Homework 2 -- Out: Sept. 21, Thursday -- DUE: Oct 2, Monday, 12:00pm (noon)

EC327 Introduction to Software Engineering – Fall 2023

Total: 100 points

Submission: ***Failure to follow the guidelines WILL result in loss of points on the assignment.***

1. Completed assignments **must** be submitted on GradeScope.
2. Write your answers clearly in your favorite text editor, and submit a .**cpp** file for each question (Q1.cpp, Q2.cpp, Q3.cpp) ZIPPED into a file named: **BUusername_HW2.zip**
   For example: **acoskun_HW2.zip** -> Has Q1.cpp Q2.cpp Q3.cpp
3. Please make sure your code compiles and runs as intended on the engineering grid. Code that does not compile will NOT be graded and will receive a 0.

Late Homework Submissions:

Recall the late submission policy and the fixed penalty of 20% for submitting up to 48 hours after the deadline. Also recall that we only grade your latest submission (e.g., if you submit both on time and after the deadline, only your late submission is graded).

Q1. *Intersecting Circles.* [30 points]

Write a program (Q1.cpp) that reads integer (x,y) center coordinates and radius (r) values of two circles from the console and checks whether the areas of given circles overlap or not. If the areas of the circles overlap by any amount, the program should state so. Otherwise, the program should calculate and print the minimum distance between the circles.

Symbols < > in the examples demonstrates the user inputs entered via the keyboard.
The actual user inputs should NOT contain < or >.

Sample run 1:
Enter the coordinates and radius for the first circle
x: <2> <enter>
y: <0> <enter>
r: <4> <enter>
Enter the coordinates and radius for the second circle
x: <4> <enter>
y: <1> <enter>
r: <2> <enter>
THE CIRCLES OVERLAP.

Sample run 2:
Enter the coordinates and radius for the first circle
x: <-2> <enter>
y: <-10> <enter>
r: <4> <enter>
Enter the coordinates and radius for the second circle
x: <10> <enter>
y: <0> <enter>
r: <2> <enter>
THE MINIMUM DISTANCE BETWEEN THE CIRCLES IS 9.62.

**You should display at least two fractional digits (e.g., 9.62 above in sample run 2) if the circles do not intersect.*

IMPORTANT NOTE: YOUR CODE WILL GO THROUGH AN AUTOGRADER SO MAKE SURE YOUR CODE TAKES THE INPUTS IN THE SAME ORDER AS THE ABOVE EXAMPLE AND ITS OUTPUT IS EXACTLY "**THE CIRCLES OVERLAP.**" OR IN THE FORMAT "**THE MINIMUM DISTANCE BETWEEN THE CIRCLES IS 9.62.**" AS IN THE ABOVE EXAMPLE (USE UPPER-CASE LETTERS AND THE SAME WORDING).

Q2. *Displaying floating points.* [30 points]

Recall the floating point representation we have discussed in class. Floating point can also be represented in *normalized* form. Normalization basically puts the *radix point after the first non-zero digit*. Write a program

(Q2.cpp) that asks the user to enter a number and normalizes this input number. The code should then print the normalized mantissa and the exponent (in base 10) as shown in the example.

Text in < > in the examples demonstrates the user inputs entered via the keyboard.
The actual user inputs should NOT contain < or >.

Sample run 1:
Enter a fractional number: <0.523431> <enter>
In normalized form:
Mantissa: 5.234, Exponent: -1
Sample run 2:
Enter a fractional number: <-523431> <enter>
In normalized form:
Mantissa: -5.234, Exponent: 5
Sample run 3:
Enter a fractional number: <5.23431> <enter>
In normalized form:
Mantissa: 5.234, Exponent: 0

You must display **only three fractional digits.*

IMPORTANT NOTE: YOUR CODE WILL GO THROUGH AN AUTOGRADER SO MAKE SURE YOUR CODE TAKES THE INPUTS IN THE SAME ORDER AS THE ABOVE EXAMPLE AND ITS OUTPUT IS EXACTLY "**Mantissa: 5.234, Exponent: 0**" AS IN THE ABOVE EXAMPLE.

Q3. *Computing the number of different digits.* [40 points]
Write a program (Q3.cpp) that prompts the user to enter two 32-bit non-negative integers in decimal form. The program should then compute the number of digits that need to be changed in the first number to obtain the second number when the numbers are represented in *hexadecimal format*. Consider the examples below:

Integer 1	Integer 2	Integer 1 in hex	Integer 2 in hex	Number of hex digits to change
257441082	256391225	**0xF583D3A**	**0xF48383**9	3
19211483	74971	**0x12524DB**	**0x00124DB**	3
8	5004	**0x0008**	**0x138C**	4

The output should be displayed as follows:

Sample run 1:
Enter two non-negative integers: <19211483> <enter>
<74971> <enter>
These numbers differ in 3 hex digits.
Sample run 2:
Enter two non-negative integers: <8> <enter>
<5004> <enter>
These numbers differ in 4 hex digits.

Hint: Think about how bitwise operators can help identify different bits across two numbers.

IMPORTANT NOTE: YOUR CODE WILL GO THROUGH AN AUTOGRADER SO MAKE SURE YOUR CODE TAKES THE INPUTS AS SHOWN IN THE ABOVE EXAMPLE AND ITS OUTPUT IS IN THE SAME FORMAT AS THE EXAMPLE, i.e., "These numbers differ in 4 hex digits."