1. Find an Element in a sorted rotated array.
2. Sort the array containing only 0 and 1
3. Count the no of each characters present in a string
4. Reverse the String using Recursion
5. Reverse the String in O(n/2) time complexity
6. Sort the Stack
7. Merge the Two Sorted Array
8. Return the first non repeated character from a String
9. Find if tow Binary Search tree are mirror of each other
10. Find the missing no in an array ex [1,2,0,4,5,6] , 0 represents missing no
11. Check if a string present in another string
12. Check if a string is anagram or not
13. Delete a node in Linked List without looping through with O(1) time complexity(
14. Check if a Linked List contains cycle
15. Find the length of a linked list
16. Find the middle element in a Linked List
17. Reverse a Linked List
18. Parenthesis matches {{{}}}} or [{()}]
19. Second highest element an array.
20. Rotate an array
21. If a given no is equal to sum of two nos in an array return the nos(unique or first two nos which sum is equal to given no)
22. Find a duplicate no in a given array
23. 1. Given an array containing only 0's and 1's, sort the array in one traversal
24. 2. Print the elements in a matrix in spiral manner
25. 3. Basic java questions. ArrayList-LinkedList difference, Thread mechanisms, synchronized, Spring basics.
26. 4. Given all available routes between different cities, find whether a route exists between 2 given cities. eg -  Route is there between (chn-bang), (bang-hyd),(del-mum).. Chn-hyd should return yes and chn-mum should return false
27. Producer, consumer problem using queues in java
28. Remove the spaces from the string without using inbuilt functions. Remove duplicates.
29. Reverse a Linked List. using recursion.
30. Given a sorted array containing duplicate numbers, find the starting index and the ending index of a particular number. Optimize it.
31. Given an array, find all the pairs of elements having the given sum.
32. Optimize merge sort .
33. find pairs in an unsorted array with given sum
34. Inter thread communication
35. Garbage collection , how memory is divided
36. Serialization
37. Singleton Design pattern
38. Immutable class and implemenation =>
39. Inheritance, static and final (Vechicle Plot parking example)
40. Insert an element in sorted array
41. How set stores unique values internally
42. diff get and post
43. find the nth largest no in an arrays
44. Find the nth element from the last index of linked list using recursion
45. When would you make a variable volatile?
46. How do you decide whether to use a HashMap or a TreeMap?
47. How do you decide whether to use a CopyOnWriteArrayList or a Collections.synchronizedList(ArrayList)?
48. When would you use mergesort over quicksort?
49. When might recursion cause stack overflows?
50. Is Math.abs(Random.nextInt()) always positive?
51. What are the pros/cons of async vs. blocking I/O?
52. What unit tests would you write for Arrays.binarySearch?
53. Intersection of two linkedList
54. Check linkedList is palindrom or not
55. Possible combination of Steps, user can use 2 steps max to climb
56. Find multiple missing nos in array
57. Permutation combination of given array
58. Print the nth number from last in linked list
59. A Program to check if strings are rotations of each other or not
60. Given a string s1 and a string s2, write a snippet to say whether s2 is a rotation of s1 using only one call to strstr routine? (eg given s1 = ABCD and s2 = CDAB, return true, given s1 = ABCD, and s2 = ACBD , return false)
61. Customer , order database design-> Find the customers who have not placed any order yet
62. Types of indexes in database
63. Difference between controller and RestController Spring
64. Difference between prototype and \_\_proto
65. Difference between ngOnInit() and constructor
66. Difference between apply and bind in javascript
67. @Input
68. Html 5 Features
69. Diff between component and directives
70. Spring Security ( What is @Secured)
71. Exception Handling in Rest
72. Authentication in Rest api (Spring)
73. 9 coin puzzle ,8 coin with same weight and 1 with lighter, using beam balance find the fake coin
74. Customer – product – order relationship Databases design
75. Hospital ( Doctor, patients, Appointment ) Database Design
76. Jenkins
77. Second highest number or nth highest number in An Array
78. Given set of chars {a,e,b,c,n,f,g,a,I,o,n,e,k,l,f,d,z,y,x} and string “anand”, find String “anand” in given set of chars, chars should be present in sequential order.  
    ex chars {a,e,b,c,n,f,g,a,I,o,n,e,k,l,f,d,z,y,x} and String “anand”, - > true  
    but “xyz” is false
79. Data sharing between different components in Angular
80. ES6 features
81. Difference between Observer and Promise
82. Difference between Subject and BehaviourSubject
83. Mixing in CSS
84. ClassNotFoundException vs. NoClassDefFoundError
85. Exception chaining/ Wrapping
86. Split a string by spaces but don’t split the string defined in double quotes “”  
    example
87. Password = previous password + 5000+ intern id   
    You will give (no of interns , password )  
    Output = intern Id
88. 10 glass of Milk, one of them contain poison, after consuming poison you need 10 hrs to be killed. You have a rat and 24hrs find glass contains glass which has poison

