https://allhackerranksolutionsbykaira.blogspot.com/search/label/HackerRank

https://awesomeopensource.com/project/Java-aid/Hackerrank-Solutions#data-structures

MEANDERING ARRAY

```
// Java program to print the array in given order
import java.util.Arrays;
public class GFG {
 // Function which arrange the array.
 static void rearrangeArray(int arr[], int n)
 {
  // Sorting the array elements
  Arrays.sort(arr);
  int[] tempArr = new int[n]; // To store modified array
  // Adding numbers from sorted array to
  // new array accordingly
  int ArrIndex = 0;
  // Traverse from begin and end simultaneously
  for (int i = 0, j = n-1; i <= n / 2 || j > n / 2;
        i++, j--) {
   if(ArrIndex < n)
   {
    tempArr[ArrIndex] = arr[i];
    ArrIndex++;
   }
   if(ArrIndex < n)
   {
    tempArr[ArrIndex] = arr[j];
    ArrIndex++;
   }
```

```
 }
 // Modifying original array
 for (int i = 0; i < n; i++)
  arr[i] = tempArr[i];
}
// Driver Code
public static void main(String args[])
{
 int arr[] = { 5, 8, 1, 4, 2, 9, 3, 7, 6 };
 int n = arr.length;
 rearrangeArray(arr, n);
 for (int i = 0; i < n; i++)
  System.out.print(arr[i]+" ");
}
}
```

DISTINCT DIGIT NUMBER

```
// Java implementation of brute
// force solution.
import java.util.LinkedHashSet;
class GFG
{
// Function to check if the given
// number has repeated digit or not
static int repeated_digit(int n)
{
 LinkedHashSet<Integer> s = new LinkedHashSet<>();
 // Traversing through each digit
 while (n != 0)
 {
  int d = n % 10;
```

```
 // if the digit is present
 // more than once in the
 // number
 if (s.contains(d))
 {
  // return 0 if the number
  // has repeated digit
  return 0;
 }
 s.add(d);
 n = n / 10;
}
// return 1 if the number has
// no repeated digit
return 1;
}
// Function to find total number
// in the given range which has
// no repeated digit
static int calculate(int L, int R)
{
 int answer = 0;
 // Traversing through the range
 for (int i = L; i < R + 1; ++i)
 {
  // Add 1 to the answer if i has
  // no repeated digit else 0
  answer = answer + repeated_digit(i);
 }
 return answer;
}
```

```
// Driver Code

public static void main(String[] args)

{

 int L = 80, R = 120;

 // Calling the calculate

 System.out.println(calculate(L, R));

}

}
```

TWIN STRING PROBLEM

```
import java.io.IOException;
import java.util.Scanner;

public class TwinStringProblem {
    //creating method for checking two string
    static boolean[] twins(String[] a, String[] b) {
        boolean[] result = new boolean[a.length];

        for (int i = 0; i < a.length; i++) {
            String aVal = a[i].toLowerCase();
            String bVal = b[i].toLowerCase();
            String[] aValArray = aVal.split("");
            String[] bValArray = bVal.split("");

            for (String s : aValArray) {
                for (int index = 0; index < aValArray.length; index++) {
                    if (bValArray[index].equals(s)) { // checking whether the
index match or not
                        if ((s.indexOf(s) % 2 == 0 && index % 2 == 0) || // ch
ecking even values
                                (s.indexOf(s) % 2 != 0 && index % 2 != 0)) {
                            result[i] = false;
                        } else if ((s.indexOf(s) % 2 == 0 && index % 2 != 0)
                                || (s.indexOf(s) % 2 != 0 && index % 2 == 0))
{
                            result[i] = true;
                            break;
                        }
                    }
                }
            }
        }
        return result;
```

