CS410C FALL 2020 DUE: 10/7/20

LAB 7L

Purpose

The purpose of this assignment is to give you practice with writing functions and working with "for" in a loop.

Problem

In 4L you wrote a program to take two pairs of (x, y) values and determine where they were on a Cartesian plane, and also compute the distance between the two points.

Modify this program to create a function called where_is_xy() that takes x and y as input arguments (float) and does exactly what it did before, it prints information about where the (x, y) point lies on the Cartesian plane.

Your main function needs to accept 10 pairs of (x, y) values in a "for" loop and for each (x, y) value it calls the where_is_xy() function which prints the location information of the point.

Next, we want to compute the distance between each consecutively input (x, y) points using the same technique as before:

$$D = \sqrt{dx^2 + dy^2}$$

where dx is the difference between the x-coordinates of the points and dy is the difference between the y-coordinates of the points.

This time we want to create a second function called compute_distance() which takes 4 input arguments (x1, y1, x2, y2) for the two (x, y) pairs for computing distance and returns the distance between them as a float.

In your main function you need to have a "for" loop that goes for 10 iterations. In this loop you will accept a pair of (x,y) values as input and call where_is_xy() which prints the location of the (x, y) point. In the same loop you should also call the compute_distance() function every time ***after*** the second pair of points are input (i.e you start computing distances between the 1st & 2nd pairs, 2nd and 3rd pairs, and so on and end it after the last pair of points is entered to compare the 9th and 10th pairs). In order to do this effectively, save the "previous" (x, y) pair of values in two separate variables so that you can compute distance between the "current" pair and the "previous pair.

Finally after all the points are input and distances between each consecutive points are computed, we want to print out the maximum distance that was encountered and the two points that were used for this distance (to do this, you need to check for the maximum every time inside your loop!). To start with assume/save the maximum distance (you need another variable to maintain this) is the first distance computed, as you compute distance between pairs check the "current" distance with the saved maximum distance and replace it if the "current" distance is greater. While saving the maximum distance you should also save the 2 pairs of (x, y) values that contribute to this maximum distance.

In the public folder for 7L you will find a sample starter file containing all the function prototype definitions and the function definitions. All you need to do is to fill in the functions with the relevant code most of it should already be available in your 4L submission! If you had omitted doing some things in your 4L, you might want to do them now so that you don't lose points again.

Requirements

- Your source file should be called cartesian.c – there is a starter file available for download from the public folder. The starter file defines all prototype function declarations and contains the stubs for all functions.
- All output values must use %0.2f for format specifier i.e. 2 decimal digits. Your program output does not have to look exactly like mine but the results for point locations, distances, maximum distance should match mine.
- You must use a "for" loop in your main() function to accept values, call the proper functions and print results for every pair of input values.
- You cannot use "break" and "continue" statements in ANY of your programs.

Sample Input & Output

Sample input and output files are available. It is recommended that you use input redirection to test against my input file and make sure that your output values match with mine as closely as possible.

All floating point output should be correct to 2 decimal places.

Locations of the points must be one of these values exactly (needs to match phrases for tests to pass): Quadrant 1, Quadrant 2, Quadrant 3, Quadrant 4, X-Axis, Y-Axis, Origin

Since we have 10 pairs of values to input, it does get tedious to do so manually. On Linux you can use "input redirection" for input to come directly from a file. In other words if you compile the program in the following manner:

```
gcc cartesian.c -o cartesian
```

The executable/program name is called "cartesian"

```
./cartesian < in1
```

will run the points executable and seek input from a file called "in1" in the same folder. You can save the input file as any text file (with or without .txt, just specify the file extension also if you have one). You can input values manually as well when the program executes if you are not able to figure out how to use input indirection.

Submission/Test Cases

- Tests t1 through t16 carry grade points for testing locations & distances
- Rubric items t17 through t20 are manual items for coding, output formatting as indicated in the grade key below

Grade Key

Locations of points (7 locations 3 points each)	**21**
Distances between adjacent points (9 distances, 5 points each)	**45**
Max distance and points	**7**
Rubric Item t17 - For loop correctly used in main()	**7**
Rubric Item t18 - Function where_is_xy() correctly coded	**6**
Rubric Item t19 - Function compute_distance() correctly coded	**6**
Rubric Item t20 - Output formatting reasonable	**8**