function does the following: if the distance is
  it will:
      1. Set the value of i to be i % 255,
      2. set the color of the turtle to be the value of i in the gre
      3. turn right 30 degrees
      4. move forward 'distance' steps
      5. stamp the turtle
      6. set the value of newDistance to distance-5
      7. call fractalRight with the parameters t, newDistance, i+25
      8. move backward by 'distance' steps, NOT newDistance,
      9. turn left 30 degrees
      10. call fractalLeft with parameters t, newDistance, i+25

fractalLeft():
  This is pretty much the same as fractalRight, just to the left:
  Takes three parameters: a turtle t, distance 'distance', and i, th
  color value. This function does the following: if the distance is
  it will:
      1. Set the value of i to be i % 255,
      2. set the color of the turtle to be the value of i in the gre
      3. turn left 30 degrees
      4. move forward 'distance' steps
      5. stamp the turtle
      6. set the value of newDistance to distance-5
      7. call fractalLeft with the parameters t, newDistance, i+25
      8. move backward by 'distance' steps, NOT newDistance,
      9. turn right 30 degrees
      10. call fractalRight with parameters t, newDistance, i+25
```

There is a subtle difference in the pseudocode of the two functions above, but it is very important and makes the difference between the two results. Observe the execution

(hopefully by increasing the speed to save some time) and think about it!!!

Again, the template program, `turtleTree.py`, is available on the CSci 127 repo on github. The grading script does not run the whole program, but instead it tests your functions separately ('unit tests') to determine correctness. As such, the function names must match exactly (else, the scripts cannot find it). Make sure to use the function names from the github program (it is expecting `setUp()`, `fractalLeft()` and `fractalRight()`).

A sample run:



39. **Due Date: 9 November** **Reading: Think CS: Chapter 6 and Section 8.10 & Lab 8**

Using the template program, `averageImage.py`, available on the CSci 127 repo on GitHub, fill in the missing functions:

- `average(region)`: Takes a region of an image and returns the average red, green, and blue values across the region.
- `setRegion(region,r,g,b)`: Takes a region of an image and red, green, and blue values, r, g, b. Sets the region so that all points have red values of r, green values of g, and blue values of b.

The functions are part of a program that averages smaller and smaller regions of an image until the underlying scene is visible (inspired by the elegant koalas to the max).

For example, if you inputted our favorite image, you would see (left to right):

and finally:

The grading script does not run the whole program, but instead runs each of your functions separately ('unit tests') to determine correctness. As such, the names of the functions must match exactly the ones listed above (else, the scripts cannot find them).

IMPORTANT Do not remove anything below **and including** the comment that says:

```
########################################################################
### DO NOT CHANGE ANYTHING BELOW AND INCLUDING THIS LINE             ###
########################################################################
```

Hint: See notes from Lecture 8.

40. **Due Date: 10 November** **Reading: Think CS: Chapter 6 and Section 7.4 & Lab 8**

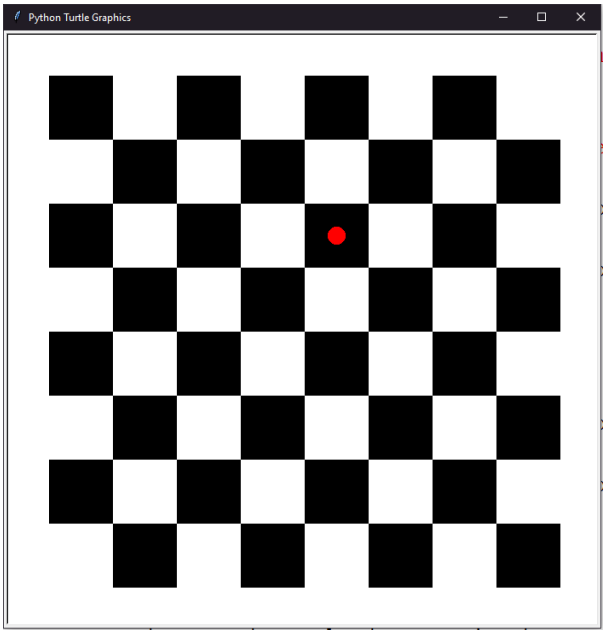
Using the template program, `checkers.py`, available on the CSci 127 repo on GitHub, fill in the missing lines of code to create a program that constructs a checkerboard and a checker piece that moves. Detailed instructions are provided in the template program.

- Import libraries necessary for the program
- `drawBoard(size)`: Creates a checkerboard of `size` pixels as a `numpy` array and saves as `board.png` using `matplotlib`.
- `setupScreen(size)`: Sets up a `turtle` screen of `size + 100` by `size + 100` with the background image as the board. **Hint:** See Lab 9 for an example of screen setup.
- `createPiece(pieceColor)`: Creates and returns a turtle to act as a checker piece. The turtle shape should be a circle and the color should be the parameter value. The turtle pen should be picked up.
- `movePiece(size, piece, row, col)`: Moves the checker piece to a specific row and column on the board.
- `main()`: Asks the user for the size of the checkerboard. Then, creates the checkerboard, sets up the screen, and a creates a single red checker piece. Lastly, asks the user for a row and a column and moves the piece to that location.

A sample run:

```
Enter size of checkerboard: 400
Enter row to move piece to: 3
Enter col to move piece to: 5
```

which would produce:



IMPORTANT: No lines of code from the template program should be removed or changed. Only add in the missing code.

Lecture / Lab 9

41. **Due Date: 11 November** **Reading: Think CS: Chapters 4 and 6 & Lab 8**

REVIEW from Lecture/Lab 4 & 7

Write a program that implements a basic calculator. The template program, `python_calculator.py` is available on the CSci 127 repo on github.

The `main` function takes as input:

- `select`: a number that indicates the operation,
- `number_1`: the first operand and
- `number_2`: the second operand.

You must implement the following functions:

- `add(num1, num2)` that takes two numbers and returns the result of their addition.
- `subtract(num1, num2)` that takes two numbers and returns the result of their subtraction.

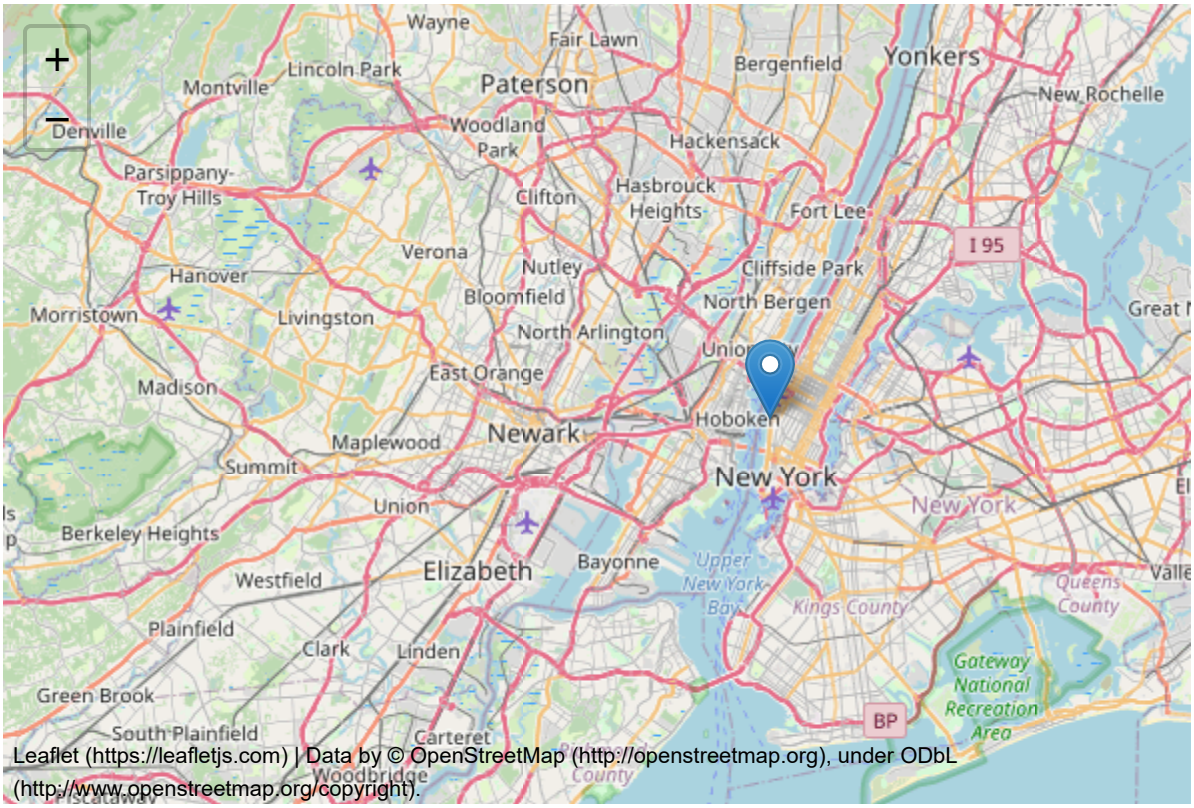
- `multiply(num1, num2)` that takes two numbers and returns the result of their multiplication.
- `divide(num1, num2)` that takes two numbers and returns the result of their division.
- `sqrt(num1)` that takes a number and returns the square root of that number.
- `squared(num1)` that takes a number and returns the result of that number squared.
- `exponent(num1, num2)` that takes two numbers and returns the result of the first number to the exponent indicated by the second number.

The grading script does not run the whole program, instead it runs each of your functions separately ('unit tests') to determine correctness. As such, the names of the functions must match exactly the ones listed above (else, the scripts cannot find them).

A sample run of your program:

```
Please select an operation -
1. Add
2. Subtract
3. Multiply
4. Divide
5. Square root
6. Squared
7. Exponent

Select operations from: 1, 2, 3, 4, 5, 6, 7 : 4
Enter first number: 5
Enter second number: 3
5 / 3 = 1.6666666666666667
```

42. **Due Date: 12 November** **Reading: Folium Tutorial & Lab 9**



Write a program that uses folium to make a map centered in New York City (40.75, -74.125) and add a marker for Little Island (in Chelsea Piers) at coordinates 40.7420577, -74.0101494. The HTML file your program creates should be called: `nycMap.html`.

Hint: See Lab 9.

43. **Due Date: 15 November** **Reading: Folium Tutorial & Lab 9**

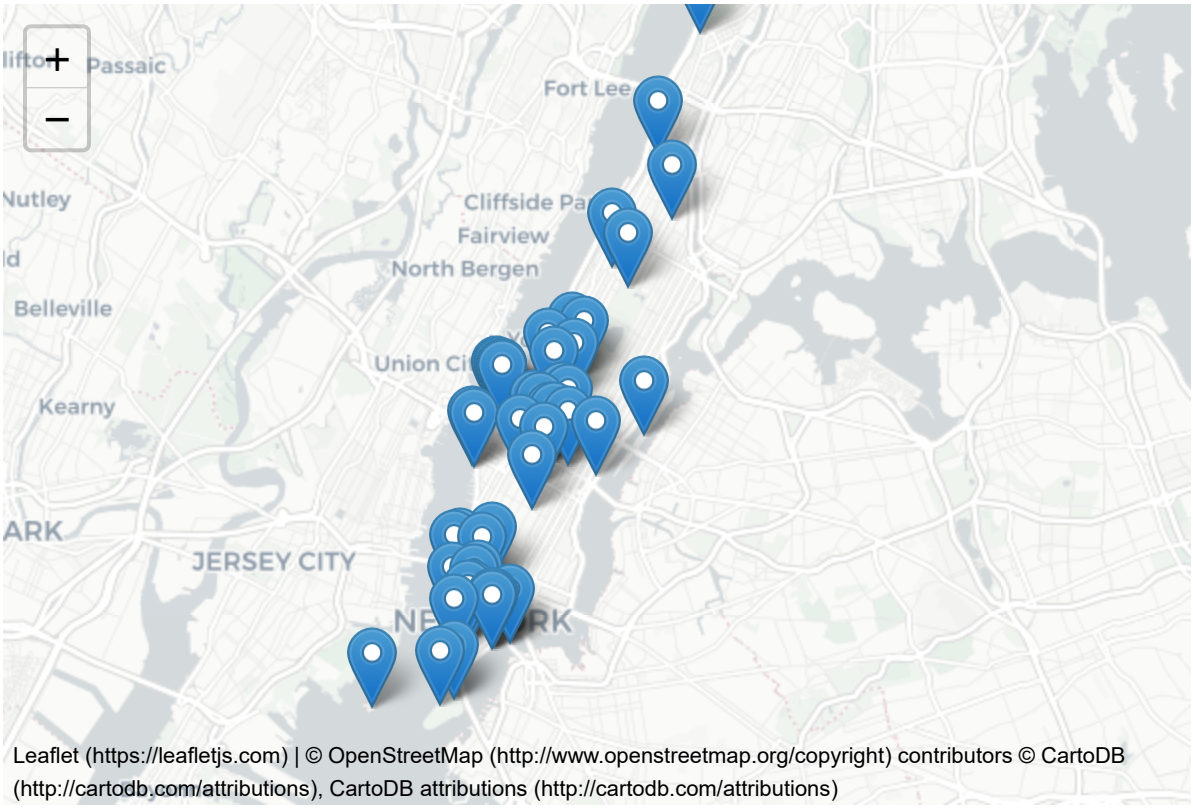
Using folium (see Lab 9), write a program that asks the user for the name of a CSV file, name of the output file, and creates a map with markers for all the NY attractions from the input file.

The map should be centered at Hunter College (40.768731, -73.964915).

A sample run:

```
Enter CSV file name:  attractions.csv
Enter output file:  thMap.html
```

which would produce the html file:



(Use the name of the attractions ("NAME") to label each marker and changed the underlying map with the option: `tiles="Cartodb Positron"` when creating the map.)

A sample file, attractions.csv can be downloaded and used to test your program.

Note: In your code, make sure that each row in the given dataset has a value for LATITUDE and LONGITUDE before attempting to create a marker for that location on the map.

Hint: For this data set, the names of the columns are "LATITUDE" and "LONGITUDE" (unlike the previous map problem, where the data was stored with "Latitude" and "Longitude").

44. **Due Date: 16 November** **Reading: Think CS: Chapter 3 & Lab 9**

The program, errorsUTAs.py, has lots of errors. Fix the errors and submit the modified program. Pay attention to what your program outputs and what it should output!

The correct program should output:

```
UTA List:
Aida
Arterio
Destiny
Ghazanfar
Ifte
Ilya
Leonardo
Mandy
Nancy
Sadab
Stephanie
Tyler

Lola is not a UTA for CSci 127 Fall 2021.
Mandy is a UTA for CSci 127 Fall 2021.
Leonardo is a UTA for CSci 127 Fall 2021.
Yash is not a UTA for CSci 127 Fall 2021.
There are 12 UTAs who love helping students in CSci 127 Fall 2021.
```

Hint: See Lab 9.

45. **Due Date: 17 November** **Reading: Folium Tutorial & Lab 9**

Using folium, write a program that asks the user for the name of a CSV file, name of the output file, the name of a borough and creates a map with markers for all the NYC Wi-Fi Hotspot locations in that borough from the input file.
Hint: You can use `groupby('City')` and `get_group()` to consider only the rows for the input borough. **Note that** the value for Manhattan in the 'City' column is actually "New York".
Use the NYC_Wi-Fi_Hotspot_locations.csv file.

The data is made publicly available by New York City Wifi Hotspot Locations. You can also obtain the data here: NYC-Wi-Fi-Hotspot-Locations.csv

A sample run:

```
Please enter the name of the input file: NYC-Wi-Fi-Hotspot-Locations.csv
Please enter the name of the output file: manhattan.html
Please enter the name of the borough: New York
```

which would produce the html file:

Note: You can use the value in the "Name" column to label each marker and changed the underlying map with the option: `tiles="Stamen Watercolor"` when creating the map.

Lecture / Lab 10

46. **Due Date: 18 November** **Reading: Chapter 8, Lab 7 & Lab 10**

Write a program that asks the user for an integer input. Using input validation (while loop), check to see if the input is a perfect square. If it is not, continue asking the user to re-enter the input until it is a perfect square.

A perfect square is the product of an integer squared. Thus, a perfect square will have a whole integer root.
e.g. 4 is a perfect square because 2^2 = 4.
e.g. 16 is a perfect square because 4^2 = 16.

Your program will consist of three functions, but you only need to implement `main()`. The other two are provided in the template program `perfectSquare.py`, available on our github repo.

The three functions are:

- `main()` is responsible for obtaining **and validating** the input and printing the output. In order to validate the input, it will **repeatedly** ask for a number until a valid number is entered (a perfect square).
- `perfectSquareChecker()` which, as the name suggests, will output whether or not the input integer is a perfect square.

- `is_square()` the function used by `perfectSquareChecker` uses to check if a number is a square. It will return true if the input integer is a perfect square, and false otherwise.

A template `perfectSquare.py` is available on our github repo. You must **fill in `main()`** to implement its desired behavior.

A sample run of your program:

```
Enter a square integer: 12
That is not a perfect square. Try again: -5
That is not a perfect square. Try again: 123
That is not a perfect square. Try again: 31
That is not a perfect square. Try again: 21
That is not a perfect square. Try again: 9
This number is a perfect square. It is the product of 3.0 squared.
```

Note: The grading scripts are expecting that your function is called `perfectSquareChecker()`. You need to use that name, since in addition to running the entire program, the scripts are also "unit testing" the function-- that is, calling that function, in isolation, with different inputs to verify that it performs correctly.

Hint: *See Lab 10 for an example of validating the input.*

47. **Due Date: 19 November** **Reading: Chapter 8 & Lab 10**

Create a rock papers scissors game. Your program must print the **output** as specified below:

- Print `ComputerMove` for each round.
- Print the winner of each round on a new line.
- At the end of the game, print the winner of the last round.

Rules:

- Player moves: "1" = rock, "2" = paper, "3" = scissors
- Computer moves: "1" = rock, "2" = paper, "3" = scissors
- Must have **winner variable** to track the winner at the end of the game (the winner of the last turn).
- If player wins (winner = "human")
- If computer wins (winner = "computer")
- If round is a tie (winner = "draw")
- If the `playerMoveove` is invalid, **> 3** (winner = "invalid")
- If If the `playerMoveove` is **< 1, terminate the game.**

Game logic: **Loop until playerMove is less than zero**

- If playerMove is **rock (1)** and computer move is **rock (1)**, its a draw.
- If playerMove is **rock (1)** and computer move is **paper (2)**, computer wins.
- If playerMove is **rock (1)** and computer move is **scissors (3)**, human wins.
- If playerMove is **paper (2)** and computer move is **rock (1)**, human wins.
- If playerMove is **paper (2)** and computer move is **paper (2)**, its a draw.
- If playerMove is **paper (2)** and computer move is **scissors (3)**, computer wins.
- If playerMove is **scissors (3)** and computer move is **rock (1)**, computer wins.
- If playerMove is **scissors (3)** and computer move is **paper (2)**, human wins.
- If playerMove is **scissors (3)** and computer move is **scissors (3)**, its a draw.

A sample run of your program:

```
enter 1 for rock, 2 for paper, or 3 for scissors: 3
computerMove: 1
round finished and winner is: computer
enter 1 for rock, 2 for paper, or 3 for scissors 1
computerMove: 3
round finished and winner is: human
enter 1 for rock, 2 for paper, or 3 for scissors 2
computerMove: 2
round finished and winner is: draw
enter 1 for rock, 2 for paper, or 3 for scissors 23
computerMove: 1
round finished and winner is: invalid
enter 1 for rock, 2 for paper, or 3 for scissors 3
computerMove: 2
round finished and winner is: human
enter 1 for rock, 2 for paper, or 3 for scissors 0
game finished and winner is: human
```

48. **Due Date: 22 November** **Reading: Ubuntu Terminal Reference Sheet & Lab 10**

Write an Unix shell script that does the following:

- Creates a directory, `csci127`.
- Creates two additional directories inside of `csci127` (as subdirectories of `csci217`): `images` and `files`.
- Creates 3 additional directories inside of `images` (as subdirectories of `images`): `png`, `jpg` and `gif`.
- Creates 2 additional directories inside of `files` (as subdirectories of `files`): `csv` and `txt`.
- Lists everything in the directory `files`

Submit a single text file containing your shell commands. See Lab 10.

49. **Due Date: 23 November** **Reading: Chapter 8, Lab 7 & Lab 10**

Write a program that creates a 10 by 10 image of a color of the users choice. Instead of simply asking the user for a color, the user will be creating the color by **iteratively**

increasing and decreasing the `rgb` values until the user quits by entring 0 for each `rgb` value.
Starting with an image of zeros, the program asks the user for the name of the file the image will be saved to and then **repeatedly** asks how much they would like to increase/decrease each `rbg` value. The `rbg` values are initially 0.
Assume that the user can input any positive or negative number, but the maximum value can be 255 and the lowest value is 0. **If the user enters a 0** for increasing/decreasing each `rgb` value, then **the program ends** and saves the image with the name provided by the user.

A sample run of your program:

```
----------------------------------------
Welcome to the Color Maker!
----------------------------------------
Please enter for each rbg color the value in which to increase/decrease them
If all 3 values given are 0, the program will end and save the resulting ima

Enter outfile name: yourColor.png

How much do you want to change the red value by? 10
How much do you want to change the green value by? 100
How much do you want to change the blue value by? 10
The current rgb values are: [0.03921568627450979,0.392156862745098,0.0392156

How much do you want to change the red value by? 100
How much do you want to change the green value by? -50
How much do you want to change the blue value by? 200
The current rgb values are: [0.4313725490196079,0.196078431372549,0.82352941

How much do you want to change the red value by? 62
How much do you want to change the green value by? 0
How much do you want to change the blue value by? 70
The current rgb values are: [0.6745098039215687,0.196078431372549,1.0]

How much do you want to change the red value by? 0
How much do you want to change the green value by? 0
How much do you want to change the blue value by? -100
The current rgb values are: [0.6745098039215687,0.196078431372549,0.60784313

How much do you want to change the red value by? 0
How much do you want to change the green value by? 0
How much do you want to change the blue value by? 0
The current rgb values are: [0.6745098039215687,0.196078431372549,0.60784313

