COSC3331 - Data Structures & Algorithms II
Programming Assignment 2 **Due Date: Friday, Oct. 1 @ 11:59 pm**

Target Topic: Stacks and Queues

Important Note:
Throughout this document, you will find questions that appear in ***bold italics***. You must address these questions in a separate file that will be the report you must submit along with the working set of programs required for this assignment.
These reports will be an important portion of the grade for these assignments as they show your understanding of the programs you submit and the underlying rationale for the strategy you have adopted to solve these programming problems.
Devise your answers carefully and in an organized manner.
The working programs without justifying report to prove your understanding of your solution will not be granted a passing grade.

Here is the description of the assignment:

You are asked to use TWO STACKS and ONE QUEUE to implement an experimental SORTING algorithm. Obviously you will also use other variables and an array that will store the sorted sequence. You will populate this output array one item at a time as you sort your data. Please note that the goal of enforcing these limitations is for you to come up with a strategy that is based on the proper use of these specific data structures. We acknowledge that this sorting algorithm will be inefficient but as mentioned before, this assignment is about the challenge of utilizing these structures in the context of a sorting algorithm.

As for data we are using objects of type `Item` defined as follows:

`class Item`

Attributes:

```
int key;          //randomly generated in the range 1-10 inclusive
int timeStamp;    //The number representing the object's order of generation
                  //The first object instantiated will have 1 for this attribute,
                  //The second will have 2, etc.
```

Methods:

The constructor that receives a randomly generated number and a time stamp starting with 1 to the total number of items to be created.

The `toString` method to display the attributes of this object in the following format: For an `Item` with 5 for the `key` and the time stamp of 1, this method will generate the following String:

`5:1`

Note that these objects will be sorted based on their `key` attributes.
Important note: We ALLOW DUPLICATE values for keys in this assignment. This means that you MUST take this possibility into consideration and incorporate that in your algorithm.

As you notice the `timeStamp` is our guide to keep track of the duplicate values from their position in the original sequence to where they will end up in the sorted sequence.

Important Notes:

- Do not attempt to sort these objects based on their time stamp. Every sort algorithm must stick with a criteria and for us that is the key attribute.
- You are devising a new sorting algorithm, do not call pre-defined sorting algorithms provided by the textbook or worst from other sources to sort this data. You MUST come up with a strategy to use these data structures to sort data.

This is how you get started on this assignment:
Prompt the user for the size of data, not exceeding 20.

You set up a loop to instantiate that many objects of type class `Item` one at a time. The key attribute of these objects will be randomly generated in the range 1-10 inclusive, whereas their `timeStamp` attribute will be sequentially assigned, the first object gets 1, second 2, etc. You pass these two arguments to the constructor at the time of object instantiation.

At the beginning, you process each object one at a time, deciding where they need to go, your choices are two stacks and a queue or a variable that holds the object with the maximum or minimum key, based on your strategy.

The point is that you do not store all these `Item` objects in an input array and THEN get to sorting them. You get the objects one at a time and decide where they go based on their keys and your strategy.

After you are done with instantiating all your objects, you should have identified the object with the maximum or minimum key of all, that item (or items in case we have duplicate of max/min key) will be placed in the proper spot/spots in your output array, hence sorting the list one item at a time.

An additional note: Notice that if you go by finding the maximum key, your output array will be populated from the end, and if you go for the minimum, it will be populated from the beginning: Your Choice. I go with the maximum for these instructions.

After you found the object with maximum key of all, you have the rest of data to process and find the next max item/items. The rest of data is stored in one of your data structures. You continue the process of sorting the rest of the data set based on their keys. You notice the size of data now is short of one or possibly more, if we had a duplicate situation, compared to the previous step.

As stated above, this sorting algorithm will build the sorted sequence one item at a time. So you need to constantly move your data among the stacks and the queue to find the next item/s to be placed in the proper position/s. When done, the two stacks and the queue will be empty and your output array will be populated by the sorted data set.

These are what you need to output for this assignment:
ONLY FOR THE PURPOSE OF THIS ASSIGNMENT, YOU SHOULD ADD THE FOLLOWING TWO METHODS IN THE STACKX AND QUEUE CLASSES GIVEN IN THE BOOK:

```
displayStack()
displayQueue()
```

These display methods must call the `toString` methods of `Item` objects to display their attributes of keys and time stamps.
This is to monitor the contents of the stacks and the queues throughout implementation of your algorithm.
A sample label of the output at every step of your algorithm would be:

```
Stack1(bottom -> top):
Stack2(bottom -> top):
Queue(front -> rear):
OutputArray:
++++++++++++++++++++++++++++++++++++++
```

where the contents of each data structure will follow on each line, at every iteration of your process. For the output array, you can display zeros for the yet-to-be populated spots. Note that you must check for null spots in this output array to avoid nullPointerException. Note that the +++ line helps the readability of the output of your program by separating the steps.

Please note that the output for this assignment could become large, so take caution in testing your program with reasonable size for the data set to be able to study/test your output and your algorithm.

You must ONLY use the classes that textbook offers for `Stack` and `Queue`, you may use any methods defined in these classes, and as was mentioned before, you MUST define the display methods for both of these classes solely for the purpose of this assignment.

The following are the points you must address in your report:

i. Explain your algorithm in a step-by-step format.

ii. What role did the queue play in your algorithm? Could you have fulfilled all the requirements of this assignment with three stacks? Explain your answer.

iii. Is this experimental sorting algorithm stable? Explain your answer.

iv. Does this experimental sorting algorithm sort in place? Explain your answer.

v. What is the running time of this algorithm in big O notation?

vi. Share your thoughts on the challenge of making this algorithm a stable algorithm

- *You must upload all the required files on blackboard by the due date.*
- *Programs that do not compile will be considered void.*
- *This is an individual assignment. If students submit fully or partially identical programs, ALL individuals involved will earn a grade of zero.*