CSCE 240 – Exam One

Due: 11:59pm on Wednesday, February 8. Late exam submissions will not be accepted.

This is an exam. As you work on these problems, you may use your textbook, class notes, and the recorded lectures. You may ask your instructor clarifying questions. You are not to discuss the problems with other students or seek help from other individuals. All work submitted must be your own. All code submitted will be examined for plagiarism and violations will be reported to the office of Student Conduct and Academic Integrity.

Test all of your code on a Linux lab computer. All source files submitted must compile and run on a Linux lab computer of the instructor's choice. Submissions that do not compile on the Linux workstation will receive no compilation or execution/correctness points.

You should submit only the following individual files to the exam assignment in Blackboard: problem1.cc, problem2funcitons.h, problem2functions.cc, problem3functions.h, problem3functions.cc

Problem 1

Deliverable: problem1.cc

Purpose: Create a program that will accept characters input from the standard input device (using cin). The program should stop accepting input when a period, question mark, or exclamation point is input (. ? or !). The program will output the number of alphabetic characters (a-z A-z) and the number of digits (0-9) that were input in the form "Input sentence contains *number* alphabetic *character/characters* and *number digit/digits*." Note: the words *character* and *digit* should be singular or plural as appropriate.

Specifications:

- Do not prompt for the program input.
- Assume that two valid integers will be input from the standard input device (using cin).
- Produce all output to the standard output device (using cout).

Sample input output pairs:

Input: Count characters in this sentence.
Output: Input sentence contains 29 alphabetic characters and 0 digits.

Input: Hey, do you count 12 letters?
Output: Input sentence contains 20 alphabetic characters and 2 digits.

Input: 1 more win!
Output: Input sentence contains 7 alphabetic characters and 1 digit.

Initial Testing:

A makefile, checkit.cc source file, and sample input and output files have been provided to run initial tests on your program. You are encouraged to create more rigorous tests. To run the tests provided, create a directory containing only your problem1.cc file and the files extracted from the attached problem1tests.zip. Then type the following commands at the command prompt:

```
make problem1test1
make problem1test2
make problem1test3
```

Points:

style: 1 point

clean compilation: 1 point

execution / correctness (passes instructor tests): 2 points

Problem 2

Deliverables: problem2functions.h and problem2functions.cc

Functions

- Write a function named *Distance* that takes four double arguments for the x and y coordinates of two points (x1, y1, x2, y2), and returns the distance between those points as a double. If the function is called with the coordinates of one point only, the second point's coordinates should default to the origin (0, 0).

  Recall, the formula to compute the distance, d, between two points is:

  $$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

  Example function calls:

  Distance(3, 4) should return 5 (the distance between the point (3, 4) and the point (0, 0))

  Distance(-1, 3, 11, -2) should return 13 (the distance between the point (-1, 3) and the point (11, -2))

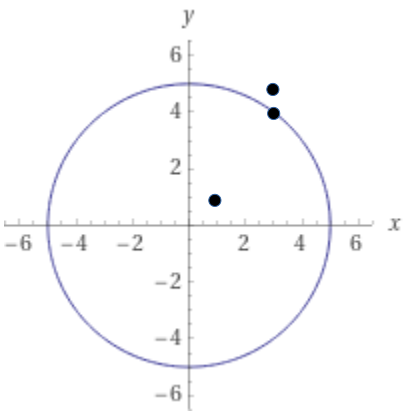
- Write a function named *OnCircle* that takes the x and y coordinates of the center of a circle as the first two parameters, the radius of the circle as the third parameter, and the x and y coordinates of a point as the fourth and fifth parameters (all doubles). The function should return -1 if the point lies inside of the circle, 0 if the point lies on the circle, and 1 if the point lies outside of the circle.

  Example calls:

  OnCircle(0, 0, 5, 1, 1) should return -1 because the point (1, 1) lies inside the circle centered at (0, 0) with radius 5

  OnCircle(0, 0, 5, 3, 4) should return 0 because the point (3, 4) lies on the circle centered at (0, 0) with radius 5

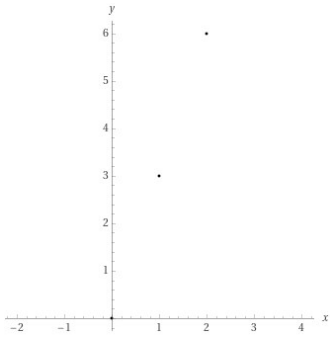
  OnCircle(0, 0, 5, 3, 5) should return 1 because the point (3, 5) lies outside the circle centered at (0, 0) with radius 5