Replace all duplicate chars so that all chars of the string become unique. This is done by keeping only the first occurrence and replacing the next occurances with the next alphabetical char  
example str=”treeface” output : trefgach  
treeface -> multiple occurances of e  
| treffacf -> now multiple occrances of f  
| trefgacg -> now multiple occurances of g  
| trefgach -> now all the chars are unique in this string.   
  
solution :  
 **private** **static** **void** removeDuplicates(String str) {

**int** count=0;

**for**(**int** i=0;i<str.length();i++){

count=0;

**for**(**int** j=0;j<str.length();j++){

**if**(str.charAt(i)==str.charAt(j)){

count++;

}

**if**(count>=2){

String tmp=str.substring(j);

**char** c=(**char**) (str.charAt(i)+1);

tmp=tmp.replace(str.charAt(i), c);

str=str.substring(0, j)+tmp;

}

}

}

System.***out***.println(str);

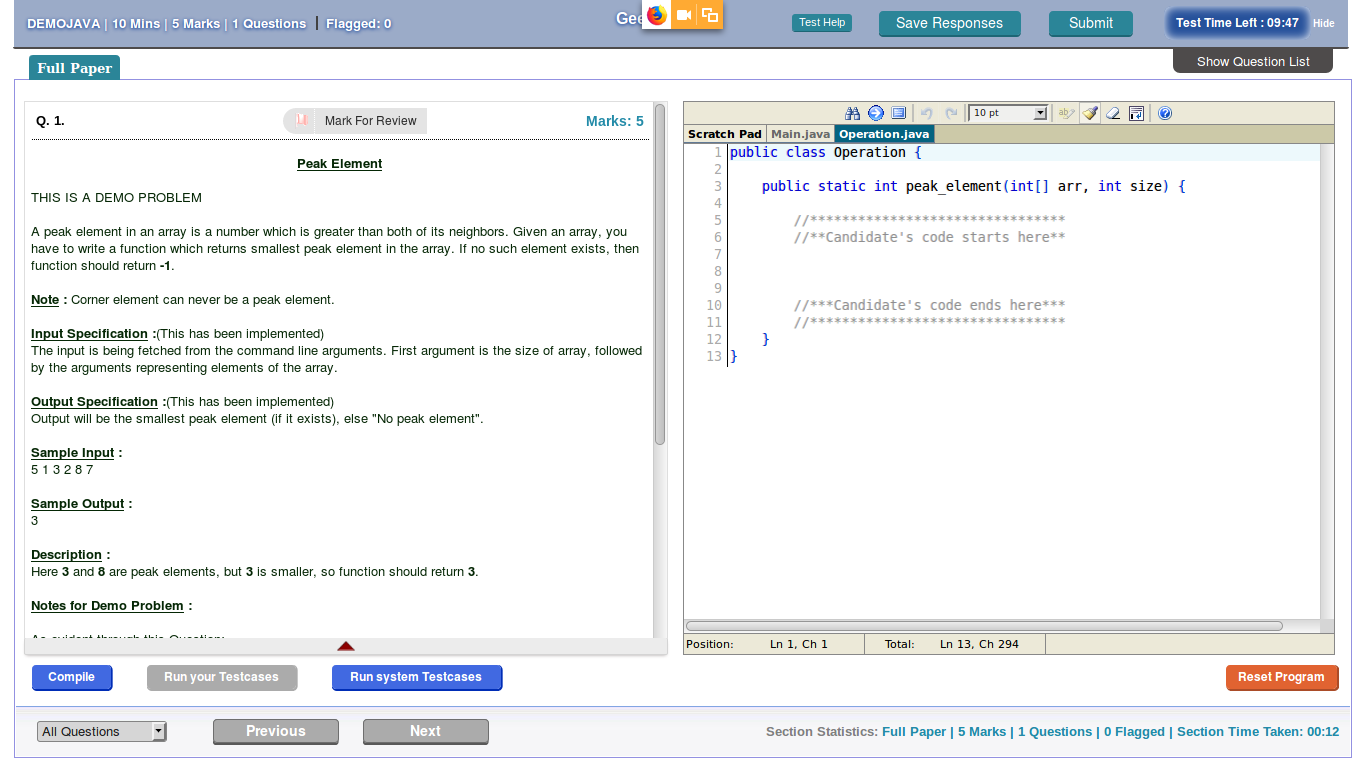
}

1. Print the matrix in the below order.

1 2 3 4  
5 6 7 8  
9 10 11 12  
13 14 15 16  
17 18 19 20  
21 22 23 24

Output would be 1,2,3,4,8,12,11,10,9,13,17,18,19,20,24

import java.util.stream.IntStream;  
  
public class ZigZagArray{  
  
    static void print(int ar[],boolean reverse){  
        if(reverse){  
            for(int i=ar.length-1;i>=0;i--){  
               System.out.print("\t"+ar[i]);  
           }  
        }  
        else{  
             for(int i=0;i<ar.length;i++){  
               System.out.print("\t"+ar[i]);  
           }  
        }  
    }  
    static int[][] createArray(int row,int col){  
        int ar[][]=new int[row][col];  
        int k=1;  
        for(int i=0;i<row;i++){  
            for(int j=0;j<col;j++){  
                ar[i][j]=k++;  
            }  
        }  
        for(int i=0;i<row;i++){  
            for(int j=0;j<col;j++){  
                System.out.print("\t"+ar[i][j]);  
