National University of Computer & Emerging Sciences (NUCES) Islamabad,

Department of Software Engineering

DATA STRUCTURES – FALL 2023

LAB 08

Queues

Learning Outcomes

In this lab you are expected to learn the following:

- Queues
- Queues Applications

TASK 1: (30 minutes)

- Implement Class Node having employee data.
- Implement Class Queue, its operation are listed below.
    a. **Queue ()**
       A non-parameterized constructor that creates an empty queue. Where should the front and rear of an empty queue point to?
    b. **enqueue ()**
       Inserts the element at the rear of queue.
    c. **dequeue ()**
       Removes the element from the front of queue.
    d. **showFront()**
       Returns the value of the element at front of the queue.
    e. **isEmpty ()**
       Returns True if queue is empty else returns False.

Task 2: (30 minutes)

Password Encryption via Queue

Encryption scrambles your password so it's unreadable and/or unusable by hackers. That simple step protects your password while it's sitting in a server, and it offers more protection as your password zooms across the internet.

Imagine that you've created the strongest password possible. Now, imagine that all of your hard work is stored in plain text on your company's server. If a hacker gets inside, what happens next? All of your efforts go to waste, and your username and password are sold on the open market to the highest bidder.

In cryptography, encryption is the process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge. The result of the process is encrypted information. In many contexts, the word encryption also implicitly refers to the reverse process, decryption to make the encrypted information readable again

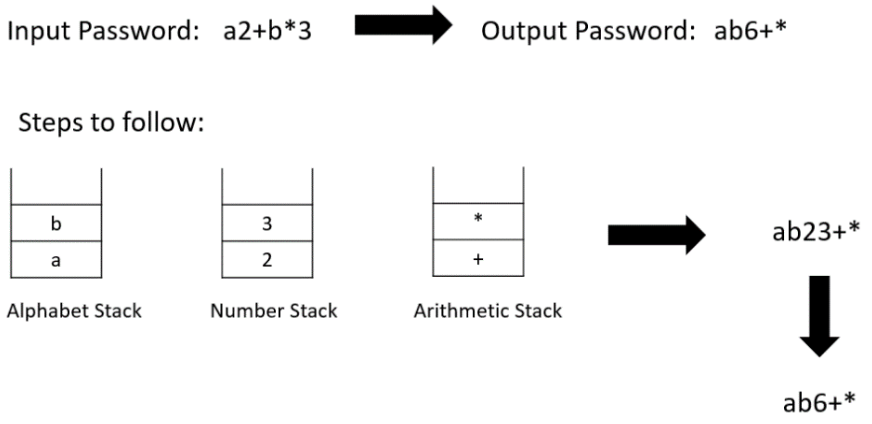
If your password is stored as **a2+b*3** then it is of no use to the hacker because this is encrypted. The actual password is **ab6+*** (decrypted)

How it works?

The answer is simple, you can decrypt it via queue operations easily!

1. Step 1: You start reading from the left most character (do this character by character)
    a. If it is an **alphabet** put it in **aplhabet queue ->** a,b
    b. If it is a **number** put it in **number queue ->** 2,3
    c. If it is an **operator** put it in **arithmatic queue ->** +,*
2. Step 2: From **arithmatic queue,** find the operator with highest precedence and then apply that operation to all the numbers present in **number queue ->** 2*3=6

3. Step 3: Final decrypted password is concatenation of **alphabet queue** + **number queue** + **arithmetic queue**->ab6+*



Your task is to take an encrypted password from the user in the form of a string. The password must only contains Alphanumeric characters and Arithmetic operations +, -, *, / (No brackets should be included).

Your decrypted password should be in the following format:

1. The alphabets must comes first.
2. Then comes the numbers.
3. At the end, there are arithmetic characters.

Note: If the input password contains other than alphanumeric characters and arithmetic operators, ignore them while processing.

Task 3: (30 minutes)

In this problem you have to implement a **stack** using **multiple queues**.

You know the concept of stacks and queues. Therefore, you have to follow all the rules of queues to implement stack.

