CPSC 217 Assignment 2

Due: Friday October 26, 2018 at 12:00 noon
Weight: 7%
Sample Solution Length: 95 lines, including some comments and the code to implement the A+ portion of the assignment

Individual Work:

All assignments in this course are to be completed individually. Students are advised to read the guidelines for avoiding plagiarism located on the course website. Students are also advised that electronic tools may be used to detect plagiarism.

Late Penalty:

Late assignments will not be accepted.

Submission Instructions:

Your program must be submitted electronically to the Assignment 2 drop box in D2L. It is your responsibility to ensure that your file has been uploaded successfully before the deadline. Save the confirmation email that D2L sends to you when your submission has been received successfully.

Description

In this assignment you will create a program that plots a hurricane's track through the Caribbean and surrounding areas. The user will enter latitude, longitude and wind speed data which your program will display on a map. Dots of different colors and sizes will be used to represent hurricanes of different strengths, and lines will be used to connect the dots to illustrate its path.

Completing this assignment will require the use of loops and if statements. You do not need to write your own Python functions to complete this assignment, though you may if you would like to. If you choose to write your own functions you must use them properly, including thorough documentation.

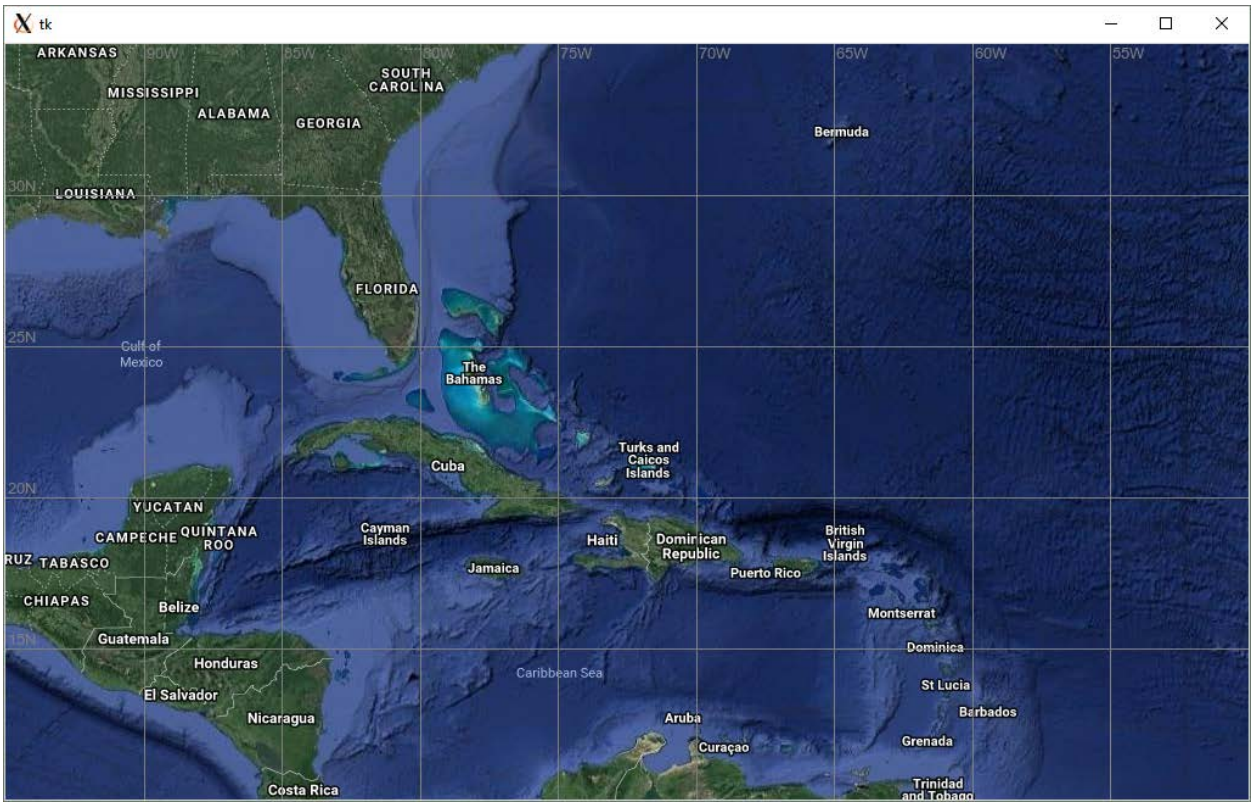
The following sections provide additional requirements and suggestions for implementing your program.