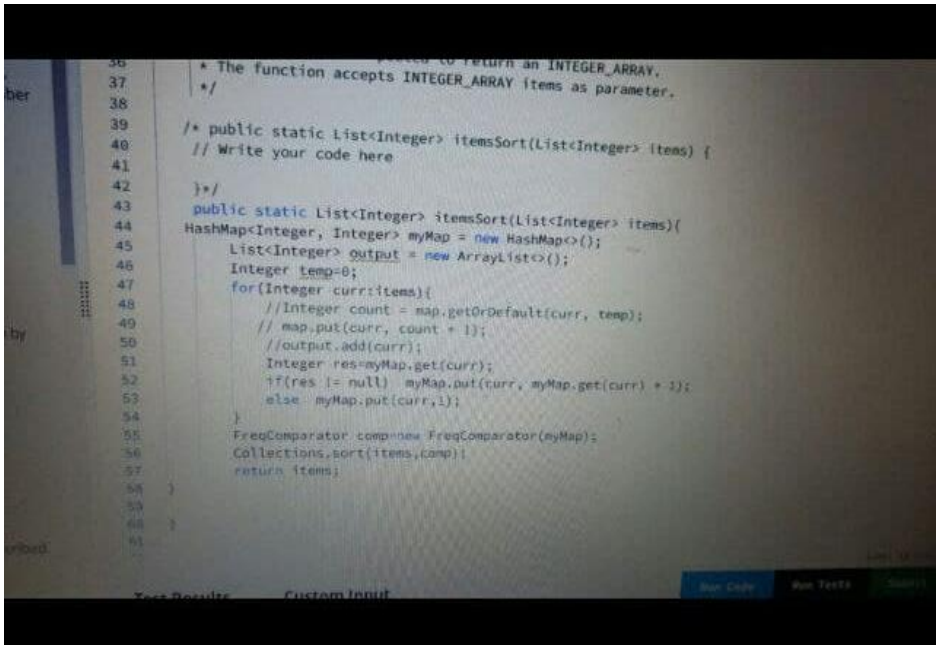
```
    }

    //main Method
    public static void main(String[] args) throws IOException {
        Scanner in = new Scanner(System.in);

        int n = Integer.parseInt(in.nextLine().trim());
        String[] a = new String[n];
        for (int i = 0; i < n; i++) {
            a[i] = in.nextLine();
        }

        int m = Integer.parseInt(in.nextLine().trim());
        String[] b = new String[m];
        for (int i = 0; i < m; i++) {
            b[i] = in.nextLine();
        }

        // call twins function
        boolean[] results = twins(a, b);

        for (int i = 0; i < results.length; i++) {
            System.out.println(results[i] ? "Yes" : "No");
        }
    }
}
```

ARRAY GAME

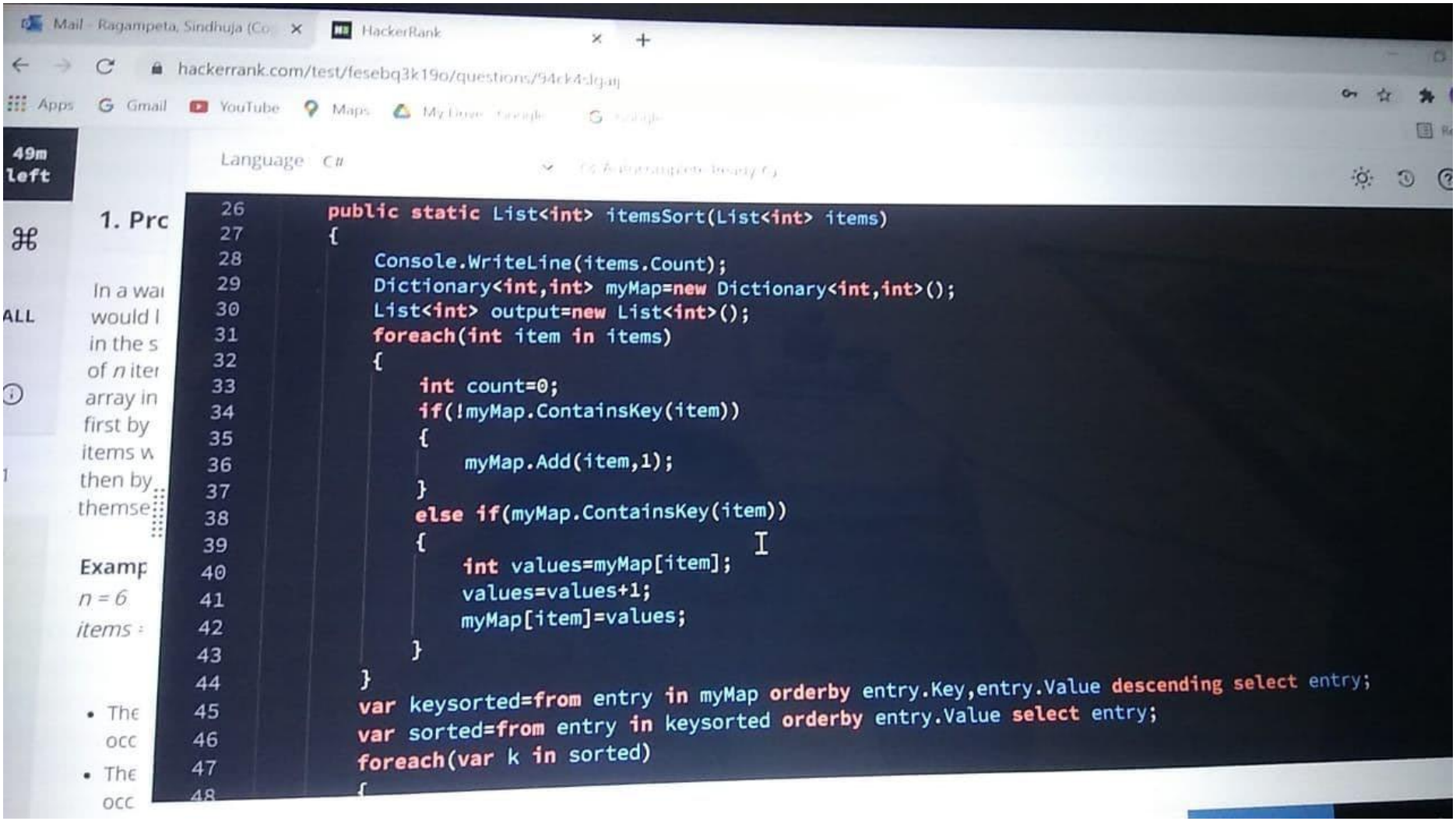
```
20      * The function accepts INTEGER_ARRAY numbers as parameter.
21      */
22
23     public static long countMoves(List<Integer> numbers) {
24     // Write your code here
25     long arrsum,smallest,arr_size=numbers.size();
26     arrsum=0;
27     smallest=(long) numbers.get(0);
28     for(int i=0;i<arr_size;i++){
29         long temp=(long) numbers.get(i);
30         if(temp<smallest){
31             smallest=temp;
32         }
33         arrsum+=temp;
34     }
35     long minoperation = arrsum-arr_size*smallest;
36     return minoperation;
37     }
38
39 }
40 public class Solution {...
```

