CST 370 – Summer A 2020
Homework 5
Due: 06/02/2020 (Tuesday) (11:55 PM)

Name:_____Andrew Bell_____

ID: ____1138_____

How to turn in?

Write your answer to the questions from 1 to 5 on a PDF file and submit it on the iLearn. Note that we **accept only a PDF** file. Do not submit a different file format. Also, don't forget to write your name and class ID at the top of your homework document.

For the questions 6, 7, 8, and 9, you should submit three source programs on the iLearn.

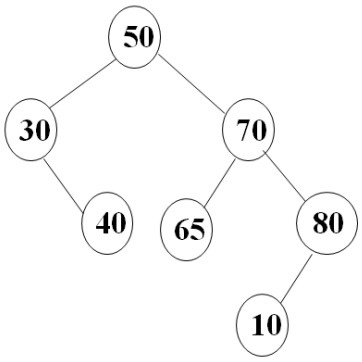
Thus, you have to submit total five files (one PDF file and four source files) on the iLearn.

1. Consider the quicksort algorithm covered in the class. Present the first partitioning operation for the list "6 8 2 5 3 7 1 9 ", You should use the first number, 6, as a pivot for the partitioning. In your answer, you have to present the indexes i and j clearly.

[Note: If it is difficult to draw the operation on the word file, draw it on paper by hand. Then take a picture of the paper and attach the picture here.]

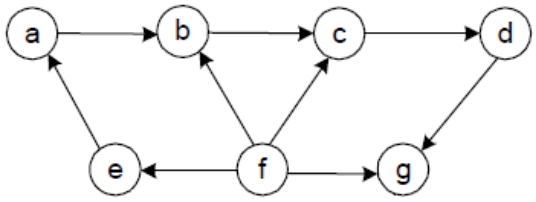
```
   i             j
6  8  2  5  3  7  1  9

   i          j
6  8  2  5  3  7  1  9

   i          j
6  1  2  5  3  7  8  9

            i,j
6  1  2  5  3  7  8  9

         j  i
6  1  2  5  3  7  8  9

         j  i
3  1  2  5  6  7  8  9
```

2. Traverse the following binary tree in preorder, inorder, and postorder. Then, present the result of each traversal.



preorder: 50,30,10,40,70,65,80
inorder: 30,10,40,50,70,65,80
postorder: 30,10,40,70,65,80,50

3. Apply the Kahn's algorithm to solve the topological sorting problem for the following digraph. Present the "in-degree" array and the topological order clearly.



f→ e → a → b → c → d → g

4. Refer to the puzzle named "A King's Reach" covered in the "week_6_4" file. In the puzzle, we assumed that a king piece can move to a horizontally, vertically, or diagonally adjacent square. Then, we identified how many different squares the king can stay after *n* moves.

For this homework problem, assume that a king piece can move to a horizontally or vertically adjacent square. It can't move to a diagonally adjacent square. Answer to the following questions.

(a) For n is 1, a king can stay ___4___ different squares.

(b) For n is 2, a king can stay ___9___ different squares.

(c) For n is 3, a king can stay ___16___ different squares.

(d) Based on the answers above, present a formula for the general number *n*.

(n+1)^2

5. Outline an algorithm to find the largest key (= number) in a binary search tree **in English**. What is the worst case time complexity of your algorithm?

To find the largest key, you would check if the current node has a right and if it has a right make the right node the current node and repeat until there is no right node and return that as the smallest node. Assuming the tree is balanced the time complexity would be logn or the height of the tree.

==========

This is the HackerRank link: https://www.hackerrank.com/cst370-su20-hw5

6. Write a C++ (or Java) program named **hw5_1.cpp** (or **hw5_1.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 covered in "week_4_5" 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
```

7. Write a C++ (or Java) program called **hw5_2.cpp (or hw5_2.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. Remember that we covered this problem in a puzzle, and read this document: https://bit.ly/2tVfxKt for your reference.

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.

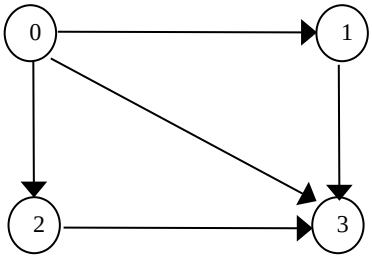
```
-10 -30 -15 25 40
-10 -30 -15 25 40
```

8. Write a C++ (or Java) program called **hw5_3.cpp (or hw5_3.java)** that conducts the topological sorting based on the Kahn's algorithm covered in the class.

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
```

