

Non Repeating Character

Code:

```
class Solution {  
  
    // Function to find the first non-repeating character in a string.  
    static char nonRepeatingChar(String s) {  
  
        HashMap<Character, Integer> map = new HashMap<>();  
  
        for(char ch : s.toCharArray())  
            map.put(ch, map.getOrDefault(ch,0)+1);  
  
        for(char ch : s.toCharArray())  
        {  
            if(map.get(ch)==1)  
                return ch;  
        }  
        return '$';  
    }  
}
```

output:

The screenshot displays a coding platform interface with a dark theme. On the left, the 'Output Window' shows 'Compilation Results' for a problem named 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. Below this, statistics are shown: 'Test Cases Passed: 1130 / 1130', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 2 / 2', and 'Time Taken: 0.54'. At the bottom, there are buttons for 'Solve Next' and 'Reverse Words', along with a link to 'Longest substring with distinct characters'. The right side of the interface shows the code editor with the Java solution for the 'Non Repeating Character' problem. The code is written in Java 1.8 and includes a class 'Solution' with a static method 'nonRepeatingChar'. The method uses a 'HashMap' to count the frequency of each character in the input string and returns the first character with a frequency of 1. If no such character is found, it returns '\$'. The code is highlighted with syntax coloring, and the editor includes a 'Start Timer' button and a 'Submit' button at the bottom right.

k largest elements

output:

Class Solution {

// Function to find the first negative integer in every window of size k

static List<Integer> kLargest(int arr[], int k) {

Arrays.sort(arr);

int n = arr.length;

int[] brr = new int[k];

int i = 0;

while (k != 0) {

brr[i] = arr[n - 1];

n--;

i++;

k--;

}

List<Integer> result = new ArrayList<>();

for (int num : brr) {

result.add(num);

}

return result;}

The screenshot displays a coding platform interface. On the left, the 'Output Window' shows 'Problem Solved Successfully' with a green checkmark. Below this, it lists 'Test Cases Passed: 1111 / 1111', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 4 / 4', and 'Your Total Score: 6'. At the bottom, it suggests 'Solve Next' problems: 'Merge k Sorted Arrays', 'Maximum Sum Combination', and 'Optimal Array'. On the right, the code editor shows the Java code for the 'k largest elements' problem, which matches the code provided in the text above. The code is in Java 1.8 and includes a 'Start Timer' button. The code is as follows:

```
1. class Solution {
2.
3. // Function to find the first negative integer in every window of size k
4. static List<Integer> kLargest(int arr[], int k) {
5.
6. Arrays.sort(arr);
7.
8. int n = arr.length;
9.
10. int[] brr = new int[k];
11.
12.
13. int i = 0;
14.
15. while (k != 0) {
16.
17.     brr[i] = arr[n - 1];
18.
19.     n--;
20.
21.     i++;
22.
23.     k--;
24.
25. }
26.
27. List<Integer> result = new ArrayList<>();
28.
29. for (int num : brr) {
30.
31.     result.add(num);
32.
33. }
34.
35. return result;
36. }
```

Bubble Sort

Code:

```
class Solution {  
    // Function to sort the array using bubble sort algorithm.  
    public static void bubbleSort(int arr[]) {  
  
        for(int i=0;i<arr.length;i++){  
            for(int j=0;j<arr.length-i-1;j++){  
  
                if(arr[j]>arr[j+1]){  
  
                    int temp= arr[j];  
                    arr[j]=arr[j+1];  
                    arr[j+1]=temp;  
  
                }  
  
            }  
        }  
    }  
}
```

Output:

The screenshot displays a code editor interface with a dark theme. On the left, the 'Output Window' is open, showing 'Compilation Results' for a problem titled 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. The results show 'Test Cases Passed: 1115 / 1115', 'Attempts: Correct / Total: 2 / 2', and 'Accuracy: 100%'. The 'Time Taken' is listed as '0.64'. A note at the bottom states: 'You get marks only for the first correct submission if you solve the problem without viewing the full solution.' On the right, the code editor shows the Java code for the Bubble Sort algorithm, which matches the code provided in the 'Code' section. The code is for a class 'Solution' with a static method 'bubbleSort'. The editor also shows a 'Start Timer' button and an 'Average Time: 15m' indicator. At the bottom right, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

