Lab Assignment 1: CS2233

August 22, 2024

Each question is of 5 marks.

Questions:

1. Write a C code that takes a postfix expression as input and outputs the corresponding result using the stack data structure. For example: if the input is ``23+'' the output should be ``5''.

   Input format

   - First line will contain postfix string.
   - Postfix string will contain following characters

     $$[0-9] \cup [(,)] \cup [+,-,*,/,\wedge]$$

   - **Note:** Order of precedence of operator is

     $$\{\wedge\} > \{/,*\} > \{+,-\}$$

   Output format

   - Your output is result for corresponding postfix statement.

   Example:
   Test Case 1: **Input:**

   ```
   561+*2-
   ```

   Output:

   ```
   33
   ```

   Test Case 2: **Input:**

   ```
   83+72-*
   ```

   Output:

```
55
```

Test Case 3: **Input:**

```
92^3*7+
```

Output:

```
250
```

Note: You can assume that the user inputs the postfix expression in the correct format.

2. Write a C program that takes a queue (over integers) and an integer k as input, and reverse the first k element of the queue. You can take an array implementation of a queue that supports `enqueue` and `dequeue` operation.

   For example: if the input queue is $[10, 20, 30, 40, 50]$ with 10 and 50 as front and rear of the queue, and $k = 3$, then the output queue should be $[30, 20, 10, 40, 50]$.

   Input Format:

   - The first line contains an integer n denoting the number of elements in the queue.
   - The second line contains n integers representing the elements of the queue in the order from front to rear.
   - The third line contains an integer k, the number of elements from the front of the queue to reverse.

   Output Format:

   - Print the queue after reversing the first k elements.

   Example:

   Test Case 1

   Input:

   ```
   5
   10 20 30 40 50
   3
   ```

   Output:

   ```
   30 20 10 40 50
   ```

   Test Case 2

   Input:

```
5
5 15 25 35 45
5
```

Output:

```
45 35 25 15 5
```

3. Write a *C* program to implement a queue using two stacks; that is, you are supposed to use the functionality of `Push` and `Pop`, and mimic the functionality of `enqueue` and `dequeue` operation. Analyze the running time of the queue operations

Input format:

1-Enqueue: Add an element to the queue.
2-Dequeue: Remove and return the front element from the queue.
3-Exit: End the program.

Test Case 1:
Input:

```
1 10   // Enqueue 10
1 20   // Enqueue 20
2      // Dequeue
1 30   // Enqueue 30
2      // Dequeue
2      // Dequeue
3      // Exit
```

Output:

```
Dequeued: 10
Dequeued: 20
Dequeued: 30
```

Test Case 2:
Input:

```
2     // Dequeue
3     // Exit
```

Output:

```
Queue is empty
```