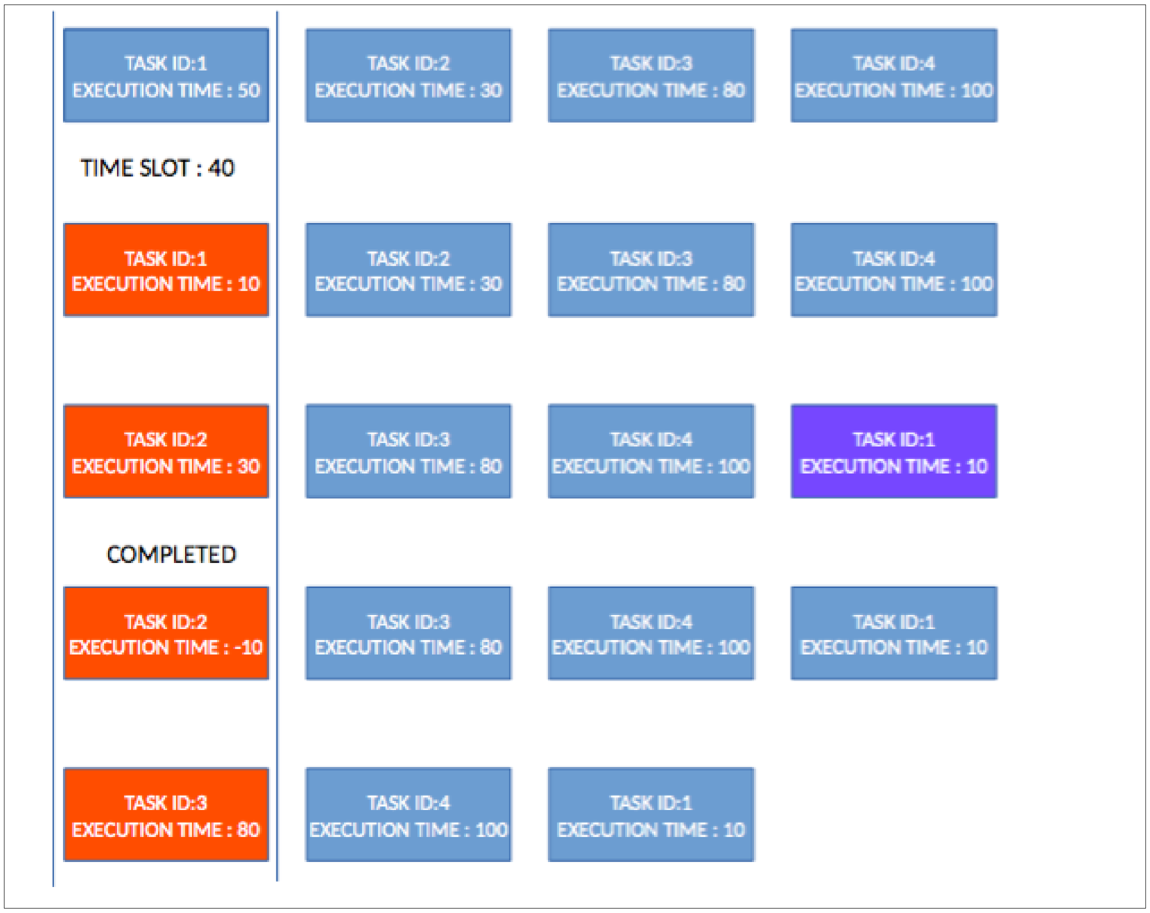
For example: if your input 1,2,3,4 respectively, your output should be 4,3,2,1. Now there are multiple ways of doing it, but in this problem, you have to use multiple queues (use 2 different queues). By using multiple queues, you should create such logic that gives you an output of 4,3,2,1 which is just like output of stack.

Note: You are not allowed to use stack. You have to implement the functionality of stack using queues.

TASK 4: (50 minutes)