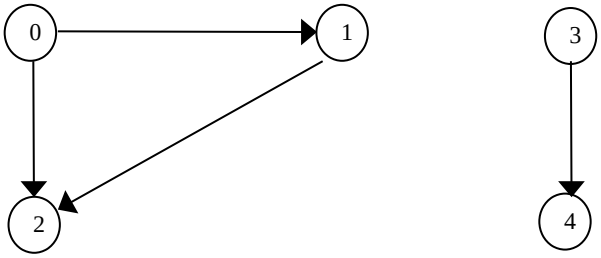
Sample Run 1: Assume that the user typed the following lines

```
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.

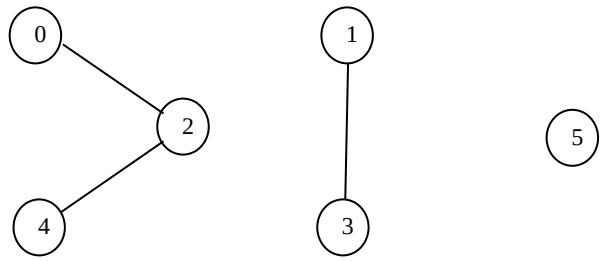
```
In-degree[0]:1
In-degree[1]:1
In-degree[2]:1
No Order:
```

9. Write a C++ (or Java) program called **hw5_4.cpp (or hw5_4.java)** that connects several connected components of a graph with minimum number of edges to create a single connected component of the graph.

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

```
6
6
0 2
2 0
2 4
4 2
3 1
1 3
```

The first line (= 6 in the example) indicates that there are six vertices in the graph. For the homework, you can assume that the first vertex starts from the number 0. The second line (= 6 in the example) represents the number of edges in the graph, and following six lines are the edges. This is the graph with the input data. Note that there are three connected components in the graph with the vertices of {0, 2, 4}, {1, 3}, and {5}.



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

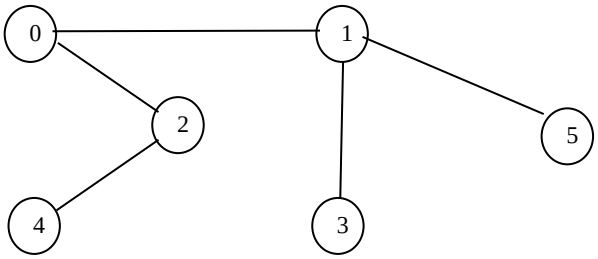
```
6
6
0 2
2 0
2 4
4 2
3 1
1 3
```

This is the correct output. Your program should display the new edge(s) to connect the three connected components to make a single connected component.

```
Edge: 0-1
Edge: 1-5
```

Because the input graph has three connected components, you need at least two new edges to connect them to make a single component. In this homework, you have to connect the vertices with the smallest

vertex numbers in each component. In other words, the vertices, 0, 1, and 5 are the smallest vertex numbers in each component. Thus, the new edges you need to connect are (0, 1) and (1, 5). This is the result graph after adding the two new edges.



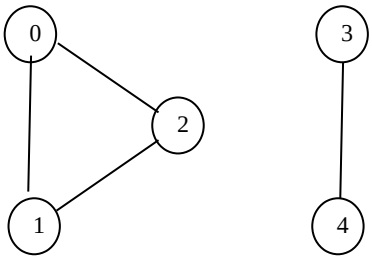
Sample Run 1: Assume that the user typed the following lines

```
5
8
0 1
1 0
1 2
2 1
4 3
3 4
2 0
0 2
```

This is the correct output.

```
Edge: 0-3
```

This is the input graph.



To make a single connected component, you need a minimum one edge. In our case, we connect the vertex 0 and vertex 3.

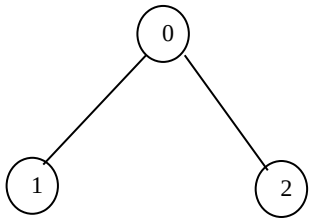
Sample Run 2: Assume that the user typed the following lines

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

This is the correct output.

```
No new edge:
```

This is the input graph. Note that the graph already has only one connected component. Thus, your program should display the message "**No new edge:**" to indicate that there is no edge necessary in this case.



Hint: Use the sample BFS program (https://repl.it/@YBYUN/BFS) to identify the connected components in a graph.