            }  
            System.out.println("");  
        }  
        System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
        return ar;  
    }  
    public static void main(String[] args) {  
        int ar[][]=createArray(10, 2);  
          
        int k=1;  
        System.out.println(ar.length);  
        boolean reverse=false;  
       boolean last=true;  
        for(int i=0;i<ar.length;i++){  
            if(k%2==0){  
                if(last){  
                    IntStream.range(0, ar[0].length-1).forEach(x->System.out.print("\t"));  
                    System.out.print("\t"+ar[i][ar[0].length-1]);  
                    last=false;  
                }  
                else{  
                   IntStream.range(0, ar[0].length-1).forEach(x->System.out.print("\t"));  
                     System.out.print("\t"+ar[i][0]);  
                    last=true;  
                }  
                System.out.println("");  
            }  
            else{  
                    if(reverse){  
                   print(ar[i],reverse);  
                   reverse=false;  
               }  
               else{  
                 print(ar[i],reverse);  
                 reverse=true;  
               }  
               System.out.println("");  
            }  
           k++;  
        }  
    }  
      
}



public class Main {  
  
 public static void main(String[] args)  
 {  
 int [] intA = new int[]{5,1,6,2,5,3,4};  
  
 int min = Integer.*MAX\_VALUE*;  
 for(public class Main {  
  
 public static void main(String[] args)  
 {  
 int [] intA = new int[]{5,1,6,2,5,3,4};  
  
 int min = Integer.*MAX\_VALUE*;  
 for(int i = 1; i <intA.length -2;i++){  
 if(intA[i-1]< intA[i] && intA[i] >intA[i+1]){  
 if(min > intA[i]){  
 min = intA[i];  
 }  
 }  
 }  
 System.*out*.println(min);  
 }  
  