Part 1: Drawing the Map

A map of Caribbean can be found in `map.gif` on the course website. Your program should begin by resizing the graphics window so that it is the same size as the map (1022 pixels wide and 620 pixels high). Then the map should be displayed. This can be accomplished using the loadImage and drawImage functions. The loadImage function takes the name of the file to load (a string) as its only argument and returns an image as its result. This image can be drawn by passing it as the first argument to drawImage. The drawImage function also requires the x and y location where the image should be drawn as its second and third arguments.

Once the map has been displayed your program should go on and add lines of latitude and longitude to it. These lines should both be 5 degrees apart and labelled with their latitude / longitude value (including the directional indicator). The left edge of the map is 95 degrees west while its right edge is at 50 degrees west. The bottom edge of the map is at 10 degrees north while its top edge is at 35 degrees north. For simplicity we will drawn the lines of longitude as vertical lines (rather than curves) and we will use equal spacing for all degrees of latitude rather than performing a complex projection. Use loops to draw these lines. It is **not** acceptable to use a (long) collection of line and text function calls to complete this task.

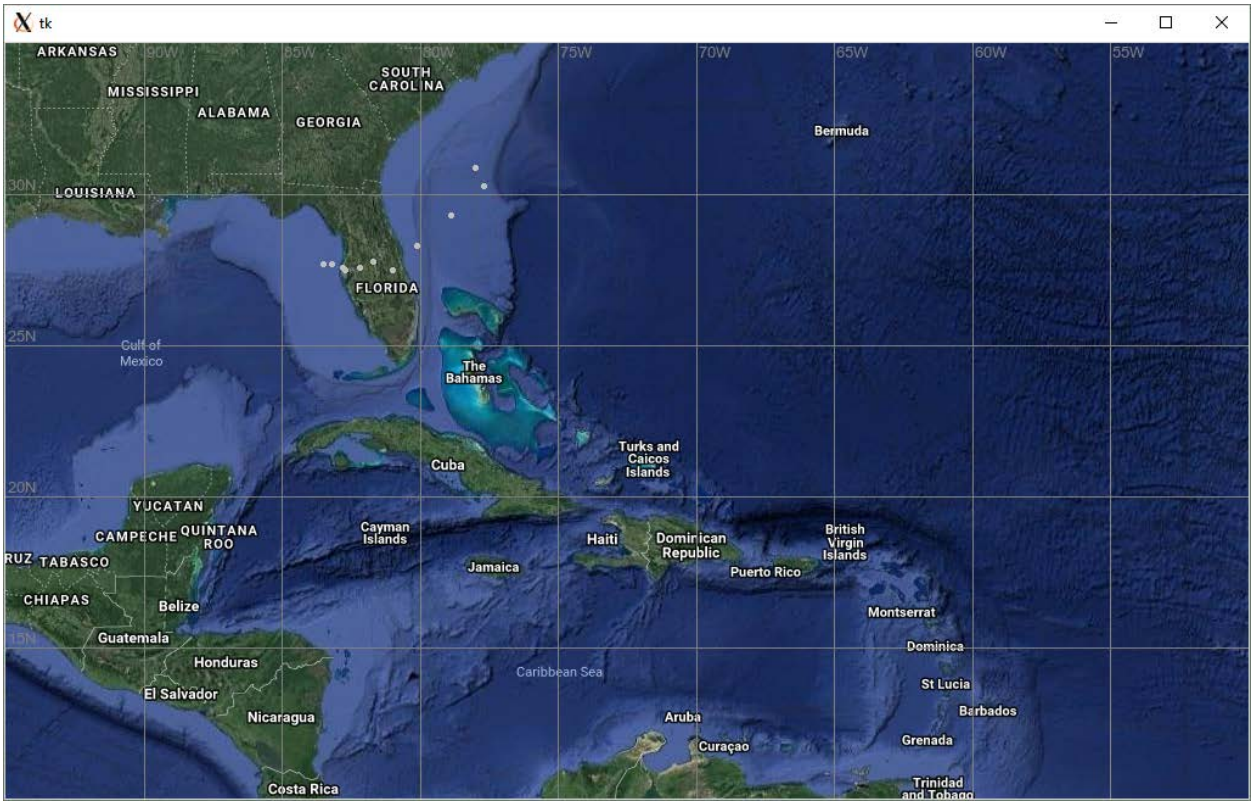
After completing this step your program should display the following image:



