CST 370 – Fall (B) 2021
Homework 4
Due: 11/30/2021 (Tuesday) (11:55 PM)

How to turn in: Write **three programs** in **either C++ or Java** and submit them on Canvas before the due.

- You **can submit** your programs **multiple times** before the due. However, the **last submission will be used for grading**.
- You have to **submit three programs together**, especially at your last submission. **If you submit, for example, only one program** at the last submission, **we are able to see only that program when we grade** your homework.
- Due time is 11:55(PM). Since there could be a long delay between your computer and Canvas, you should submit it early.
- When you submit your homework program, don't forget to include "Title", "Abstract", "ID", "Name", and "Date".

1. Write a C++ (or Java) program called **hw4_1.cpp (or hw4_1.java)** that reads a number of input values and the values themselves. Then, your program should put all negative numbers in front of all positive numbers. Read this document for your reference: https://bit.ly/2tVfxKt

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

```
8
5 -3 1 -9 -8 2 -4 7
```

The first line (= 8 in the example) indicates that there are 8 integer values in the second line, and the actual 8 values in the second line.

Sample Run 0: Assume that the user typed the following lines

```
8
5 -3 1 -9 -8 2 -4 7
```

This is the correct output. Your program should display the results of the two approaches described in the document (= **https://bit.ly/2tVfxKt**) on the screen.

```
-4 -3 -8 -9 1 2 5 7
-3 -9 -8 -4 1 2 5 7
```

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

```
8
-4 3 9 -6 2 -5 8 7
```

This is the correct output.

```
-4 -5 -6 9 2 3 8 7
-4 -6 -5 3 2 9 8 7
```

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

```
5
-10 -30 25 -15 40
```

This is the correct output.

```
-10 -30 -15 25 40
-10 -30 -15 25 40
```

2. Write a C++ (or Java) program named **hw4_2.cpp** (or **hw4_2.java**) which displays the biggest number in an array with *n* integer numbers using **a divide-and-conquer technique**. For example, if your algorithm has an input array such as 1, 3, 11, 7, 5, 6, 4, 9, your algorithm should display 11.

In this program, you **have to use a divide-and-conquer technique** to display the max value. For the grading, we will read your source code. **If you do not use a divide-and-conquer technique** to find it, you will **get zero even if your program passes all test cases**.

Remember that a divide-and-conquer program should use a recursive function. Refer to a sample divide-and-conquer program to add the values in an array at https://repl.it/@YBYUN/sumdivNconqcpp

Sample Run 0: Assume that the user typed the following data

```
8
1 3 11 7 5 6 4 9
```

The first line (= 8 in the example) indicates the number of input data, and the following line shows the input values. This is the correct output of your program.

```
11
```

Sample Run 1: Assume that the user typed the following one line

```
3
-3 1 -5
```

This is the correct output of your program.

```
1
```

Sample Run 2: Assume that the user typed the following one line

```
4
10 99 99 10
```

This is the correct output of your program.

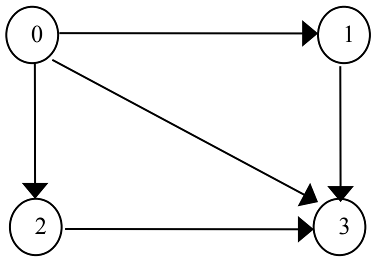
```
99
```

3. Write a C++ (or Java) program called **hw4_3.cpp (or hw4_3.java)** that conducts the topological sorting based on the Kahn algorithm covered in the lecture.

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

```
4
5
0 1
0 2
0 3
1 3
2 3
```

The first line (= 4 in the example) indicates that there are four vertices in the graph. For the homework, you can assume that the first vertex starts from the number 0. The second line (= 5 in the example) represents the number of edges in the graph, and following five lines are the edges. This is the graph with the input data.



Sample Run 0: Assume that the user typed the following lines

```
4
5
0 1
0 2
0 3
1 3
2 3
```

This is the correct output. Your program should display the numbers of incoming degrees of each vertex first. For example, the vertex 3 has three incoming degrees which is represented as "In-degree[3]:3". After the incoming degree information, your program should display the topological order as you learned in the class.

```
In-degree[0]:0
In-degree[1]:1
In-degree[2]:1
In-degree[3]:3
Order:0->1->2->3
```

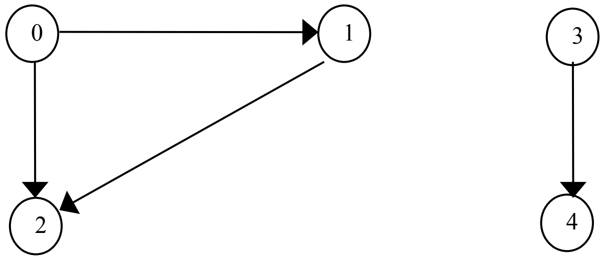
Sample Run 1: Assume that the user typed the following lines

```
5
4
0 1
1 2
0 2
3 4
```

This is the correct output.

```
In-degree[0]:0
In-degree[1]:1
In-degree[2]:2
In-degree[3]:0
In-degree[4]:1
Order:0->3->1->4->2
```

This is the input graph.



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

```
3
3
0 1
1 2
2 0
```

This is the correct output. Note that this graph is not a DAG (= directed acyclic graph) and there’s no topological order for a non-DAG.

```
In-degree[0]:1
In-degree[1]:1
In-degree[2]:1
No Order:
```