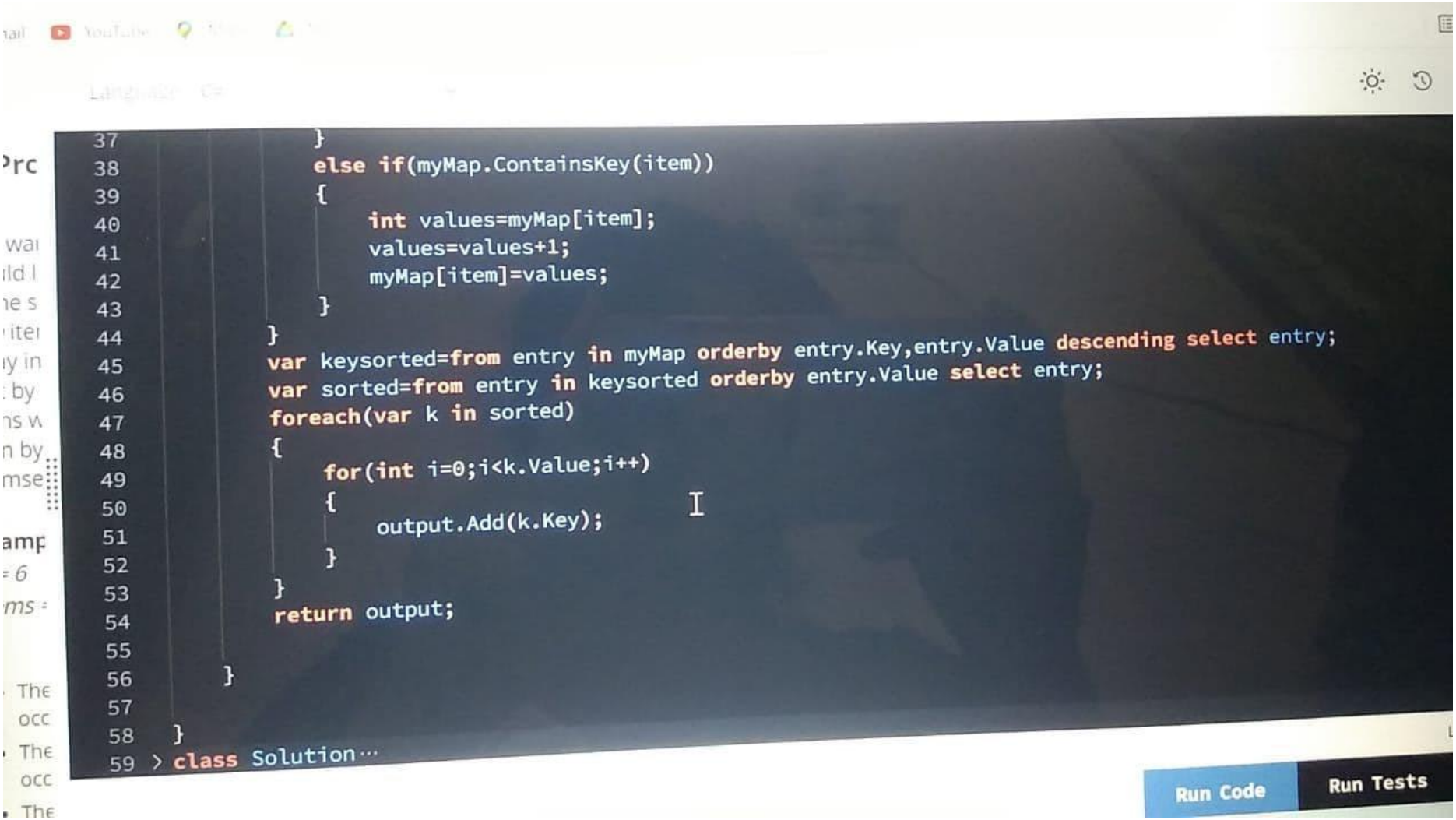
PRODUCT SORT

```
10
11 class FreqComparator implements Comparator<Integer> {
12     private final Map<Integer, Integer> freq;
13     FreqComparator(Map<Integer, Integer> tempFreqMap)
14     {
15         this.freq = tempFreqMap;
16     }
17     @Override
18     public int compare(Integer k1, Integer k2)
19     {
20         int freqCompare = freq.get(k1).compareTo(freq.get(k2));
21         int valueCompare = k1.compareTo(k2);
22         if (freqCompare == 0)
23             return valueCompare;
24         else
25             return freqCompare;
26     }
27 }
28
29
30 class Result {
31
32     /*
33      * Complete the 'itemSort' function below.
```



