ENGG1340 / COMP2113 Programming Techniques (2019-2020) Assignment 1

General Instructions

Please read the instructions in this document carefully. In this assignment you will solve six tasks and a tester would automatically test your submitted program. Sample test cases are provided with each task in this document (**user inputs are printed in blue**). However, please note that we will also use additional test cases when marking your assignment submission.

Total: 100 points
5 points for proper code comments and indentation
95 points for program correctness

Input and output format

Your C++ programs should read from the **standard input**. Also, your answer should be printed through the **standard output**. If you failed to follow this guide, the tester may not able to give a score for your program. Additionally, you should strictly follow the sample output format (including space, line breaker, etc.), otherwise, your answer might be considered as wrong.

Submission

Please name your C++ programs as the following table shows and put them together into one directory. Compress this directory as a *.zip file. Make sure to only upload the source files (*.cpp) above. **Do not submit any other files**. There should only be the 6 files in the zip archive. Please **use your university number to name the zip archive** and check carefully that the correct files have been submitted. We suggest you download your submitted files, extract them, and check for correctness. **You will receive 0 marks for this assignment if you submit incorrect files.** Resubmission after the deadline is not allowed.

Filename	Description
1.cpp	Task 1
2.cpp	Task 2
3.cpp	Task 3
4.cpp	Task 4
5.cpp	Task 5
6.cpp	Task 6

Deadline

The due date is 23:55, 15 Dec, 2019. The cut-off date is 23:55, 22 Dec, 2019. **No late submission after the cut-off date will be accepted**.

Late submission

You will receive 0 marks if you submit after the cut-off date.

Wrong submission

You will receive 0 marks if you submit incorrect files.

Evaluation

Your code will be tested for technical correctness. However, the correctness of your implementation – not the tester's judgments – will be the final judge of your score. If necessary, we will review your work individually to ensure that you receive due credit for your work.

Academic Dishonesty

We will be checking your code against other submissions in the class and from the Internet for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know. We trust you all to submit your own work only; please don't let us down. If you do, we will pursue the strongest consequences available to us.

Getting help

You are not alone! If you find yourself stuck on something, post your question to the course forum. We want this assignment to be rewarding and instructional, not frustrating and demoralizing. But we don't know when or how to help unless you ask.

Discussion Forum

Please be careful not to post spoilers. Please don't post any code that is directly related to the assignments. Whenever you are welcome and encouraged to discuss general ideas on the discussion forums. If you have any questions about this assignment you should post them in the discussion forums.

Task 1 Puppies

(10 points)

Here you will use C++ to solve the problem. Note that **the input / output requirements are different** though.

Queenie is living together with n puppies. Queenie is worried to lose sight of her puppies. Write a C++ program that let Queenie know which of her puppies are *too far away* from Queenie. A puppy p at position (p_x, p_y) is too far away from Queeny at (q_x, q_y) if

$$Sqrt\ ((p_x - q_x)^2 + (p_y - q_y)^2) > 10,$$

where Sqrt (x) denotes the square root of x

The position of Queenie, the number n, and the puppy positions are given by the user and you can assume all positions are given with Integers.

The input contains multiple lines: the first line contains two integers q_x and q_y representing the position of Queenie, the second line contains first an integer n for the number of puppies and then n pairs of integers (p_x and p_y) indicating the locations of **Puppy 1 to Puppy *n***.

Your program should output the puppy numbers (in the range of 1 to n) of the puppies which are too far away from Queeny on the first output line, and then the total number of puppies which are too far away on the second output line. Check the sample test cases below for the required output format.

Note: You may use sqrt(n) defined in the header <cstdlib> to compute the square root of an integer n.

Sample test case 1.1
2 1
4 10 1 14 -2 1 3 0 4
Puppy 2 too far away
Total 1 puppy too far away

Sample test case 1.2
2 1
4 15 15 14 -2 1 3 0 4
Puppy 1 and Puppy 2 too far away
Total 2 puppies too far away

Sample test case 1.3
2 1
4 15 15 14 -2 16 16 0 4
Puppy 1 and Puppy 2 and Puppy 3 too far away
Total 3 puppies too far away

Sample test case 1.4
2 1
0
No puppies too far away

Sample test case 1.5
0 0
2 5 1 6 1
No puppies too far away

Task 2 Exponential Calculation

(15 points)

Write a C++ program which takes two user input integers x ($0 \le x \le 30$) and n, and calculate an estimation of e^x using the formula:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \frac{x^n}{n!}, \qquad -\infty < x < \infty$$

where n in this question assumes the range from 0 to 100, $i! = 1 \times 2 \times 3 \times \ldots \times i$ denote the factorial of i.

Your program should output the estimations of e^x as n increases. Specifically, your program should

- Output the values of n and e on the subsequent lines as n increases. The values n and e on each line are separated by a space. **The value e is printed as a fixed floating number with 8 decimal places**.
- Use the double data type for floating point calculations.

Note:

- **Check carefully your results against the same test outputs.** Divisions involving some large numbers can easily run into numerically inaccuracy issues (you will learn more about this in the machine organization course).Try to implement the given formula in different ways to test out.
- You may use setprecision(n) defined in the header <iomanip> to set the precision parameter of the output stream. Or you may use the C-style printf to format your output if you like.