Part 2: Plotting the Hurricane's Location

Once the map is drawn your program will read data about the hurricane's location and wind speed from the user. The latitude will be entered first (as a floating-point number of degrees north of the equator), followed by the longitude (as a floating-point number of degrees east of the prime meridian), followed by the wind speed (as a floating-point number of miles per hour). Note that all of the longitude values entered will be negative because the Caribbean is west of the prime meridian.

Latitude, longitude and wind speed values will continue to be entered until the user enters a latitude of 0 degrees. Each time a latitude and longitude value are read your program should plot a small gray circle with a width and height of 5 pixels on the map at the indicated location. For example, plotting the data for Emily (technically a tropical storm, not a hurricane) should draw several dots across Florida, as shown below:



Hint: Every time you run your program you will need to enter several data values. You can automate this by using a feature known as I/O redirection, which will allow your program to read the values from a file instead of from the keyboard. For example, to use the values from the file named Emily.txt instead of reading values from the keyboard, enter the command:

```
python Asn2.py < Emily.txt
```

The details of how I/O redirection works aren't important within the scope of this course, but your ability to use it by following the pattern above will save you a significant amount of typing. Several sample data sets saved in the correct format are available on the course website. Feel free to get as much help as you need with I/O redirection from the TAs or instructor. I/O redirection can be used from the terminal / command prompt on Windows, MacOS and Linux.

