COIS 2020H-Data Structures & Algorithms

Winter 2024

Assignment 3 (15 %)

Overview:

In this assignment, I assume you have enough knowledge to build simple data structures, like ArrayList and linked lists. So, you will notice that this assignment involves fewer details, allowing you to examine your programming thinking.

Requirement:

This assignment involves creating and testing a priority queue. Following are more details about each requirement.

A. Circular Array & Priority Queue

Remember that *priority queues* are special queues where each element has a priority value. As in other queues, priority queues add from the back and remove from the front, but *priority* means being inserted ahead of everything with a lower priority. Equal priority items are still first in, first out. So, keep these rules in mind when you implement the following.

1) Create your own **circular array** to support a **priority queue**. This circular array should include *Front*, *Back*, and the following methods or features:
    a. *No-argument constructor*. To initialize an empty queue of size 20.
    b. *Constructor* that takes the size as a parameter to initialize a queue of a specific size.
    c. *Enqueue*, to add a node **in the correct place**

```
public void enqueue(T input, int inputPriority){…}
```

    d. *Dequeue*, to return **and** remove a node

```
public T dequeue(){…}
```

    e. *PrintAll*. should print the data in each node along with its index in the array. You need to override the ToString() method.
    f. *Deleteall*. should delete the nodes, not the array itself

B. Node class and Main

The nodes in the priority queue represent patients' information, including *firstName*, *lastName*, *Age*, and *emergencyLevel*. It should also include the *Next* and *Previous* references.

In the main:

- Create a queue using the circular array and give it the size 10.
- In a while loop, present the following list of options to the user:
    - 1: to add a patient,
    - 2: to remove (dequeue) one patient
    - 3: print information of all patients
    - 4: Delete All patients
    - 0: to exit the loop.

- If the user wants to add a patient, your program should create a new patient object (a new node). Then call a method called "*AddPatient*" that takes a Node object and does the following:
    - Get patient's information from the user and assign them to the corresponding attributes (*firstName*, *lastName*, *Age*).
    - Give a random priority number for "*emergencyLevel"* attribute—the emergency level ranges between 1 (lowest level) and 10 (highest level, or most urgent), inclusively.
- Then, the new object (node) should be added to the queues.
- When the loop exits, the program should print the queue.

Notes:

- In the case of a circular array, Enqueue should consider whether the array is full. In this case, you can assume that the array cannot grow. Therefore, the method should print a message indicating that the queue is full and the new node will not be added.
- Dequeue should consider all cases: an empty queue, a queue with only one node, and the general case.

Submission guidelines:

Use the provided submission template where you provide a copy of your testing and code. Name it for your trentusername.docx. **Upload that, along with a copy of the same document saved as a PDF and a zip file (named for your trentusername.zip) containing your Visual Studio project directories.**

Your source code MUST be fully documented.