Sample test case 2.1

```
0 0
0 1.00000000
```

Sample test case 2.2

```
1 4
0 1.00000000
1 2.00000000
2 2.50000000
3 2.66666667
4 2.70833333
```

Sample test case 2.3

```
3 20
0 1.00000000
1 4.00000000
2 8.50000000
3 13.00000000
4 16.37500000
5 18.40000000
6 19.41250000
7 19.84642857
8 20.00915179
9 20.06339286
10 20.07966518
11 20.08410308
12 20.08521256
13 20.08546859
14 20.08552346
15 20.08553443
16 20.08553649
17 20.08553685
18 20.08553691
19 20.08553692
20 20.08553692
```

Task 3 Digits Permutations

(20 points)

Write a C++ program that will read one positive integer *M* supplied by the user, and your program should output the number of all permutations of the digits, and the list of the permutations in an increasing order. If given input is 123 then your program should print an integer 6 with all 6 permutations in an increasing order e.g. 6 123 132 213 231 312 321

Note:

- If there are duplicated permutations, eliminate the duplicated one and only keep one permutations. For example, 121, will have 3 permutations which are 112 121 211.
- If there are leading 0(s) in any permutation, eliminate the 0(s). For example, 120, will have permutations which are 12 21 102 120 201 210

Sample test case 3.1
1
1 1

Sample test case 3.2
123
6 123 132 213 231 312 321

Sample test case 3.3
121
3 112 121 211

Sample test case 3.4
120
6 12 21 102 120 201 210

Sample test case 3.5
100
3 1 10 100

Task 4 Bounding Boxes

(20 points)

Write a C++ program to compute a **minimum-sized** axis-aligned bounding box (AABB) for a set of input geometries. An AABB is a rectangle with sides parallel to the x-, y-axes which encloses some given geometries. The input and output of your program are as follows:

Input. Each line of the user input begins with a character indicating the type of geometry, followed by some parameters of the geometric object. The input line can be one of the followings:

- R *x y width height*
  where R represents an input rectangle, *x*, *y* are floating-point numbers for the x-, y-

coordinates of the rectangle center, *width* and *height* are floating-point numbers for the rectangle size along the x- and y-axes, respectively.

- C *x y radius*
  where C represents an input circle, *x*, *y* are floating-point numbers for the x-, y-coordinates of the circle center, and *radius* is a floating-point number for the radius of the center
- P $n\ x_1\ y_1\ x_2 y_2 \ldots x_n y_n$
  where P represents an input point set, *n* is an integer indicating the number of points in the set, and xi, y_i, $i = 1, \ldots, n$, are floating point numbers for the x-, y-coordinates of the *n* points
- #
  indicates end of input

Output. A single line

x y width height

where *x* and *y* are floating-point numbers for the x-, y-coordinates of the center of the minimum-sized AABB, *width* and *height* are floating-point numbers giving the sizes of the AABB along the x-, y-axes, respectively.

Use the double data type for floating point calculations

Note: **The questions assume no bound for the floating-point parameters (*x*, *y*, *width*, *height*, *radius*) of the input geometries and therefore they can be any values that a double data type can hold.** You may use std::numeric_limits<double>::lowest() and std::numeric_limits<double>::max() defined in the header <limits> in your program to obtain the smallest and the largest possible values, respectively, for the double data type.

Sample test case 4.1

```
R 0 0 3 2
#
0 0 3 2
```

Sample test case 4.2

```
C  -0.5  3.2  1.6
#
-0.5 3.2 3.2 3.2
```

Sample test case 4.3

```
P  2  3  -2  -1  4
#
1 1 4 6
```

Sample test case 4.4

```
P 3  -1.5 3  3  3  5  3
#
1.75 3 6.5 0
```

Sample test case 4.6

```
P 2 3 -2 -1 4
C -0.5 3.2 1.6
P 3 -1.5 3 3 3 5 3
R 0 0 3 2
#
1.45 1.4 7.1 6.8
```

Task 5 One-shot Stock Trader

(10 points)

Suppose you are a Stock Trader. Given an array of daily Stock Price, you are supposed to make at most one transaction (firstly buy one and then sell one share of stock) so that you can find the maximum profit.

Input: A list of Integers which are the daily Stock Prices
Output: The maximum profit you can gain (Integer).

Sample test case 5.1 (Note: there is no output in this test case)
1 2 3 4 5
4

Sample test case 5.2
5 4 3 2 1
0

Sample test case 5.3
9 8 7 6 2 3 4 3 2 7 8
6

Task 6 Multi-shots Stock Trader

(20 points)

Suppose you are a Stock Trader. Given an array of daily Stock Price, you are supposed to make as many as you like transactions (firstly buy one and then sell one share of stock) so that you can find the maximum profit. Note that a new transaction must be made before the previous transaction ends (i.e. you can only hold at most one share of stock all the time).

Input: A list of Integers which is the daily Stock Prices
Output: The maximum profit you can gain (Integer)

Sample test case 5.1 (Note: there is no output in this test case)
1 2 3 4 5
4

Sample test case 5.2
5 4 3 2 1
0

Sample test case 5.3
9 8 7 6 2 3 4 3 2 7 8
8