Lab Practice Sheet 4: Queues & Deques

CS G526: Advanced Algorithms & Complexity

Instructor: Tulasimohan Molli **Date:** 2026-09-26

! Evaluation Instructions for Students

- **Practice all 8 problems below.**
- In the next lab session, **one question will be chosen at random** for on-the-spot coding and evaluation.
- Your code will be evaluated by running it against automated hidden test cases.
- You may write your solutions in **Python**, **C++**, or **Java**.
- **Library Usage:** You are allowed and encouraged to use built-in standard libraries for data structures (e.g., `queue`, `deque` in C++, `collections.deque`, `queue.Queue` in Python, `ArrayDeque`, `LinkedList` in Java).
- Standard I/O (reading from standard input `stdin` and printing to standard output `stdout`) is required.

Q1: Implement Queue using Stacks

Description: Implement a First-In-First-Out (FIFO) queue using only two stacks. The implemented queue must support all normal queue operations: `push`, `pop`, `peek`, and `empty`. Each operation must run in amortized $O(1)$ time.

- **Input Format:**
  - Line 1: An integer Q, the number of operations.
  - Next Q lines: Operation descriptions:
    - `push X`: Push integer X to the back of the queue.
    - `pop`: Remove and print the element from the front of the queue. If empty, print `-1`.
    - `peek`: Print the front element without removing it. If empty, print `-1`.
    - `empty`: Print `true` if the queue is empty, `false` otherwise.
- **Output Format:**
  - Print the result of each `pop`, `peek`, and `empty` operation on a new line.

i Sample Input

```
7
push 10
push 20
peek
pop
push 30
pop
empty
```

Sample Output

```
10
10
20
false
```

Q2: First Non-Repeating Character in a Stream

Description: Given a stream of lowercase English letters, for each incoming character determine the earliest arriving character in the stream that has appeared exactly once so far. If all characters received so far have appeared at least twice, output `-1`.

Solve this in $O(N)$ total time using a FIFO Queue and a frequency array.

- **Input Format:**
  - A single string S consisting of lowercase English letters.
- **Output Format:**
  - Space-separated characters (or `-1`) corresponding to each step in the stream.

Sample Input 1

```
racecar
```

Sample Output 1

```
r r r r r r e
```

Sample Input 2

```
aabbcdefgh
```

Sample Output 2

```
a -1 b -1 c c c c c c
```

Q3: Sliding Window Maximum (Monotonic Deque)

Description: Given an array of N integers and an integer K, find the maximum element in every sliding window of size K as the window moves from left to right across the array.

Your algorithm must run in strictly $O(N)$ time using a monotonic double-ended queue (`deque`). Solutions running in $O(N \cdot K)$ will not be accepted.

- **Input Format:**
  - Line 1: Two space-separated integers N and K ($1 \leq K \leq N$).
  - Line 2: N space-separated integers representing the array.
- **Output Format:**
  - A single line containing $N - K + 1$ space-separated integers representing the maximum of each sliding window.

Sample Input 1

```
8 3
1 3 -1 -3 5 3 6 7
```

Sample Output 1

```
3 3 5 5 6 7
```

Sample Input 2

```
6 2
4 2 5 6 1 3
```

Sample Output 2

```
4 5 6 6 3
```

Q4: Circular Tour / Gas Station Circuit

Description: There are N gas stations arranged in a circle. Each station i has an amount of gas `gas[i]`, and the cost to travel from station i to the next station $(i+1)$ mod N is `cost[i]` units of gas. You start with an empty fuel tank.

Find the starting gas station index (0-based) from which you can travel around the entire circuit once clockwise without running out of fuel. If no such starting station exists, return `-1`. If multiple exist, return the smallest starting index.

- **Input Format:**
  - Line 1: An integer N, the number of gas stations.
  - Line 2: N space-separated integers representing `gas[i]`.
  - Line 3: N space-separated integers representing `cost[i]`.
- **Output Format:**
  - A single integer representing the 0-based starting gas station index, or `-1`.

Sample Input

```
5
1 2 3 4 5
3 4 5 1 2
```

Sample Output

```
3
```

Explanation: Starting at station 3 (gas=4, cost=1): fuel becomes $4-1+5-2+1-3+2-4+3-5=0 \geq 0$, completing the loop.

Q5: Interleave Queue Elements using One Stack

Description: Given a queue of N integers (where N is guaranteed to be even), rearrange the elements by interleaving the first half of the queue with the second half.

You must read input into a Queue and are strictly allowed to use **at most one Stack** as auxiliary storage. No additional arrays or lists are allowed.

- **Input Format:**
  - Line 1: An even integer N.
  - Line 2: N space-separated integers representing the queue elements.
- **Output Format:**
  - A single line containing the N interleaved integers separated by a space.

Sample Input 1

```
4
11 12 13 14
```

Sample Output 1

```
11 13 12 14
```

Sample Input 2

```
8
1 2 3 4 5 6 7 8
```

Sample Output 2

```
1 5 2 6 3 7 4 8
```

Q6: Time Needed to Buy Tickets

Description: There are N people in a line queuing to buy tickets, where person 0 is at the front and person $N-1$ is at the back. Person i wants to buy `tickets[i]` tickets. Each ticket transaction takes exactly 1 second. After buying 1 ticket, if a person still needs more tickets, they immediately go to the back of the queue.

Calculate the total number of seconds required for the person at index K (0-based) to finish buying all their tickets.

- **Input Format:**
  - Line 1: Two integers N and K ($0 \leq K < N$).
  - Line 2: N space-separated integers representing `tickets[i]`.
- **Output Format:**
  - A single integer representing the total time taken in seconds.

Sample Input 1

```
3 2
2 3 2
```

Sample Output 1

```
6
```

Sample Input 2

```
4 0
5 1 1 1
```

Sample Output 2

```
8
```

Q7: Reverse First K Elements of a Queue

Description: Given a queue containing N integers and an integer K, reverse the order of the first K elements of the queue, while keeping the relative order of the remaining $N - K$ elements unchanged.

Solve this using a Queue and an auxiliary Stack.

- **Input Format:**
  - Line 1: Two space-separated integers N and K ($1 \leq K \leq N$).
  - Line 2: N space-separated integers representing the queue elements.
- **Output Format:**
  - A single line containing N space-separated integers representing the modified queue.

Sample Input

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

Sample Output

```
30 20 10 40 50
```

Q8: Binary Numbers Generation from 1 to N

Description: Given an integer N, generate and print the binary representations of all numbers from 1 to N in increasing numerical order.

To ensure $O(N)$ optimal performance, you must use a Queue of binary strings (pushing `str + "0"` and `str + "1"` sequentially) rather than repeatedly invoking arithmetic radix conversions.

- **Input Format:**
  - Line 1: An integer N.
- **Output Format:**
  - A single line containing N space-separated binary strings.

Sample Input 1

```
5
```

Sample Output 1

```
1 10 11 100 101
```

Sample Input 2

```
7
```

Sample Output 2

```
1 10 11 100 101 110 111
```