Assignment Questions 24

Question 1. Roman to Integer.

code:-

```
class Solution {
 public int romanToInt(String S) {
    int ans = 0, num = 0;
    for (int i = S.length()-1; i >= 0; i--) {
       switch(S.charAt(i)) {
          case 'I': num = 1; break;
          case 'V': num = 5; break;
          case 'X': num = 10; break;
          case 'L': num = 50; break;
          case 'C': num = 100; break;
          case 'D': num = 500; break;
          case 'M': num = 1000; break;
       }
       if (4 * num < ans) ans -= num;
       else ans += num;
    }
    return ans;
 }
}
```

Question 2. Longest Substring Without Repeating Characters

code:-

```
import java.io.*;
import java.util.*;

class GFG{
public static Boolean areDistinct(String str,int i, int j)
{
	boolean[] visited = new boolean[26];

	for(int k = i; k <= j; k++)
	{
		if (visited[str.charAt(k) - 'a'] == true)
			return false;

		visited[str.charAt(k) - 'a'] = true;
	}
	return true;
}
public static int longestUniqueSubsttr(String str)
{
	int n = str.length();
	int res = 0;
	for(int i = 0; i < n; i++)
		for(int j = i; j < n; j++)
			if (areDistinct(str, i, j))
```

```
                                        res = Math.max(res, j - i + 1);


        return res;
}
public static void main(String[] args)
{
        String str = "geeksforgeeks";
        System.out.println("The input string is " + str);
        int len = longestUniqueSubsttr(str);
        System.out.println("The length of the longest " +"non-repeating character " +"substring is " +
len);
}
}
```

Question 3. Majority Element

code:-

```
class Solution {
    public int majorityElement(int[] nums) {
        int count = 0, maxElement = 0;
                                        for(int num: nums) {
                                                if(count == 0) {
                                                        maxElement = num;
                                                }
                                                if(num == maxElement) {
                                                        count++;
                                                } else {
```

```
                                        count--;
                                    }
                                }

                                return maxElement;
    }
}
```

Question 4. Group Anagram

code:-

```
 class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> map = new HashMap<>();

        for (String word : strs) {
            char[] chars = word.toCharArray();
            Arrays.sort(chars);
            String sortedWord = new String(chars);

            if (!map.containsKey(sortedWord)) {
                map.put(sortedWord, new ArrayList<>());
            }

            map.get(sortedWord).add(word);
        }
```

```
        return new ArrayList<>(map.values());
    }
}
```

Question 5. Ugly Numbers

code:-

```
class Solution {
    public int nthUglyNumber(int n) {
        int c2 = 0,c3 = 0,c5 = 0;
        int[] dp = new int[n+1];
        dp[0] = 1;
        for(int i=1;i<=n;i++)
        {
            dp[i] = Math.min(2*dp[c2],Math.min(3*dp[c3],dp[c5]*5));
            if(dp[i] == 2*dp[c2])
                c2++;
            if(dp[i] == 3*dp[c3])
                c3++;
            if(dp[i] == 5*dp[c5])
                c5++;
        }
        return dp[n-1];
    }
}
```

Question 6. Top K Frequent Words

code:-

```
class Solution {
  public List<String> topKFrequent(String[] words, int k) {

      // map hold the word: counts
      HashMap<String, Integer> map = new HashMap();

      // sort the map by frequency high->low order, sort words lexi order
      PriorityQueue<Map.Entry<String, Integer>> heap = new PriorityQueue<>(
         (a,b)->{
            if(a.getValue() != b.getValue())
               return a.getValue().compareTo(b.getValue());
            return -a.getKey().compareTo(b.getKey());
         }
      );

      // fill the map
      for(String word: words){
         map.merge(word, 1, Integer::sum);
      }

      // put into heap
      for(Map.Entry<String, Integer> entry: map.entrySet()){
         heap.offer(entry);
         if(heap.size() > k)
            heap.poll();
      }
```

```
        // pop out the answer
        List<String> ans = new ArrayList();
        while(heap.size() > 0)
            ans.add(heap.poll().getKey());

        // check the order
        Collections.reverse(ans);
        return ans;
    }
}
```

Question 7. Sliding Window Maximum

code:-

```
class Solution {
    public class Pair implements Comparable<Pair>{
        int num;
        int index;
        public Pair(int num, int index){
            this.num=num;
            this.index=index;
        }
        @Override
        public int compareTo(Pair p2){
            return p2.num-this.num; // Sort in descending Order
        }
    }
    public int[] maxSlidingWindow(int[] nums, int k) {
```

```
        PriorityQueue<Pair> pq= new PriorityQueue<>();
        int[] res=new int[nums.length-k+1];
        for(int i=0; i<k; i++){
            pq.add(new Pair(nums[i],i));
        }
        res[0]=pq.peek().num;
        for(int i=k; i<nums.length ;i++){
            while(pq.size()>0 && pq.peek().index<= (i-k)){
                pq.remove();
            }
            pq.add(new Pair(nums[i],i));
            res[i-k+1]=pq.peek().num;
        }
        return res;
    }
}
```

Question 8. Find K Closest Elements

code:-

```
class Solution {
    public List<Integer> findClosestElements(int[] arr, int k, int x) {

        int start = 0;
        int end = arr.length - 1;
        while (end - start >= k) {
            if (Math.abs(arr[start] - x) > Math.abs(arr[end] - x)) {
```

```
            start++;
        } else {
            end--;
        }
    }

    List<Integer> result = new ArrayList<>(k);
    for (int i = start; i <= end; i++) {
        result.add(arr[i]);
    }
    return result;
  }
}
```