

# CODING QUESTION PRACTICE – 6

## 1. Bubble Sort

The screenshot shows a coding practice interface for the Bubble Sort problem. The left panel displays the 'Output Window' with 'Compilation Results' indicating 'Problem Solved Successfully'. It shows 'Test Cases Passed: 1115 / 1115', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 2 / 2', and 'Time Taken: 0.63'. The right panel shows the Java code for the bubble sort algorithm.

```
1 // Driver Code Ends
2 // User function Template for Java
3
4 class Solution {
5     // Function to sort the array using bubble sort algorithm.
6     public static void bubbleSort(int arr[]) {
7         // code here
8         int n = arr.length;
9         int temp = 0;
10        for(int i=0; i < n; i++){
11            for(int j=1; j < (n-i); j++){
12                if(arr[j-1] > arr[j]){
13                    temp = arr[j-1];
14                    arr[j-1] = arr[j];
15                    arr[j] = temp;
16                }
17            }
18        }
19    }
20 }
21 // Driver Code Ends
```

Time Complexity:  $O(n^2)$

Space Complexity:  $O(1)$

## 2. Quick Sort

The screenshot shows a coding practice interface for the Quick Sort problem. The left panel displays the 'Output Window' with 'Compilation Results' indicating 'Problem Solved Successfully'. It shows 'Test Cases Passed: 1120 / 1120', 'Attempts: Correct / Total: 1 / 2', 'Accuracy: 50%', 'Points Scored: 4 / 4', and 'Time Taken: 0.68'. The right panel shows the Java code for the quick sort algorithm.

```
1 class Solution
2 {
3     //Function to sort an array using quick sort algorithm.
4     static void quickSort(int arr[], int si, int ei)
5     {
6         // code here
7         if(si==ei){
8             return;
9         }
10        int pivotIdx=partition(arr,si,ei);
11        quickSort(arr, si, pivotIdx-1);
12        quickSort(arr, pivotIdx+1,ei);
13    }
14    static int partition(int arr[], int si, int ei)
15    {
16        // your code here
17        int pivot=arr[ei];
18        int i=si-1;
19        for(int j=si;j<ei;j++){
20            if(arr[j]<pivot){
21                i++;
22                int temp=arr[j];
23                arr[j]=arr[i];
24                arr[i]=temp;
25            }
26        }
27        i++;
28        int temp=pivot;
29        arr[ei]=arr[i];
30        arr[i]=temp;
31    }
32 }
```

Time Complexity:  $O(n \log n)$  average case

Space Complexity:  $O(\log n)$

### 3.Non Repeating Character

The screenshot shows a coding platform interface with a green header bar containing navigation links: Courses, Tutorials, Jobs, Practice, and Contests. The main area is divided into two panels. The left panel, titled 'Output Window', displays 'Compilation Results' for a problem solved successfully. It shows 'Test Cases Passed: 1130 / 1130', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 2 / 2', and 'Your Total Score: 76'. Below this, it suggests solving the next problem, 'Reverse Words'. The right panel shows the code editor for a Java solution. The code defines a class 'Solution' with a static method 'nonRepeatingChar' that uses a 'HashMap' to find the first non-repeating character in a string. The code is as follows:

```
1 // } Driver Code Ends
2
3 // User function Template for Java
4
5 class Solution {
6     // Function to find the first non-repeating character in a string.
7     static char nonRepeatingChar(String s) {
8         // Your code here
9
10        HashMap<Character, Integer> map = new HashMap<>();
11        for(char ch : s.toCharArray())
12            map.put(ch, map.getOrDefault(ch,0)+1);
13        for(char ch : s.toCharArray())
14        {
15            if(map.get(ch)==1)
16                return ch;
17        }
18        return '$';
19    }
20 }
```

Time Complexity:  $O(n)$

Space Complexity:  $O(1)$

### 4.k largest elements

The screenshot shows a coding platform interface with a green header bar containing navigation links: Courses, Tutorials, Jobs, Practice, and Contests. The main area is divided into two panels. The left panel, titled 'Output Window', displays 'Compilation Results' for a problem solved successfully. It shows 'Test Cases Passed: 1111 / 1111', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 4 / 4', and 'Your Total Score: 80'. Below this, it suggests solving the next problem, 'Merge k Sorted Arrays'. The right panel shows the code editor for a Java solution. The code defines a class 'Solution' with a static method 'kLargest' that finds the k largest elements in an array. The code is as follows:

```
1 // } Driver Code Ends
2
3 class Solution {
4     // Function to find the first negative integer in every window of size k
5     static List<Integer> kLargest(int arr[], int k) {
6         // write code here
7
8         Arrays.sort(arr);
9         List<Integer> r = new ArrayList<>();
10        for (int i = arr.length - 1; i >= arr.length - k && i >= 0; i--) {
11            r.add(arr[i]);
12        }
13        return r;
14    }
15 }
```

Time Complexity:  $O(n \log n)$

Space Complexity:  $O(k)$

## 5. Form the Largest Number

The screenshot displays a coding platform interface with the following components:

- Navigation Bar:** Includes links for Courses, Tutorials, Jobs, Practice, and Contests.
- Problem Header:** Shows the problem title 'Form the Largest Number' and the user 'Y.O.G.I. (AI Bot)'.
- Compilation Results:** A green banner indicates 'Problem Solved Successfully'.
- Test Cases Passed:** 1111 / 1111.
- Attempts:** 1 / 1.
- Accuracy:** 100%.
- Points Scored:** 4 / 4.
- Time Taken:** 1.27.
- Solve Next:** A list of recommended problems: 'Max sum in the configuration', 'Maximum Index', and 'Maximum Index'.
- Code Editor:** Displays the Java solution code for the problem.

```
1 // Driver Code Ends
28
29
30 // User function Template for Java
31
32 class Solution {
33     String printLargest(int[] arr) {
34         String[] sArr = new String[arr.length];
35         for (int i = 0; i < arr.length; i++) {
36             sArr[i] = String.valueOf(arr[i]);
37         }
38         Arrays.sort(sArr, (a, b) -> {
39             String order1 = a + b;
40             String order2 = b + a;
41             return order2.compareTo(order1);
42         });
43         if (sArr[0].equals("0")) {
44             return "0";
45         }
46         StringBuilder result = new StringBuilder();
47         for (String s : sArr) {
48             result.append(s);
49         }
50         return result.toString();
51     }
52 }
53 }
```

**Time Complexity:**  $O(n \log n)$

**Space Complexity:**  $O(n)$