QUITE SORT

Code:

```
class Solution {  
    // Function to sort an array using quick sort algorithm.  
    static void quickSort(int arr[], int low, int high) {  
        if(low<high){  
            int pi = partition(arr,low,high);  
            quickSort(arr,low,pi-1);  
            quickSort(arr,pi+1,high);  
        }  
    }  
  
    static int partition(int arr[], int low, int high) {  
        int pivot = arr[high];  
        int i = low - 1;  
        for (int j = low; j <= high - 1; j++) {  
            if (arr[j] < pivot) {  
                i++;  
                swap(arr, i, j);  
            }  
        }  
        swap(arr, i + 1, high);  
        return i + 1;  
    }  
  
    static void swap(int[] arr, int i, int j) {  
        int temp = arr[i];  
        arr[i] = arr[j];  
        arr[j] = temp;  
    }  
}
```

```

}

}

```

Output:

The screenshot shows a coding platform interface. On the left, the 'Output Window' displays 'Problem Solved Successfully' with a green checkmark. Below this, it shows 'Test Cases Passed: 1120 / 1120', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 0 / 4', and 'Time Taken: 0.64'. A message at the bottom states: 'No marks, because you solved this problem after visiting Full solution on 2024-11-18.' On the right, the code editor shows a Java solution for the Quick Sort algorithm, including a recursive function and a partition function.

Edit Distance

CODE:

```

class Solution {
    public int editDistance(String s1, String s2) {
        // Code here
        int m=s1.length();
        int n=s2.length();
        int[][] dp=new int[m+1][n+1];

        for(int i=0;i<=m;i++)
            dp[i][0]=i;

        for(int j=0;j<=n;j++)
            dp[0][j]=j;

        for(int i=1;i<=m;i++){
            for(int j=1;j<=n;j++){
                if(s1.charAt(i-1)==s2.charAt(j-1))
                    dp[i][j]=dp[i-1][j-1];

                else
                    dp[i][j]=Math.min(dp[i-1][j],Math.min(dp[i][j-1],dp[i-1][j-1]))+1;
            }
        }
    }
}

```

```

    }
    return dp[m][n];
}
}

```

Output:

The screenshot shows a coding platform interface with a dark theme. On the left, the 'Output Window' is open, displaying 'Compilation Results' for a problem solved successfully. The statistics shown are: Test Cases Passed: 1115 / 1115, Attempts: Correct / Total: 1 / 1, Accuracy: 100%, Points Scored: 8 / 8, and Time Taken: 0.16. Below these, it says 'Your Total Score: 16' with an upward arrow. At the bottom, there are buttons for 'Solve Next' and 'Interleaved Strings', 'Shortest Common Supersequence', and 'Form a palindrome'. On the right, the code editor shows a Java solution for the edit distance problem. The code is as follows:

```

22
23
24 class Solution {
25     public int editDistance(String s1, String s2) {
26         // Code here
27         int m=s1.length();
28         int n=s2.length();
29         int[][] dp=new int[m+1][n+1];
30
31         for(int i=0;i<=m;i++)
32             dp[i][0]=i;
33
34         for(int j=0;j<=n;j++)
35             dp[0][j]=j;
36
37         for(int i=1;i<=m;i++){
38             for(int j=1;j<=n;j++){
39                 if(s1.charAt(i-1)==s2.charAt(j-1))
40                     dp[i][j]=dp[i-1][j-1];
41                 else
42                     dp[i][j]=Math.min(dp[i-1][j],Math.min(dp[i][j-1],dp[i-1][j-1]))
43             }
44         }
45         return dp[m][n];
46     }
47 }
48
49
50

```