CSC/DCSCI 2720: Data Structures
Lab 9

Instructor: Shiraj Pokharel

Due : 48 hours after release
Late Submission deadline (with 25% penalty) 24 hours after due date

Answer the below questions.
You must submit your responses as a SINGLE Jupyter Notebook, where each program is put in separate Jupyter Notebook cells within that SINGLE Jupyter Notebook. Do **NOT** submit Colab links.
Failure to comply with this simple requirement will result in a score of Zero. Please, be careful not to be assigned a Zero score this way.

Few Rules to be followed, else will receive a score of ZERO

(1) Your submissions will work exactly as required.

(2) Your files shall not be incomplete or worse corrupted such that the file does not compile at all. Make sure you submit a file that compiles.

(3) Your submission will show an output. Should you recive a Zero for no output shown do not bother to email me with "but the logic is perfect" !

Note that your program's output must **exactly** match the specs(design , style) given here for each problem to pass the instructor's test cases .
Design refers to how well your code is written (i.e. is it clear, efficient, and elegant), while *Style* refers to the readability of your code (commented, correct indentation, good variable names).

PROBLEM STATEMENT :
In today's Lab we will explore ways to design a Queue with $O(1)$ lookup time of the Maximum element. You must implement this design using the Deque.

URL reference here:
https://docs.python.org/3/library/collections.html#collections.deque

You will solve the problem as stated below:-

Solution Hint :
Here you will Maintain two Queues - a Main Queue and a Queue holding the Maximum value(s) from the Main Queue (AKA Max Queue). The Main Queue contains the elements. The Max Queue contains the elements with Maximum value. The Max Queue would have to be a double ended Queue as you would like to be able to remove elements from both ends.

Example :
Let's say we have the following:
We add an integer 1 into our Main Queue and I hope it is really obvious that when the Main Queue contains a single element, the Max Queue can be populated without confusion :)
Main Queue: 1 << front of Queue
Max Queue : 1 << front of Queue

Now, let's say we insert a 4 into the Main Queue. the Main Queue will look as follows:
Main Queue: $4 \rightarrow 1$ << front of Queue

In the Max Queue, we don't need 1 anymore, since 1 can never be the Max of this Queue now. So we remove 1 and insert 4.
Main Queue: $4 \rightarrow 1$ << front of Queue
Max Queue : 4 << front of Queue

Say we insert 2 into the Main Queue. We know 2 is not the Max, but it can be the Max if we de-Queue 1 and 4 from the Queue. So, we insert it onto the Max Queue:
Main Queue: $2 \rightarrow 4 \rightarrow 1$ << front of Queue
Max Queue : $2 \rightarrow 4$ << front of Queue

Further, If we insert a 3 into the Main Queue, we can get rid of the 2 from the Max Queue, because 2 can no longer be the Max of the Queue, even if 4 and 1 are de-Queued. In that case our Queues become:
Main Queue: $3 \rightarrow 2 \rightarrow 4 \rightarrow 1$ << front of Queue
Max Queue : $3 \rightarrow 4$ << front of Queue

In the process of inserting 3, we removed elements from the back of the Max Queue until we found an element ≥ 3.
This is because elements < 3 could never be Max after 3 is inserted. What I stated above is exactly the algorithm for inserting an element in the Max Queue.

To lookup the Maximum Value (AKA Max), we just check the front of the Max Queue which ensures $O(1)$ lookup time.

While de-queuing elements, we check if they are equal to the front of the Max Queue,and if so, we de-Queue from the Max Queue too. For example, after

de-queuing 1, lets say we want to de-Queue 4. We see that 4 is the front of the Max Queue, so we remove both the 4s. This does indeed make sense as 4 can no longer remain the Maximum after it is removed from the Main Queue.

If the process described above is followed and you code up the example provided we end up with the complexity stated below.
Time Complexity of lookup on the Max Queue: $O(1)$
Space Complexity on the Max Queue : $O(n)$
Submissions that don't meet the mentioned Time and Space complexities will have 50% credit taken off.

Very Very Important :

(1) Your code should be well commented which explains all the steps you are performing to solve the problem. A submission without code comments will immediately be deducted 15 points !

(2) As a comment in your code, please write your test-cases on how you would test your solution assumptions and hence your code.
A submission without test cases (as comments) will immediately be deducted 15 points ! Please Remember : Although, written as comments - You will address your test cases in the form of code and not prose :)