  
}

**public** **class** PeakElement {

**public** **static** **void** main(String[] args) {

// 5 1 3 2 8 7 peak elements are 2 and 8

**int** ar[] = { 5, 1, 6, 2, 5, 3, 4 };

**int** left, right, mid = 0;

**int** min = 0;

List<Integer> peakElements = **new** ArrayList();

**for** (**int** i = 0; i < ar.length; i++) {

**if** (i + 2 >= ar.length) {

**break**;

}

left = ar[i];

mid = ar[i + 1];

right = ar[i + 2];

**if** (mid > left && mid > right) {

peakElements.add(mid);

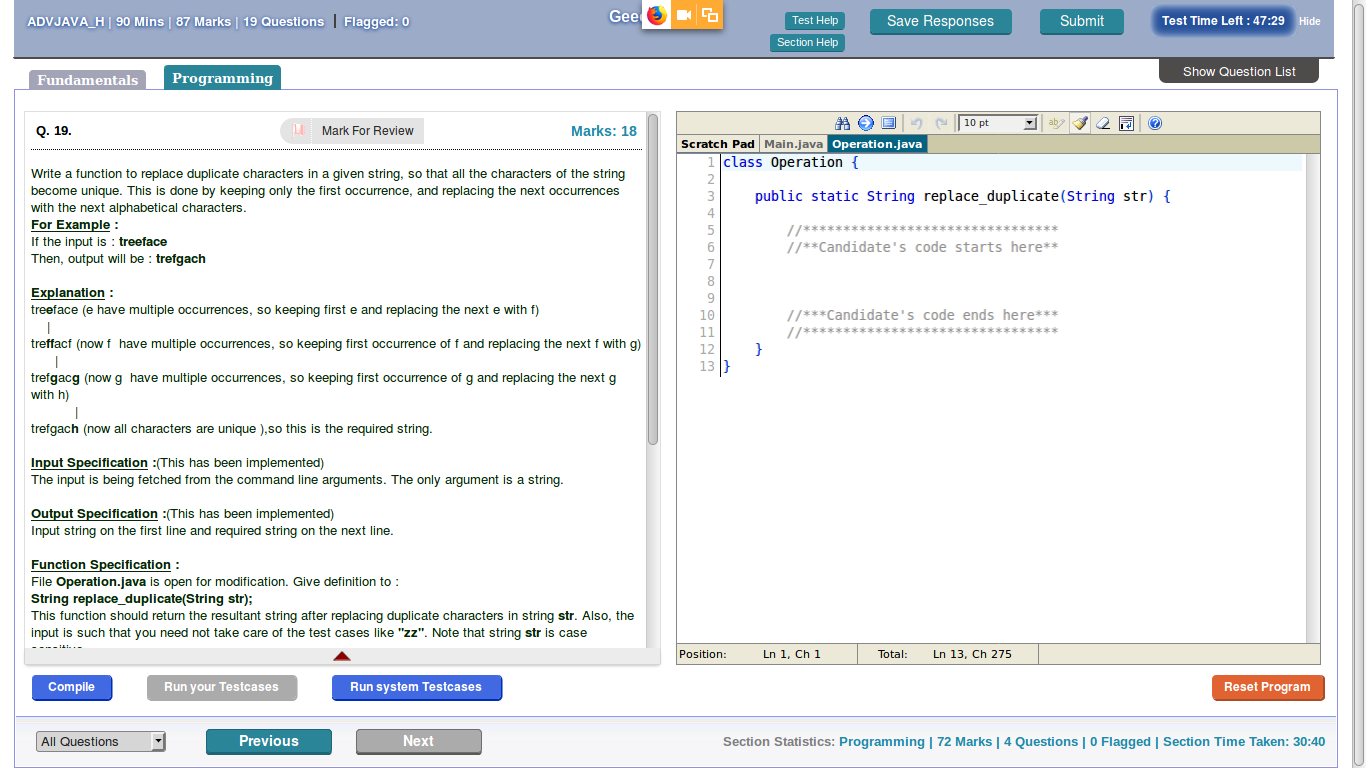