ANOTHER SOLUTION


Mail - Ragampeta, Sindhuja (Co
HackerRank
hackerrank.com/test/fesebq3k19o/questions/94ck4slgajj
Apps
Gmail
YouTube
Maps
49m left
Language C#
1. Prc
In a wai
would l
in the s
of n iter
array in
first by
items w
then by
themse
Examp
n = 6
items =
• The
occ
• The
occ
ALL
public static List<int> itemsSort(List<int> items)
{
Console.WriteLine(items.Count);
Dictionary<int,int> myMap=new Dictionary<int,int>();
List<int> output=new List<int>();
foreach(int item in items)
{
int count=0;
if(!myMap.ContainsKey(item))
{
myMap.Add(item,1);
}
else if(myMap.ContainsKey(item))
{
int values=myMap[item];
values=values+1;
myMap[item]=values;
}
}
var keysorted=from entry in myMap orderby entry.Key,entry.Value descending select entry;
var sorted=from entry in keysorted orderby entry.Value select entry;
foreach(var k in sorted)
{



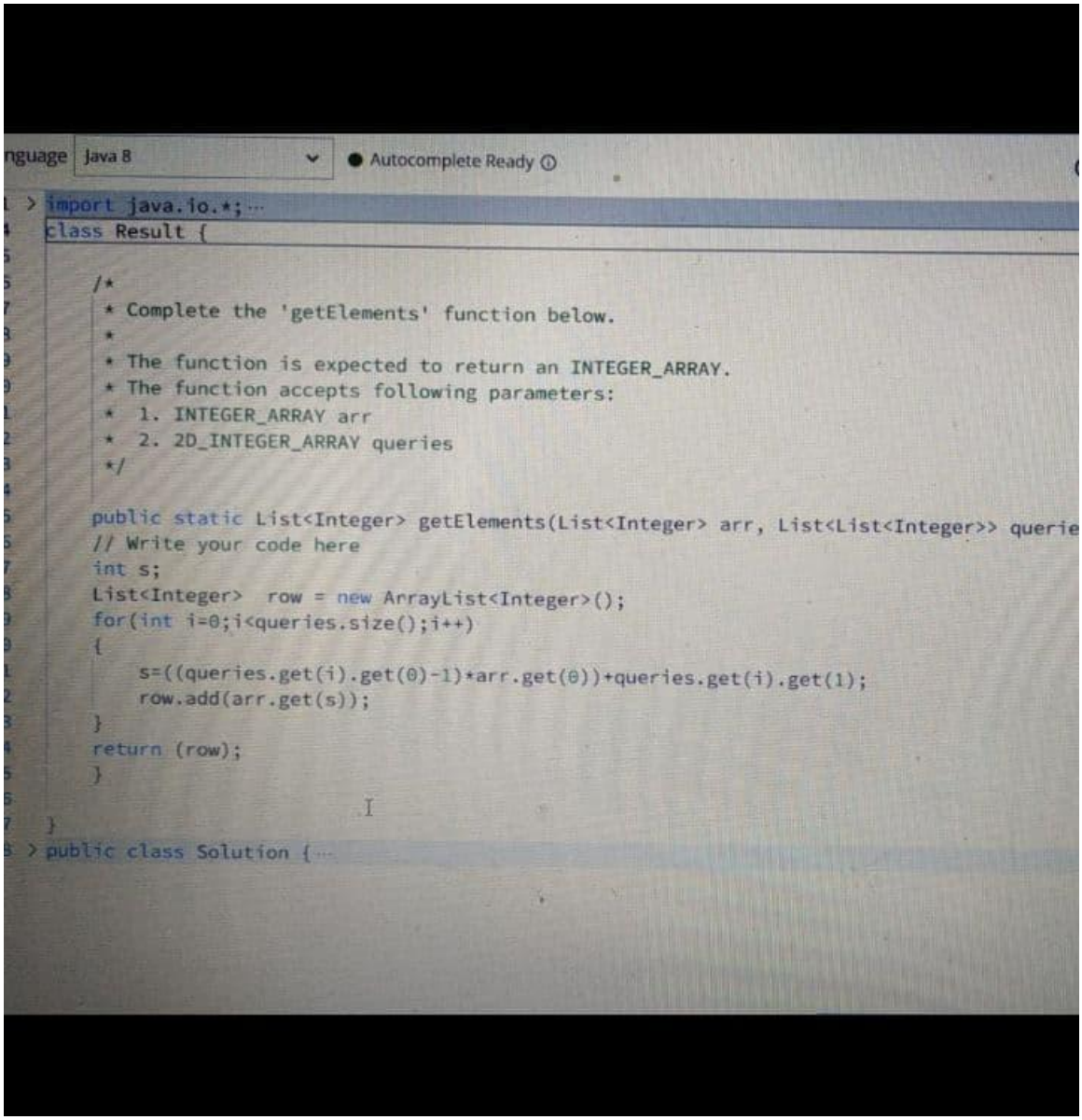
```
            }
            else if(myMap.ContainsKey(item))
            {
                int values=myMap[item];
                values=values+1;
                myMap[item]=values;
            }
        }
        var keysorted=from entry in myMap orderby entry.Key,entry.Value descending select entry;
        var sorted=from entry in keysorted orderby entry.Value select entry;
        foreach(var k in sorted)
        {
            for(int i=0;i<k.Value;i++)
            {
                output.Add(k.Key);
            }
        }
        return output;

    }

}
class Solution…
```