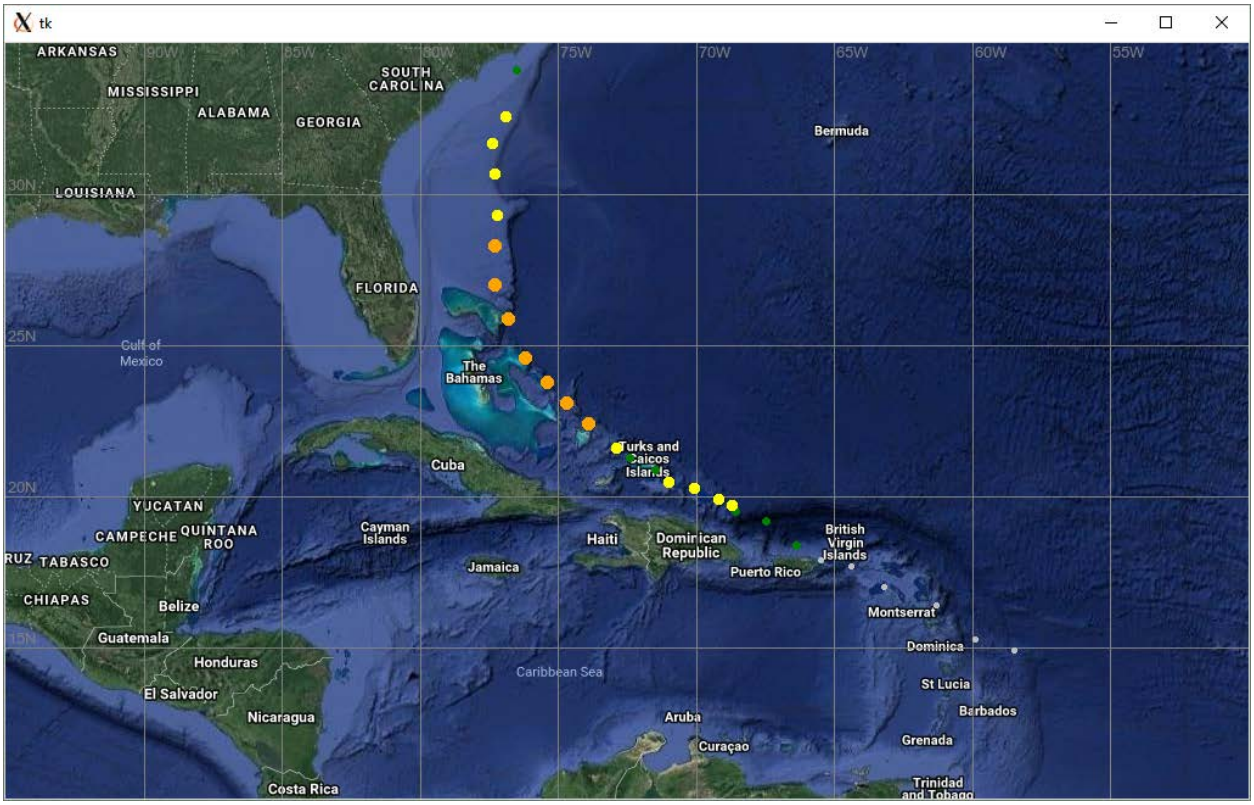
Part 3: Adding Color and Size

Hurricanes are divided into 5 categories based on their wind speed. These are summarized in the following table:

Wind Speed	Category	Color	Dot Diameter (pixels)
>= 157 mph	5	Purple	15
130 mph to less than 157 mph	4	Red	13
111 mph to less than 130 mph	3	Orange	11
96 mph to less than 111 mph	2	Yellow	9
74 mph to less than 96 mph	1	Green	7

The hurricane data plotted on the map will be more meaningful if different colors and sizes are used for different storm categories. Use the data in the preceding table to color the dots according to the storm's wind speed, and use the dot diameters from the table so that more server storms occupy more space on the map. Storms with a wind speed below 74 miles per hour are classified as tropical storms (wind speeds of 39 mph or more) or tropical depressions (wind speeds less than 38 mph). Such storms should continue to be drawn in gray using a circle that is 5 pixels in diameter.