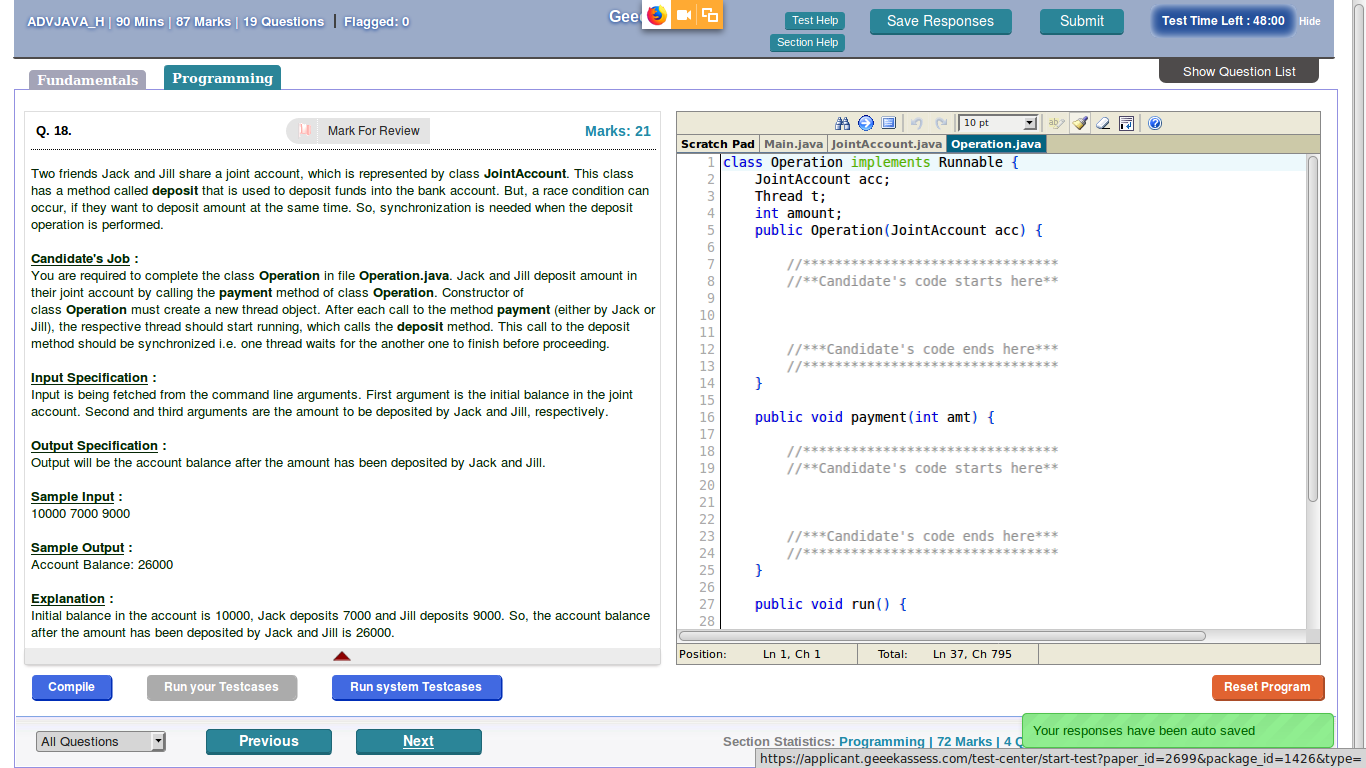
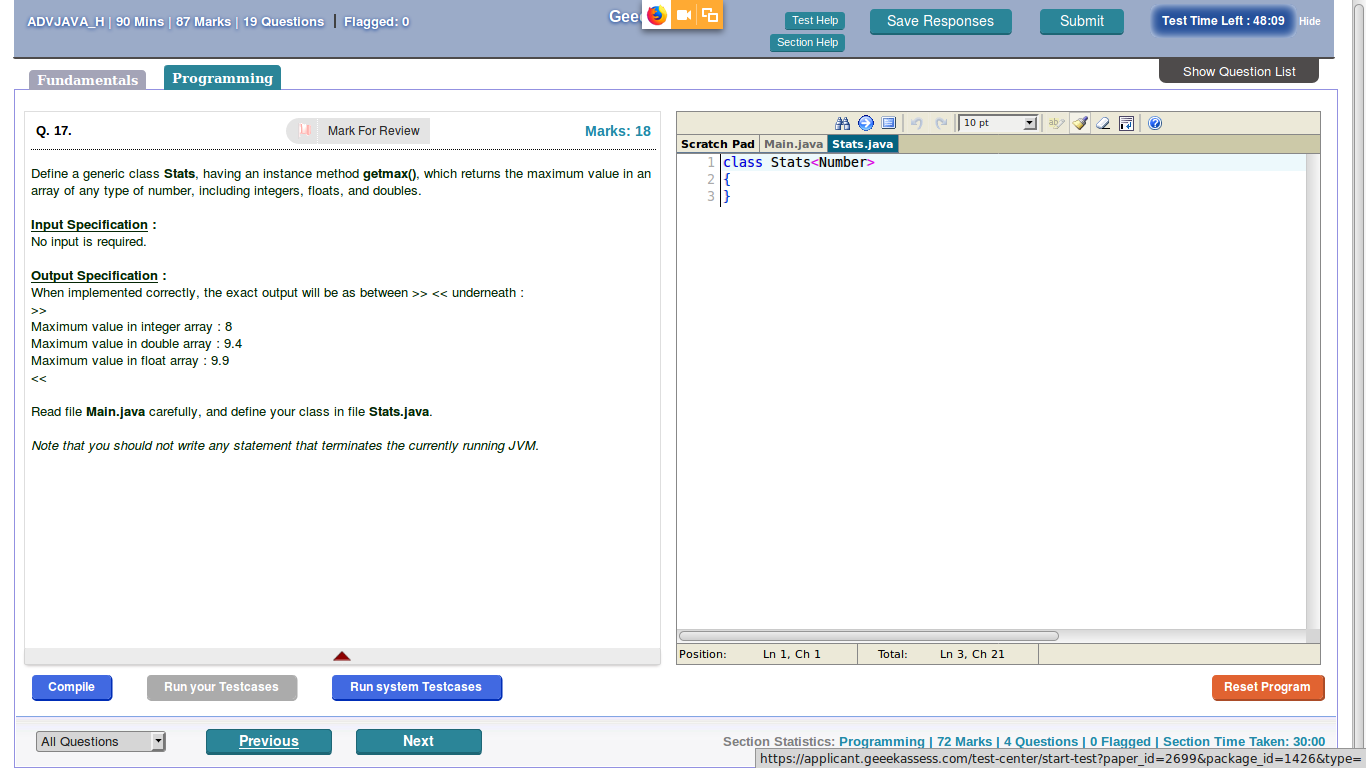
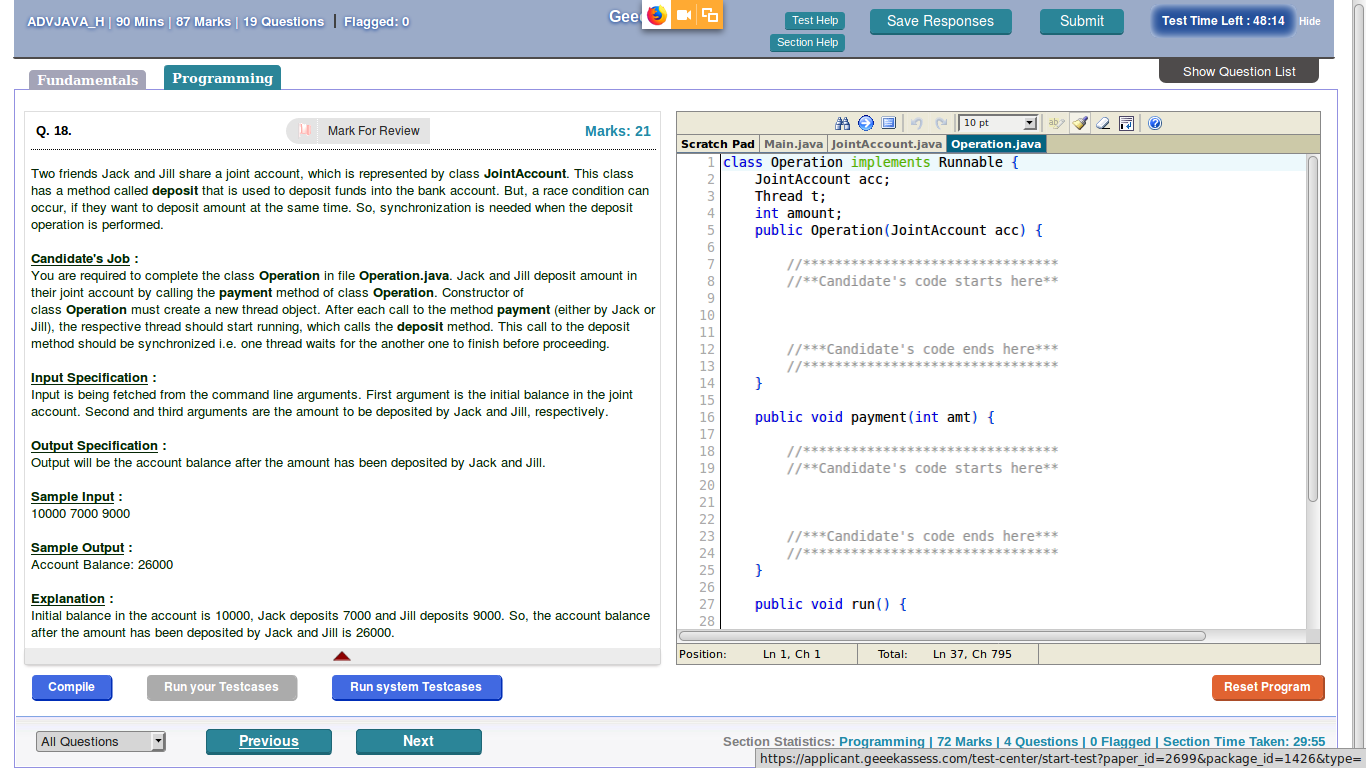
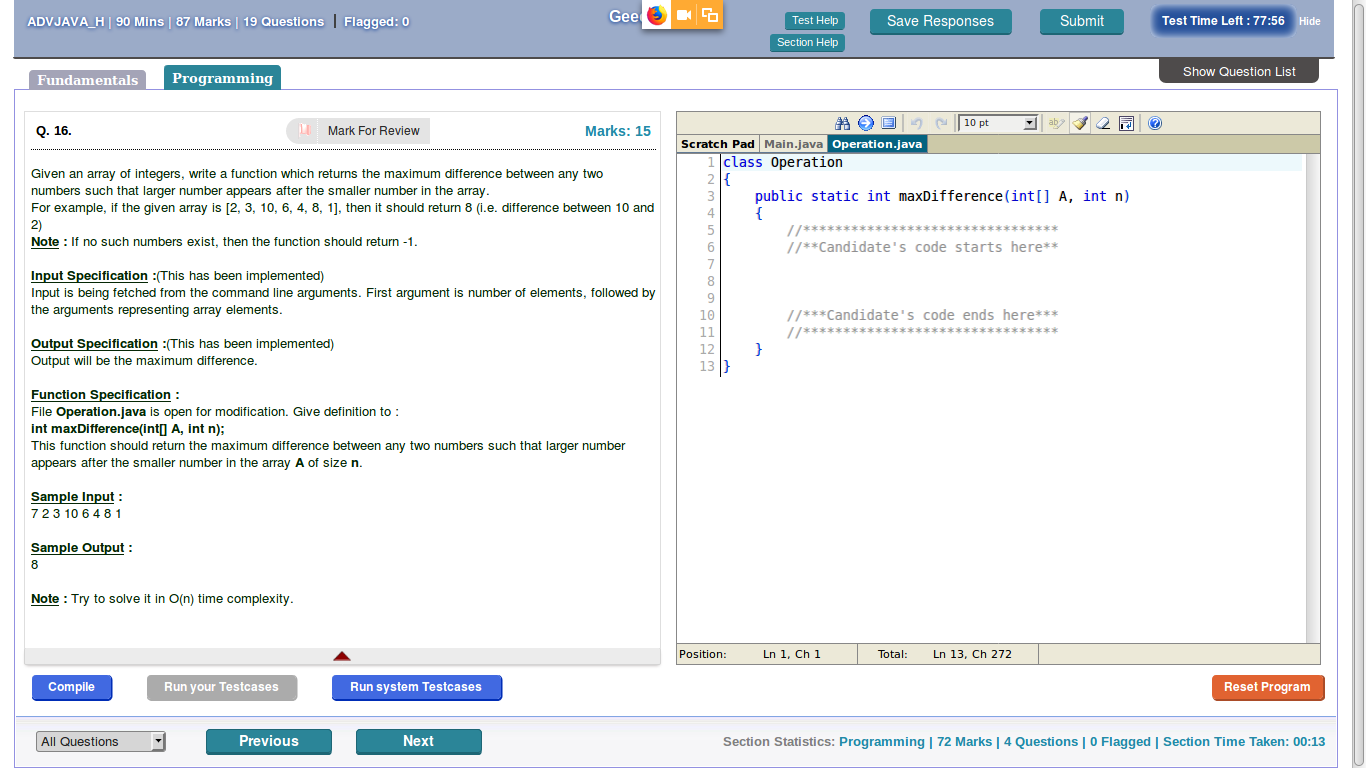
}

