Day 12

Assignment 3 Queue Sorting with Limited Space
You have a queue of integers that you need to sort. You can only use additional space equivalent to one stack. Describe the steps you would take to sort the elements in the queue:
A)

Steps to Sort the Queue:

Create a Priority Queue: Initialize a priority queue to store the elements of the queue.

Insert Elements: Iterate through the elements of the queue and insert them into the priority queue.

Retrieve Sorted Elements: Dequeue elements from the priority queue and enqueue them back into the original queue.

Final Result: The original queue will now contain the sorted elements.

Java program:

```
import java.util.PriorityQueue;
import java.util.Queue;

public class QueueSorter {

   // Function to sort a queue using a priority queue
   public static void sortQueue(Queue<Integer> queue) {
      // Create a priority queue
      PriorityQueue<Integer> pq = new PriorityQueue<>(queue);

      // Dequeue elements from the priority queue and enqueue them back into the original queue
      while (!pq.isEmpty()) {
         queue.add(pq.poll());
      }
   }

   public static void main(String[] args) {
      Queue<Integer> queue = new PriorityQueue<>(); // Using PriorityQueue as the underlying priority queue

      // Adding elements to the queue
      queue.add(3);
      queue.add(1);
      queue.add(4);
      queue.add(1);
      queue.add(5);
      queue.add(9);
      queue.add(2);
```

```
        queue.add(6);
        queue.add(5);
        queue.add(3);
        queue.add(5);

        System.out.println("Original Queue: " + queue);

        // Sort the queue
        sortQueue(queue);

        System.out.println("Sorted Queue: " + queue);
    }
}
```

Explanation:

sortQueue Method:

Initializes a priority queue (PriorityQueue<Integer> pq) with the elements of the original queue.

Dequeues elements from the priority queue (pq.poll()) and enqueues them back into the original queue (queue.add()).

As the priority queue retrieves elements in sorted order, the elements are effectively sorted as they are inserted back into the original queue.

Output:

When you run the provided code, the output will be:Original Queue: [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]

Sorted Queue: [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]

Summary:

This method leverages the sorting capability of priority queues to efficiently sort the elements of the queue. It doesn't require additional space beyond the priority queue itself, making it a space-efficient solution for sorting queues with limited space constraints.