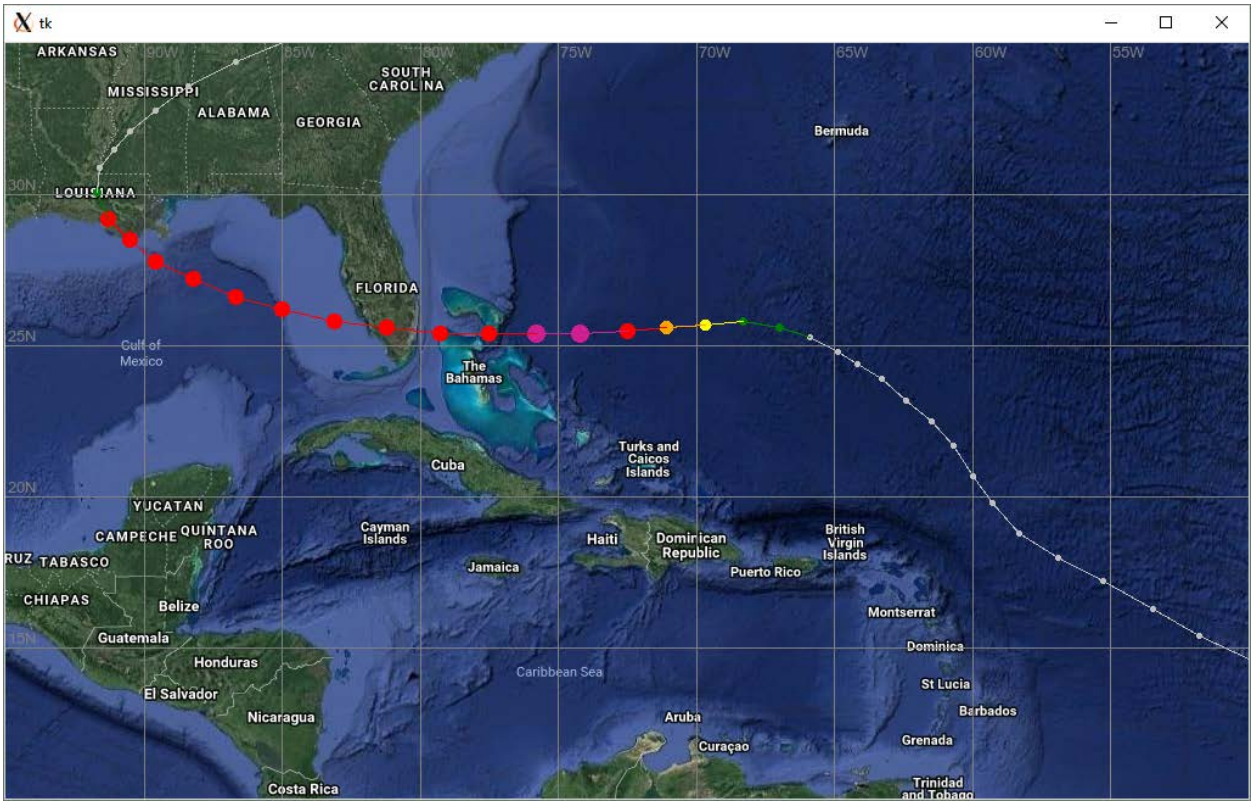
The dots for Hurricane Irene is are shown below:



Part 4: Connecting the Dots

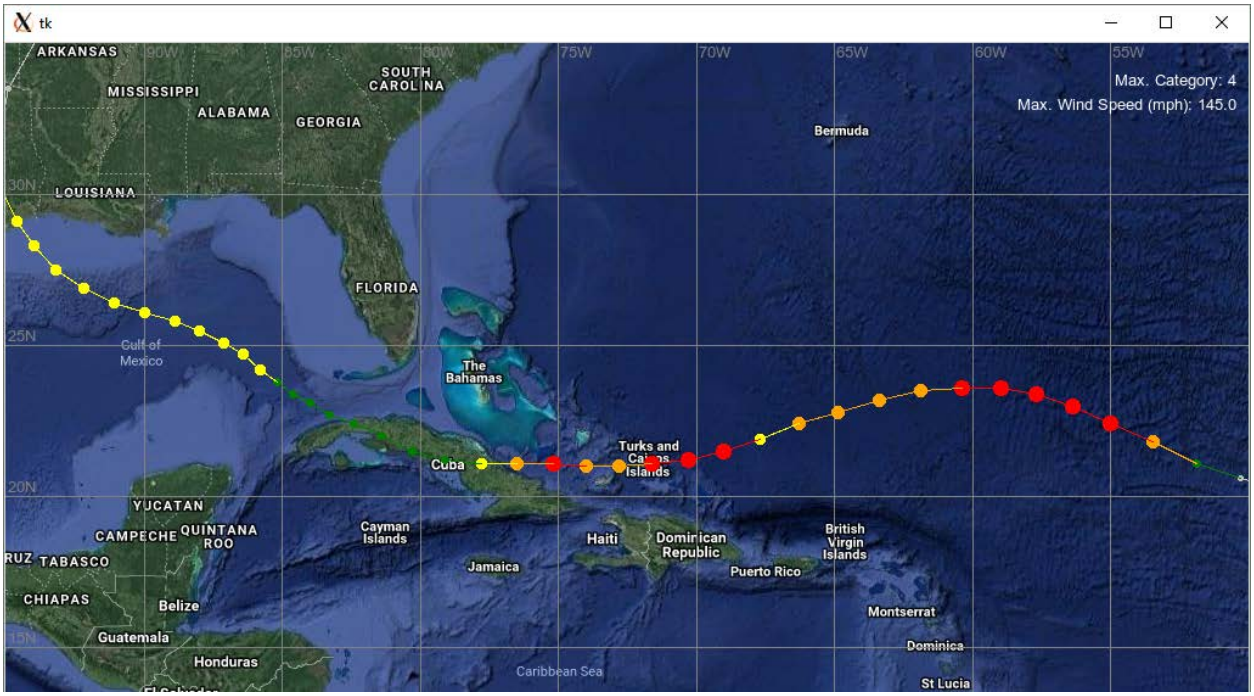
Your next task is to connect the dots that you drew previously so that the hurricane's track is a line instead of a collection of disconnected points. Two points are required to draw a line segment. As a result you will need the (x, y) location of the previous dot, as well as the (x, y) location of the current dot, in order to draw the line segment. The previous x and y values can be retained by storing them into appropriately named variables at the bottom of the loop. Then these variables can be read in the next loop iteration to draw the line segment, noting that such a line segment should not be drawn the first time the loop executes. The color of the line segment should match the color of the current dot.