}

System.***out***.println(peakElements.stream().min((x, y) -> x.compareTo(y)).orElse(0));

}

}



**public** **class** MaxElement {

**public** **static** **void** main(String[] args) {

**int** ar[] = { 10,29,2,3,4,5,67,8 };

Set<Integer> list = **new** LinkedHashSet<>();

**for** (**int** j = 0; j < ar.length; j++) {

**int** prev = ar[j];

**for** (**int** i = j+1; i < ar.length; i++) {

**if** (prev < ar[i]) {

list.add(ar[i] - prev);

}

}

}

System.***out***.println(list);

System.***out***.println(list.stream().max((x, y) -> x.compareTo(y)).orElse(0));

}

}

85. Find the department information which has maximum number of employees.

SELECT count(\*), department\_id  
FROM employees  
GROUP BY department\_id  
HAVING count(\*) =  
(SELECT max(count(\*))  
FROM employees  
GROUP BY department\_id)

# 86. Maximum consecutive repeating character in string

**public** **class** ValidPattern {

**static** **char** maxRepeating(String str)

{

**int** n = str.length();

**int** count = 0;

**char** res = str.charAt(0);

**int** cur\_count = 1;

// Traverse string except last character

**for** (**int** i = 0; i < n; i++)

{

// If current character matches with next

**if** (i < n - 1 && str.charAt(i) == str.charAt(i + 1))

cur\_count++;

**else**

{

**if** (cur\_count > count)

{

count = cur\_count;

res = str.charAt(i);

}

cur\_count = 1;

}

}

**return** res;

}

**public** **static** **void** main(String[] args) {

String str="aaaabbcbbb"; // b=1, a=4 , b=5

**char** prev=str.charAt(0);

**int** count=0;

**int** prevCount=0;

**char** c=' ';//32;

**for**(**int** i=1;i<str.length();i++){

**if**(prev==str.charAt(i)){

count++;

**if**(prevCount<count){

prevCount=count;

c=prev;

}

}

**else**{

count=0;

}

prev=str.charAt(i);

}

System.***out***.println(c);

}

# }