Run Code Run Tests

VOWELS

RESTRUCTURED ARRAY

(another solution also available)

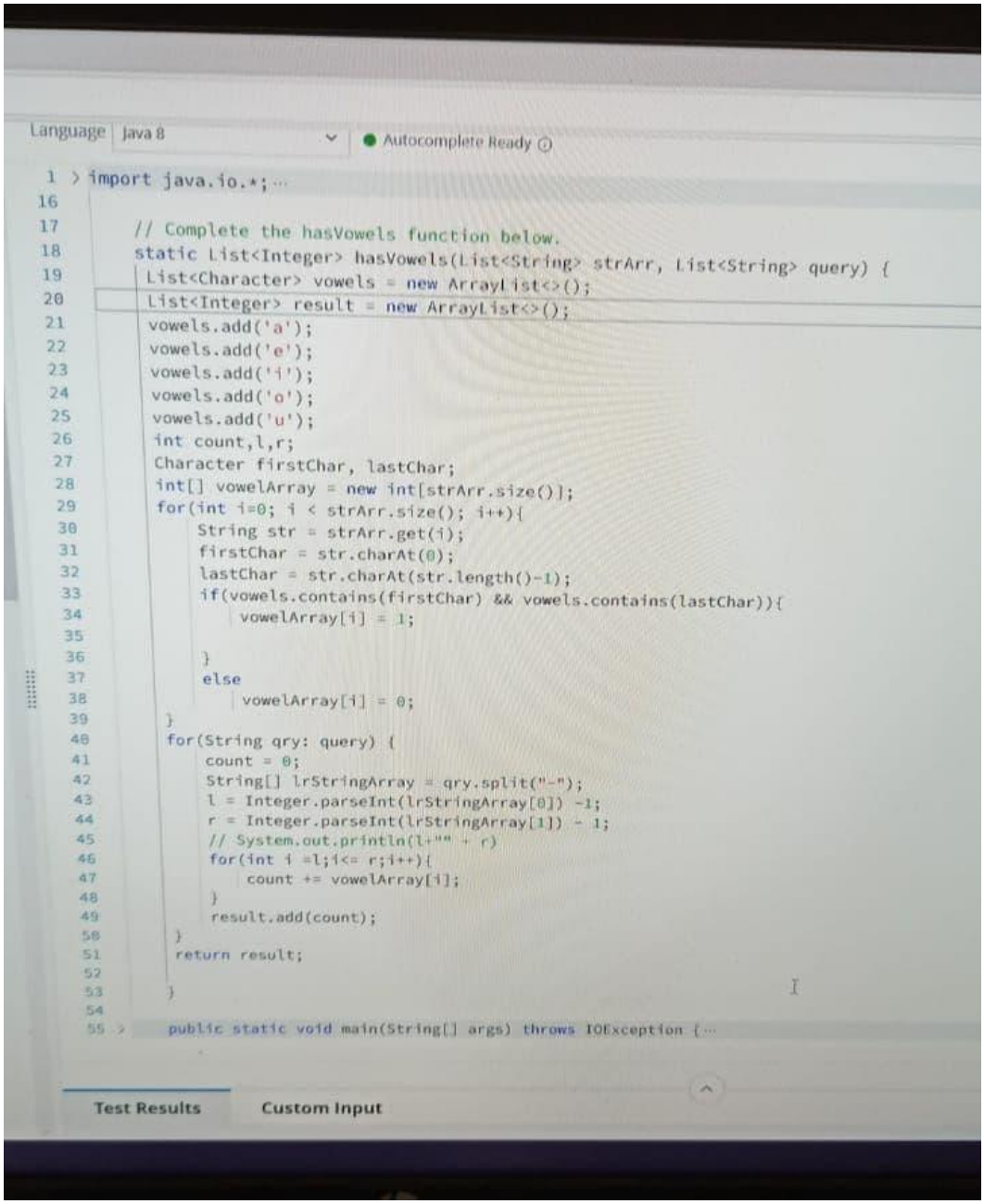


IS IT POSSIBLE

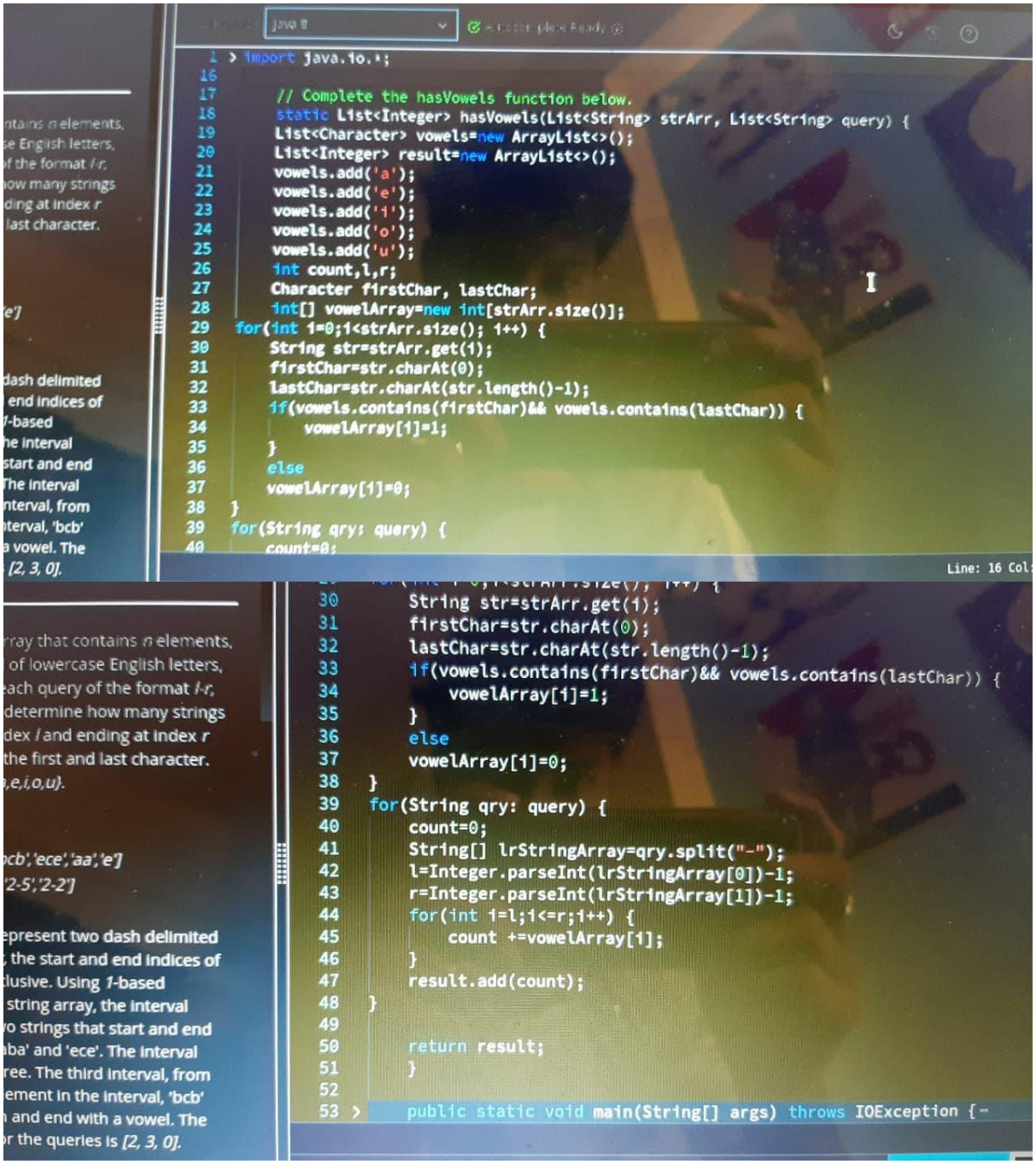
```
static LinkedList<Pair<Integer,Integer>> pairs = new LinkedList<Pair<Integer, Integer>>();


    public static String isItPossible(Integer a, Integer b, Integer c, Integer d){
        pairs.addLast(new Pair<Integer, Integer>(a,b));
        while (!pairs.isEmpty()){
            Pair<Integer,Integer> pair = pairs.poll();
            Integer key = pair.getKey();
            Integer value = pair.getValue();
            if(key.equals(c) &&
```

```
            value.equals(d)){
        return "YES";
      }
      int sum=key+value;
      if (sum<=c){
        pairs.addLast(new Pair<Integer, Integer>(sum,value));
      }
      if (sum<=d){
        pairs.addLast(new Pair<Integer, Integer>(key,sum));
      }

    }

    return "NO";
}
```