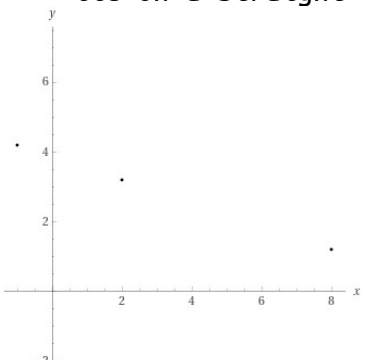
- Write a function named *OnLine* that takes the x and y coordinates (all doubles) of three points (x1, y1, x2, y2, x3, y3). The function should return true if the three points lie on a straight line, and false if they do not.

  Example calls:

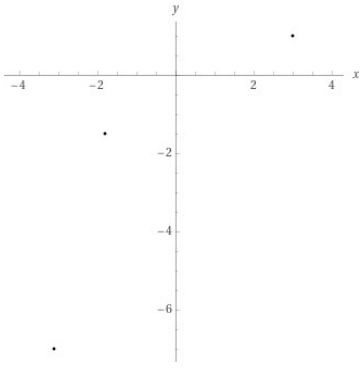
  OnLine(0, 0, 1, 3, 2, 6) should return true because the three points (0, 0) (1, 3) and (2, 6) lie on a straight line



  OnLine(2, 3.2, -1, 4.2, 8, 1.2) should return true because the three points (2, 3.2) (-1, 4.2) and (8, 1.2) lie on a straight line



  OnLine(-1.8, -1.5, 3, 1, -3.1, -7) should return false because the three points (-1.8, -1.5) (3, 1) and (-3.1 -7) do not lie on a straight line



Specifications:

- All three function prototypes should be included in problem2functions.h
- All three functions should be implemented in problem2functions.cc

Initial Testing:

A makefile and some minimal unit tests have been included in problem2tests.zip. You are encouraged to create more rigorous tests. To run the tests provided, create a directory containing only your problem2functions.h file, your problem2functions.cc file, and the files extracted from the attached problem2tests.zip. Then type:

```
make testDistance
make testOnCircle
make testOnLine
```

Points:

style: 1 point

clean compilation: 1 point

Distance passes instructor's unit tests: 1 point

OnCircle passes instructor's unit tests: 1 point

OnLine passes instructor's unit tests: 2 points

Problem 3

Deliverables: problem3functions.h and problem3functions.cc

Functions

- Write a function named *NumDigits* that takes an integer parameter and returns the number of digits in the integer.

  Example calls:

  NumDigits(1078) should return 4

  NumDigits(-10) should return 2

- Write a function named *FindAndReplace* that replaces digits within an integer variable.

  The function will take an integer variable to search, a positive integer to find, and a positive integer to replace as arguments. The function should find all occurrences of the find integer within the integer variable and replace them with replace integer, moving from right to left.

  Preconditions: the second and third arguments should be positive integers, and the second argument should contain at least as many digits as the third argument. If the preconditions are not met, the function should return false and leave the first argument unchanged.

  Example calls:

  example 1:

  x = 12325;

  FindAndReplace(x, 2, 7); - the function should return true, and after the function call x should hold 17375

  example 2:

  x = 11811;

  FindAndReplace(x, 11, 6); - the function should return true, and after the function call x should hold 6806

example 3:

x = 118111;

FindAndReplace(x, 11, 23); - the function should return true, and after the function call x should hold 238123

example 4:

x = 10345;

FindAndReplace(x, 3, 10); - the function should return false because the second argument has fewer digits (1) than the third argument (2), and after the function call x should hold 10345

example 5:

x = 18;

FindAndReplace(x, -1, 1); - the function should return false because the second argument is negative, and after the function call x should hold 18

Specifications:

- Both function prototypes should be included in problem3.h
- Both functions should be implemented in problem3.cc

Initial Testing:

A makefile and some minimal unit tests have been included in problem3tests.zip. You are encouraged to create more rigorous tests. To run the test provided, create a directory containing only your problem3functions.h file, your problem3functions.cc file, and the files extracted from the attached problem3tests.zip. Then type

```
make testNumDigits
make testFindAndReplace
```

Points:

style: 1 point

clean compilation: 1 point

NumDigits passes instructor's unit tests: 1 point

FindAndReplace passes instructor's unit tests: 2 points