You're done! Congrats on the color, it looks beautiful!
```

Hint: *See Lab 4 for converting input to numbers*

Lecture / Lab 11

50. **Due Date: 24 November** **Reading: MIPS Wikibooks & Lab 11**

    Write a **simplified machine language program** that prints: `MIPS is cool!`

    See Lab 11 for details on submitting the simplified machine language programs.

    Hint: You may find the following table useful:

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

(Image from wikimedia commons)

Hint: The grading scripts are matching the phrase exactly, so, you need to include the spacing and punctuation.

51. **Due Date: 29 November** **Reading: MIPS Wikibooks& Lab 11**

    Write a **simplified machine language program** that has register `$s0` decrements from 100, 75, 50, 25, 0

    Your program must use the `ADDI` instructions and store all numbers in registers for computation.

    See Lab 11 for details on submitting the simplified machine language programs.

52. **Due Date: 30 November**

You've now mastered **Reading:** [Ubuntu Terminal Reference Sheet] **&** [Lab 10] **&** [Lab 11]**.** UNIX Scripts.

You will organize some files in a cloned GitHub repository.

Clone the repository https://github.com/HunterCSci127/CSci127 with the git command. (Refer to Lab 8 to refresh on cloning a repository)

Navigate to this repository and make a new directory called `myprojectfiles`

Move `nestingPolygons.py` into `myprojectfiles`

Go back to the repository directory, and make another directory called `pictures`

Move any `.png` file found in the cloned repo into the `pictures` directory you just made

Hint: See Lab 11.

Lecture / Lab 12

53. **Due Date: 1 December** **Reading:** [Cplusplus Tutorial] **&** [Lab 12]

    Write a **C++ program** that prints "Hello, World!" on one line and "Hello, C++!" on a new line.

    The output of your program should be:

    ```
    Hello, World!
    Hello, C++!
    ```

    Hint: See Lab 12 for getting started with C++.

54. **Due Date: Due Date: 2 December** **Reading:** [Cplusplus Tutorial] **&** [Lab 12]

    Write a **C++ program** that will print "01" 5 times in one row, for 5 rows.

    The output of your program should be:

    ```
    0101010101
    0101010101
    0101010101
    0101010101
    0101010101
    ```

    Hint: See Lab 12 for getting started with C++.

    Extra hint: You can accomplish this very quickly with a nested for loop. Write a for loop for each row, and inside of that for loop write another for loop to iterate through each column.

55. **Due Date: 3 December** **Reading: Cplusplus Tutorial & Lab 12**

Write a **C++ program** that converts cryptocurrencies to USD. Your program should prompt the user for an amount in cryptocurrency and then print out the conversion from Bitcoin, Ethereum, and Dogecoin to USD.

The current conversion rates (from Coindesk) as of July 16, 2021:

- 1 BTC = $31,901.19 USD
- 1 ETH = $1,901.54 USD
- 1 DOGE = $0.179733 USD

IMPORTANT: For grading purposes, set the precision to two decimal places. To do so, add `#include <iomanip>` at the top of the file, and `cout << fixed << setprecision(2);` before outputting the USD amounts.

A sample run of your program:

```
Enter amount in cryptocurrency: 1.27
1.27 BTC in USD: $40514.51
1.27 ETH in USD: $2414.95
1.27 DOGE in USD: $0.22
```

See Lab 4 for designing Input-Process-Output programs and Lab 12 for getting started with C++.

56. **Due Date: 7 December** **Reading: Cplusplus Tutorial & Lab 12**

Write a **C++ program** program that asks the user for a number n and a word for what this is a reminder for. The program then counts down from n and finally prints "Time for `[word]`!".

A sample run:

```
Please enter a number: 6
Please type a word: Class
6,
5,
4,
3,
2,
1,
Time for Class!
```

Lecture / Lab 13

57. **Due Date: 7 December** **Reading: Cplusplus Tutorial & Lab 13**

Write a **C++ program** that asks the user for the current temperature and prints

- "It's freezing!" if the given temperature is 32 degrees or lower
- "It's cold!" if the given temperature is less than 68 degrees and greater than 32

degrees

- "It's warm!" if the given temperature is at least 68 degrees and less than 73 degrees
- "It's hot!" if the given temperature is 73 degrees or greater

A sample run:

```
Enter the temperature: 0
It's freezing!
```

Another sample run:

```
Enter the temperature: 49
It's cold!
```

And another run:

```
Enter the temperature: 68
It's warm!
```

58. **Due Date: 8 December** **Reading: Cplusplus Tutorial & Lab 13**

Write a **C++ program** that asks the user for a number n and a character c, and prints a trapezoid using that character (shown below).

A sample run:

```
Please enter a number: 5
Please enter a character: @
@
@@
@@@
@@@@
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
```

Another sample run:

```
Please enter a number: 3
Please enter a character: *
*
**
***
***
***
***
```

59. **Due Date: 9 December** **Reading: Cplusplus Tutorial & Lab 13**

Write a **C++ program** that asks the user for their monthly budget on food and coffee and ask the user how much they spend on coffee in a week and print out how much they have left to spend every week after purchasing coffee.

A sample run:

```
Enter your monthly budget for food and coffee: 400
How much are you spending in a week for coffee: 40.50
Budget left after week 1      359.5
Budget left after week 2      319
Budget left after week 3      278.5
Budget left after week 4      238
```

60. **Due Date: 10 December** **Reading: Cplusplus Tutorial & Lab 13**

Write a **C++ program** that asks the user for a whole number between -31 and 31 and prints out the number in "two's complement" notation, using the following algorithm:

1. Ask the user for a number, n.
2. If the number is negative, print a 1 and let x = 32 + n.
3. If the number is not negative, print a 0 and let x = n.
4. Let b = 16.
5. While b > 0.5:
    1. If x >= b then print 1, otherwise print 0
    2. Let x be the remainder of dividing x by b.
    3. Let b be b/2.
6. Print a new line ('\n').

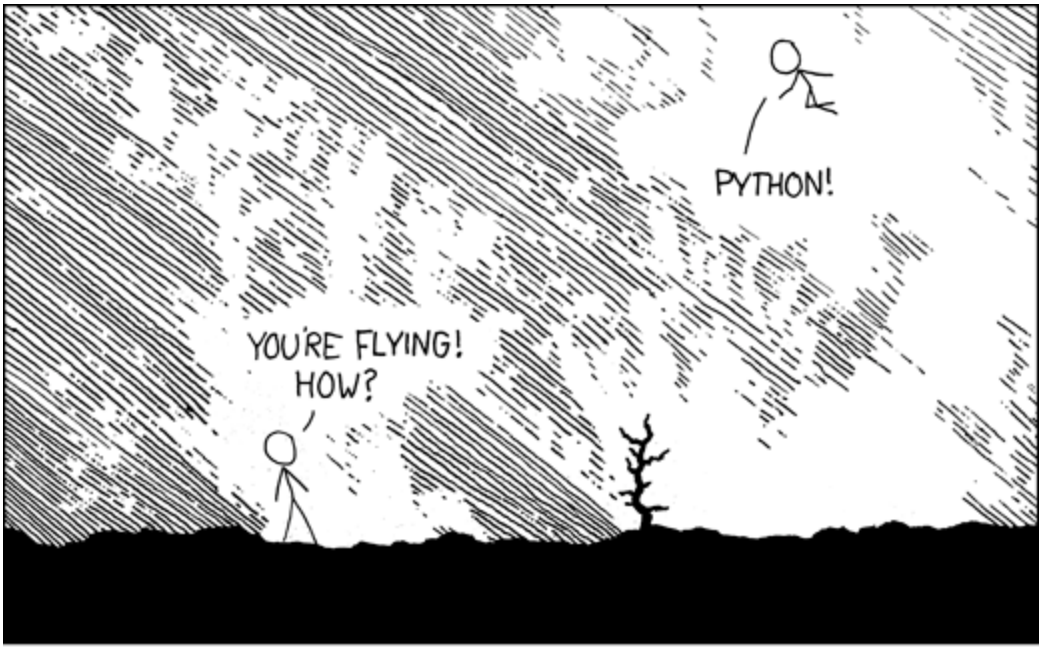
A sample run:

```
Enter a number:  8
001000
```

Another run:

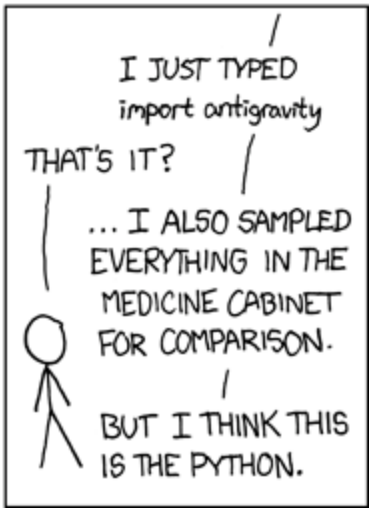
```
Enter a number: -1
111111
```

Here's xkcd on the simplicity of Python:


PYTHON!
YOU'RE FLYING! HOW?


I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"


I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?


I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.