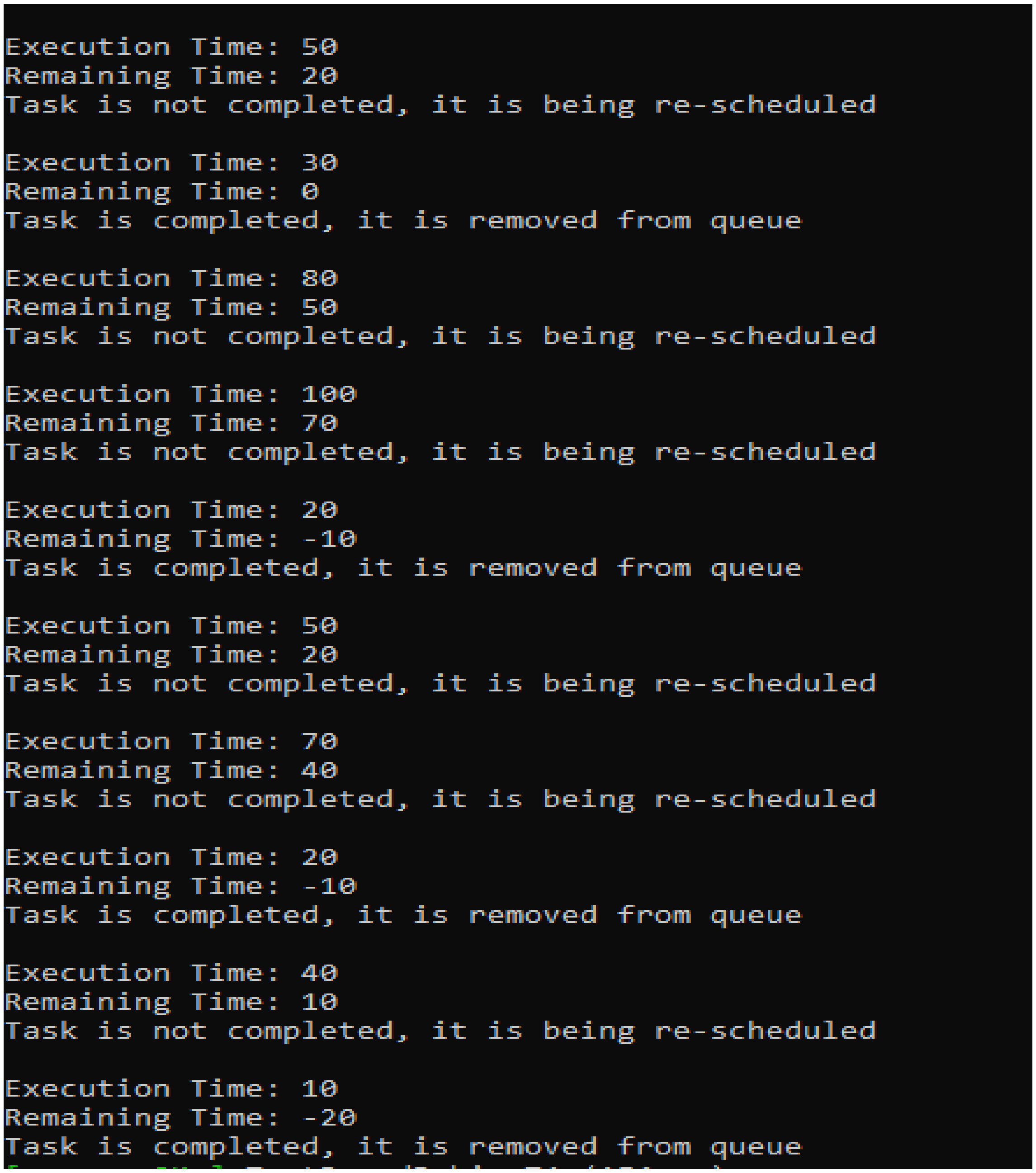
Round robin is a scheduling algorithm that an operating system uses to time share computational resources of a processor between tasks. Each task is given a specific time slot **(Quantum)** to execute on a processor (CPU Time), once this time slot expires and the task is not yet completed, it is preempted **(dequeue)** and added to the back of the queue **(enqueue)** with its **Remaining**

Execution Time. Then the next task **(front)** in the queue is selected and this processes continues until all tasks have finished execution.



Your task is to simulate this process using a queue. Implement a function **roundRobin()** that takes input a **Queue of execution time and a time quantum**. Then it simulates the process of task execution until the queue gets empty.

Example of output is displayed below



```
Execution Time: 50
Remaining Time: 20
Task is not completed, it is being re-scheduled

Execution Time: 30
Remaining Time: 0
Task is completed, it is removed from queue

Execution Time: 80
Remaining Time: 50
Task is not completed, it is being re-scheduled

Execution Time: 100
Remaining Time: 70
Task is not completed, it is being re-scheduled

Execution Time: 20
Remaining Time: -10
Task is completed, it is removed from queue

Execution Time: 50
Remaining Time: 20
Task is not completed, it is being re-scheduled

Execution Time: 70
Remaining Time: 40
Task is not completed, it is being re-scheduled

Execution Time: 20
Remaining Time: -10
Task is completed, it is removed from queue

Execution Time: 40
Remaining Time: 10
Task is not completed, it is being re-scheduled

Execution Time: 10
Remaining Time: -20
Task is completed, it is removed from queue
```

Note: Create a proper menu in Main so that the user can perform different functionalities. You can use the Main Class from previous lab and make appropriate changes to it.