The connected track for Hurricane Andrew is shown below:



Part 5: Classifying the Storm

After all of the dots and lines have been drawn your program should report the storm's maximum category and maximum wind speed in the upper right corner of the window. Report a maximum category of 0 for tropical storms and tropical depressions. This data is shown for Hurricane Ike below. Note that the window has been cropped so that it will fit on this page.



Additional Specifications:

Your program should follow good programming practices. This includes using meaningful variable names, avoiding the use of magic numbers and including adequate comments. Magic numbers you should specifically avoid using include:

- The gap between the lines of latitude and longitude drawn on the map
- The dimensions of the map image
- The latitude / longitude values for the boundary of the map.

Ideally, if you wanted to use a different map then you should only need to change the image dimensions and boundary values in one place in your program and then every would continue to work correctly. You should also ensure that you include a comment at the top of the file with your name, student number and a brief description of the purpose of your program.

Break and continue are generally considered bad form. As a result, you are **NOT** allowed to use them when creating your solution to this assignment. In general, their use can be avoided by using a combination of if statements and writing better conditions on your while loops.

You do not need to perform any error checking in your program. In particular, you may assume that the user always enters valid floating-point numbers. Any latitude / longitude values that are outside of the window should be drawn just like values inside the window. SimpleGraphics will simply ignore any requests to draw outside of the window.

Your program should leave the window open until the user closes it by clicking on the close button.

Grading:

This assignment will be graded on a combination of functionality and style. A base grade will be determined from the general level of functionality of the program (Does it draw the map successfully? Are the lines of latitude and longitude drawn correctly? Does it plot the dots in the correct location? Are the correct colors used for storms in different categories? Is the hurricane's track connected properly?). The base grade will be recorded as a mark out of 12.

Style will be graded on a subtractive scale from 0 to -3. For example, an assignment which receives a base grade of 12 (A) because it generates perfect output, but has several stylistic problems such as magic numbers, poor variable names and missing comments resulting in a -2 adjustment will receive an overall grade of 10 (B+).

Total Score (out of 12)	Letter Grade
12	A
11	A-
10	B+
9	B
8	B-
7	C+

6	C
5	C-
4	D+
3	D
0-2	F

Looking for an A+? Adjust the program so that the color of each line segment matches the stronger of the two storms at its end points. The revised track for Hurricane Andrew is shown below with the differences marked.