Birla Institute of Technology and Science, Pilani Hyd Campus

BITS F232: Foundations of Data Structures and Algorithms

1st Semester 2023-24 Lab No:7 (Queue ADT)

General Instructions

- You should use STL only for the program where it is mentioned explicitly. For other programs you should solve the given task without it.

Program 1: Prog1.cpp is the implementation of a queue using Circular linked list as discussed in the class. This file is attached alongwith the lab sheet for you to carry out a sample run to get the below output.

```
1 : Enqueue
2 : Dequeue
3 : Get front element
4 : Get size
5 : Check if queue is empty
6 : Display Queue items
7 : Exit
+-------------------------+
Enter your choice: 1
Enter an item: Pilani
Pilani enqueued into the Queue.
+-------------------------+
Enter your choice: 1
Enter an item: Dubai
Dubai enqueued into the Queue.
+-------------------------+
```

```
Enter your choice: 1
Enter an item: Goa
Goa enqueued into the Queue.
+-------------------------+
Enter your choice: 1
Enter an item: Hyd
Hyd enqueued into the Queue.
+-------------------------+
Enter your choice: 6
FRONT -> Pilani Dubai Goa Hyd
+-------------------------+
Enter your choice: 2
Pilani dequeued from the Queue.
+-------------------------+
Enter your choice: 3
Elem at the FRONT = Dubai.
+-------------------------+
Enter your choice:
```

Now your task is to modify the same code (Prog1.cpp) to get the below output for the same input that you gave in the upper part, i.e. Pilani, Dubai, Goa and Hyd. (You are free to modify any function i.e. enqueue, dequeue, traverse, add, remove etc.).

Output:

```
Enter your choice: 6
FRONT -> Dubai Goa Hyd
+-------------------------+
Enter your choice:
```

Program 2: In the class we discussed the implementation of a Stack using two queues with a costly Push Operation and a cheaper Pop operation. In this program, you need to implement a Stack using two queues where the Pop operation is made costly and Push takes constant time. This program should use STL queue classes and functions to implement required Queues. You should write down missing code A and B in Prog2.cpp file given as attachment. Your output should be as shown below:

```
Stack is Empty!
15.5 pushed into the Stack.
23 pushed into the Stack.
20.5 pushed into the Stack.
TOP ->  20.5
        23
        15.5
Size of the Stack = 3
Elem at the TOP: 20.5
Popping...
Elem at the TOP: 23
TOP ->  23
        15.5
35 pushed into the Stack.
TOP ->  35
        23
        15.5
```

Program 3: This program is based on the Job scheduling concept in Operating Systems. Assume there are N jobs (numbered 1 to N) in the Idle queue (hard disk) which are to be scheduled for execution. You are given two factors: P (> 0) and Q (> 0) where, P represents the rate at which Jobs enter into the Ready Queue and Q represents the rate at which Jobs leave the Queue i.e. given to CPU for their execution. Note that both operations P and Q happen at the same time instant. The program given (Prog3.cpp) displays the jobs currently in the Ready queue for a given instant (T). Explanation for the first test run is as given below:

Test input 1:
N = 10, P = 2, Q = 1, T = 3
At time t = 0, Ready Queue = [ ], Idle queue: [1,2,…,10]
At time t = 1, Ready Queue = [2], Idle queue: [3, 4,…,10] // Job 1 and 2 enter into Ready Queue, Job 1 leaves
At time t = 2, Ready Queue = [3,4], Idle queue: [5, 6,…,10] // Job 3 and 4 enter into Ready Queue, Job 2 leaves
At time t = 3, Ready Queue = [4,5,6], Idle queue: [7, 8,…,10] // Job 5 and 6 enter into Ready Queue, Job 3 leaves

Answer: [4,5,6]

Output:

Test run1

```
Enter N (> 0): 10
Enter P (> 0): 2
Enter Q (> 0): 1
Enter T (> 0): 3
[4, 5, 6]
```

Test run2

```
Enter N (> 0): 10
Enter P (> 0): 3
Enter Q (> 0): 2
Enter T (> 0): 3
[7, 8, 9]
```

Test run3

```
Enter N (> 0): 10
Enter P (> 0): 3
Enter Q (> 0): 2
Enter T (> 0): 5
[]
```

Your task in this program is to complete writing the missing code A (multiple lines), missing code B (expressions within the for ()), missing code C (one line) (in Prog3.cpp) and provide an explanation similar to the one given above for Test run 2 and Test run 3 describing how 7,8,9 and [] were output for those.

Program 4: You may go through the attached Prog4a.cpp file and run to understand the application of Deque to simulate a Stack behaviour. This is an example of Adapter design pattern, where you use a Deque (already implemented using a doubly linked list) to implement Push and Pop functionalities of a Stack ADT.

Using the above understanding, now you need to implement a Queue using a Deque (using STL classes). Write down the missing codes (A, B, and C) with appropriate error handling/ exceptions in Prog4b.cpp to get the below output:

Output:

```
Queue is Empty!
Pilani enqueued into the Queue.
Goa enqueued into the Queue.
Hyd enqueued into the Queue.
FRONT -> Pilani Goa Hyd
Size of the Queue = 3.
Elem at the FRONT: Pilani
Dequeuing...
Elem at the FRONT: Goa
FRONT -> Goa Hyd
Dubai enqueued into the Queue.
FRONT -> Goa Hyd Dubai
```
