CST 370 – Fall 2019
Homework 1
Due: 09/05/2019 (Thursday) (11:55 PM)

Name: ___Hamza Saleem________________

Class ID: ___6969____________________

How to turn in?

- Write your answer to the questions 1, 2, and 3 and submit it on the iLearn. Note that we **accept only a PDF** file. Do not submit a different file format. And also, don't forget to write your name and class ID at the top of your homework document.
- For the questions 4 and 5, you should submit your C++ source files (**hw1_1.cpp** and **hw1_2.cpp**) on the iLearn.
- So, you have to submit three files (one PDF file and two C++ source files) on the iLearn.
- Note that the due time is 11:55(PM). This is the iLearn's timestamp, not your submission time. Since there could be a long delay between your computer and iLearn, you should **submit early**.

1. Consider the following pseudocode. **Describe what this algorithm computes.**

```
Algorithm DoSomething (A[0.. n – 1])
num1 ← A[0];
num2 ← A[0]
i ← 1
while i < n do
    if A[i] < num1
        num1 ← A[i];
    if A[i] > num2
        num2 ← A[i];
    i ← i + 1
return (num2 – num1);
```

Sol:

First this program is declaring an array and initiating on variable num1 and num2. Then there is a loop for checking if number is smaller than first one if yes then num1 will become smallest number and number two will become 2^{nd} smallest number. Then it will return the difference of two furthest numbers.

2. (a) Find **gcd(140, 310)** by applying Euclid's algorithm. Present the **intermediate steps** of the algorithm clearly.

Sol:

Gcd (140, 310) = gcd(310,140) = gcd(140,30) = gcd(30, 20) = gcd(20/10) = 10

(b) Find **gcd(140, 310)** by applying the middle school approach (= prime factors) we covered in the class. Present the **intermediate steps** of the approach clearly.

Sol:
310 = 2 * 5 * 31
140 = 2 * 2 * 5 * 7
Gcd = 2 * 5 = 10

3. What is the maximum number of modulus operation(s) made by the Euclid's algorithm for the two numbers *m* and *n* in the ranges below? Present the number of modulus operation(s) and sample *m* and *n* for the case.

$1 <= m <= 100$
$1 <= n <= 100$

[**Hint**: Implement the Euclid algorithm in C++ (or Java). After that, add the code to display the number of modulus operation(s) for the numbers *m* and *n* in the ranges.]

Sol:

Number = 4 **m = 29, n = 8**

===

There are two programming problems in the homework.
This is the HackerRank link: https://www.hackerrank.com/cst370-f19-hw1

4. Write a C++ program (or Java program) called **hw1_1.cpp (or hw1_1.java)** that reads input numbers from a user and displays the closest distance between two numbers among all input numbers.

Input format: This is a sample input from a user.

```
5
7
-7
20
8
15
```

The first number (= 5 in the example) indicates that there will be five integer numbers in the input. Then, all numbers from the second line (= 7 in the example) to the last line (15 in the example) are actual numbers. Thus, your program should read them and present the closest distance among five numbers 7, -7, 20, 8, and 15. Because the distance between 7 and 8 is 1, your answer should be 1. Note that the distance between two numbers should be **always positive**. For the problem, you can assume that the actual **numbers are distinct (= no duplication)**.

Sample Run 1: Assume that the user typed the following six lines

```
5
7
-7
20
8
15
```

This is the correct output of your program.

```
Min distance: 1
Two numbers for min distance: 7 and 8
```

Note that **the sequence of two numbers for the minimum distance is important**. Your program has to display the small number first. So, the correct answer is 7 and then 8. If your program displays 8 and then 7, it’s not correct.

Sample Run 2: Assume that the user typed the following twelve lines

```
10
-1
10
-15
3
72
20
15
-20
5
30
```

This is the correct output of your program.

```
Min distance: 2
Two numbers for min distance: 3 and 5
```

For the program, you have to use a dynamic memory or a vector to store the input data. For details on the dynamic memory, refer to http://www.cplusplus.com/doc/tutorial/dynamic/ For the vector, refer to http://www.cplusplus.com/reference/vector/vector/

When you submit your homework program, don’t forget to include six items such as “**HackerRank link**”, “Title”, “Abstract”, “ID”, “Name”, and “Date”.

5. Write a C++ program (or Java program) called **hw1_2.cpp (or hw1_2.java)** that reads a set of ranges of two numbers from a user and displays the intersection of all ranges.

Input format: This is a sample input from a user.

```
3
1 6
-2 8
3 7
```

The first number (= 3 in the example) indicates that there will be three ranges in the input. Then, the following lines are the actual ranges. For example, the line "1 6" indicates that there's a range from 1 to 6. The next line indicates that there's a range from -2 to 8. The last range is from 3 to 7. Since your job is to find the intersection of all ranges, the result should be from 3 to 6.
If **no such intersection** exists, then your program should **print -1**

Sample Run 1: Assume that a user typed the following four lines

```
3
1 6
-2 8
3 7
```

This is the correct output of your program.

```
3 6
```

Sample Run 2: Assume that a user typed the following three lines

```
2
-10 15
15 20
```

This is the correct output of your program.

```
15 15
```

Sample Run 3: Assume that a user typed the following five lines

```
4
8 20
2 7
0 4
```

```
4 7
```

This is the correct output of your program.

```
-1
```

For the program, you have to use a dynamic memory or a vector to store the input data. For details on the dynamic memory, refer to http://www.cplusplus.com/doc/tutorial/dynamic/ For the vector, refer to http://www.cplusplus.com/reference/vector/vector/

When you submit your homework program, don’t forget to include six items such as “**HackerRank link**”, “Title”, “Abstract”, “ID”, “Name”, and “Date”.