The objective of this lab is to:

understand Queue data structure and find FIFO behavior in different problems.

ALERT!

1. This is an individual lab, you are strictly **NOT** allowed to discuss your solution with fellow colleagues, even not allowed asking how is he/she is doing, it may result in negative marking. You can **ONLY** discuss with your TAs or with me.
2. Pay attention to **GOOD coding conventions** e.g.
   - Proper indentation.
   - Meaning variable and function names.
   - Use camelCase naming convention
   - Use meaningful prompt lines/labels for all input/output
3. **Anyone caught in act of plagiarism would be awarded an "F" grade in this Lab.**

Task 01: [5 Marks]

You have already implemented queue with templates as part of your homework. Now, add a member function *showStructure( );* in your queue ADT. The *showStructure();* function should display the queue status with its rear pointing to the correct location on the console.

Sample Run:

```
queue.Enqueue(5.0);
queue.Enqueue(6.5);
queue.showStructure();
```

```
5
6.5     <-- rear
```

```
queue.Enqueue(-3.0);
queue.Enqueue(-8.0);
queue.showStructure();
```

```
5
6.5
-3
-8      <-- rear
```

```
queue.Dequeue( );
queue.Dequeue( );
queue.showStructure();
```

```
-3
-8      <-- rear
```

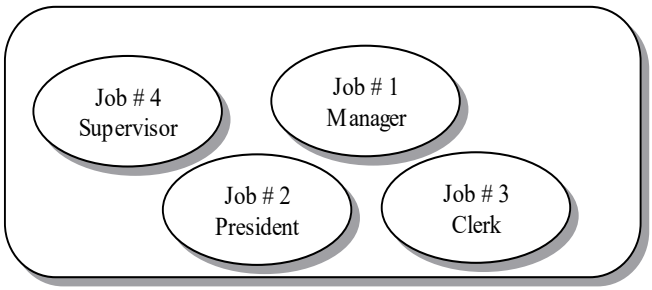
Task 02: [10 Marks]

We have already studied queue data structure in class and I believe you are well aware of queue data structure by now. So let's talk about a variant of queue data structure called, *priority queue*.

- A *priority queue* is a special kind of queue in which each element has an associated **priority** or value
- Unlike the FIFO queues, the order of deletion from a priority queue (e.g., who gets served next) is determined by the element priority. That is, the highest priority item is deleted first.
- Elements are deleted by order of priority rather than by the order in which they arrived in the queue.

For example:

You got bellow 4 jobs in your queue in sequential order. Obviously you don't want to keep president on waiting for long time so you would first serve president's job, then Manager's job, then supervisor's and at the end you'll serve the clerk. So jobs entered the queue in sequential order but will be removed in the order #2, #1, #4, #3.



Implement a priority queue with all the basic functionality of a queue. Also find time bound of each function. Selection and implementation of data members are up to you but should justify the need of your selected structures. Your main should display output like task 01 task.

Task 03: [10 Marks]

A Stack is a Last In First Out (LIFO) structure, i.e, the element that is added last in the stack is taken out first. In this task, our goal is to implement a Stack using Queue for which we will be using two queues and design them in such a way that pop operation is same as dequeue but the push operation will be a little complex and more expensive too.

Since we are using Queue which is First In First Out (FIFO) structure, i.e, the element which is added first is taken out first, so we will implement the push operation in such a way that whenever there is a pop operation, the stack always pops out the last element added. You may use the Queue ADT you have implemented in task 01.