VOWELS

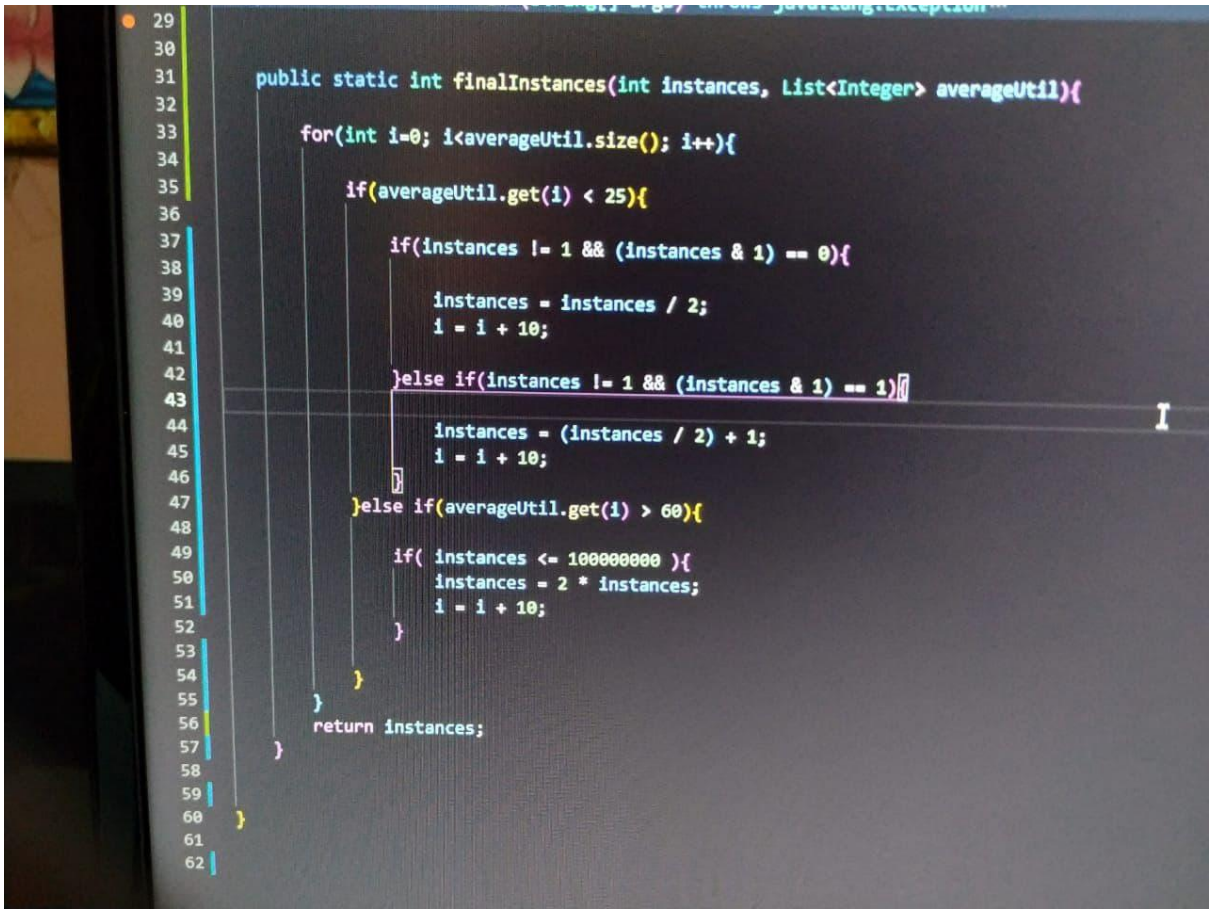


ANOTHER SOLUTION



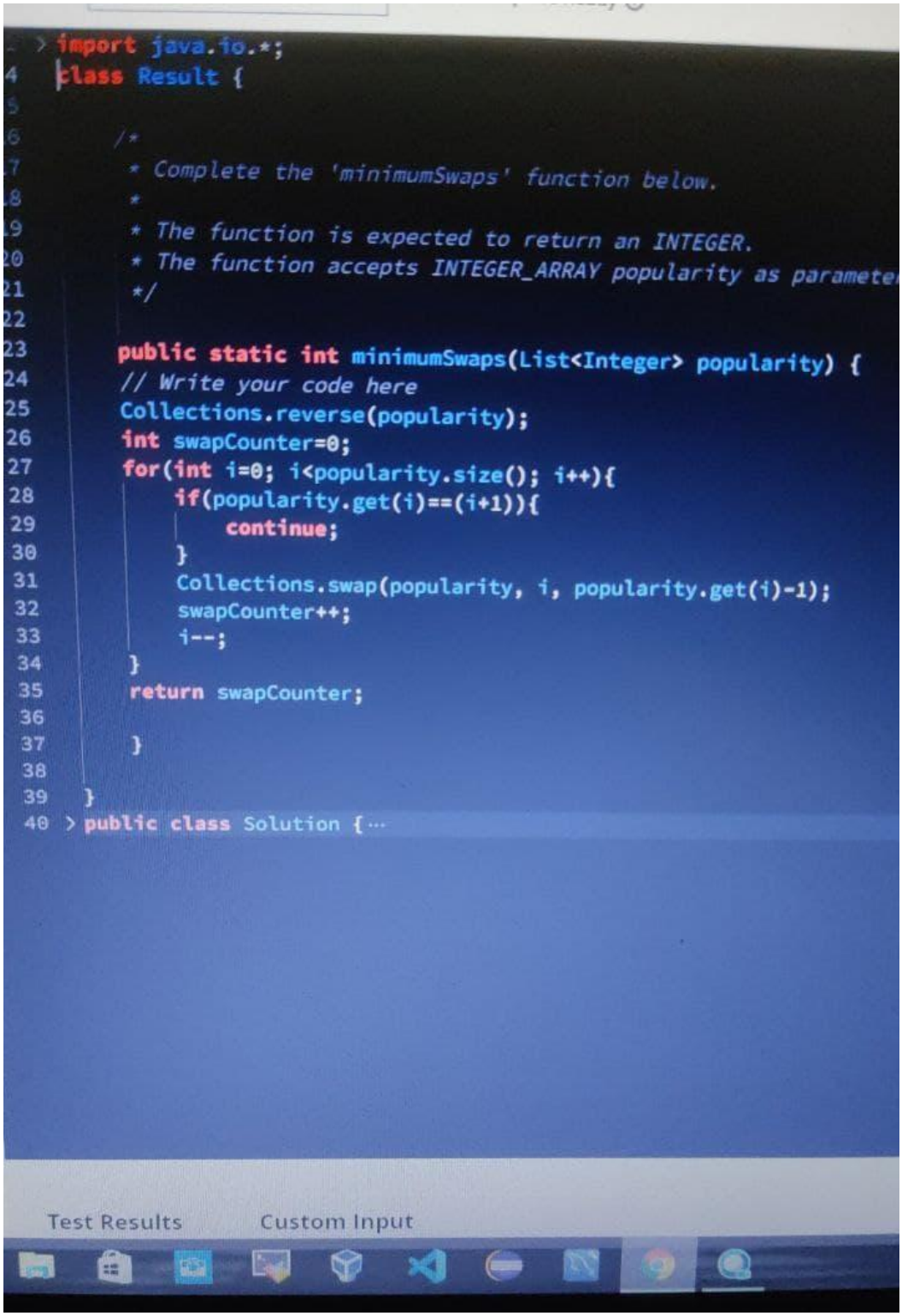
```
import java.io.*;

    // Complete the hasVowels function below.
    static List<Integer> hasVowels(List<String> strArr, List<String> query) {
    List<Character> vowels=new ArrayList<>();
    List<Integer> result=new ArrayList<>();
    vowels.add('a');
    vowels.add('e');
    vowels.add('i');
    vowels.add('o');
    vowels.add('u');
    int count,l,r;
    Character firstChar, lastChar;
    int[] vowelArray=new int[strArr.size()];
for(int i=0;i<strArr.size(); i++) {
    String str=strArr.get(i);
    firstChar=str.charAt(0);
    lastChar=str.charAt(str.length()-1);
    if(vowels.contains(firstChar)&& vowels.contains(lastChar)) {
        vowelArray[i]=1;
    }
    else
    vowelArray[i]=0;
}
for(String qry: query) {
    count=0;
    String[] lrStringArray=qry.split("-");
    l=Integer.parseInt(lrStringArray[0])-1;
    r=Integer.parseInt(lrStringArray[1])-1;
    for(int i=l;i<=r;i++) {
        count +=vowelArray[i];
    }
    result.add(count);
}

    return result;
    }

    public static void main(String[] args) throws IOException {
```

AUTOSCALE POLICY



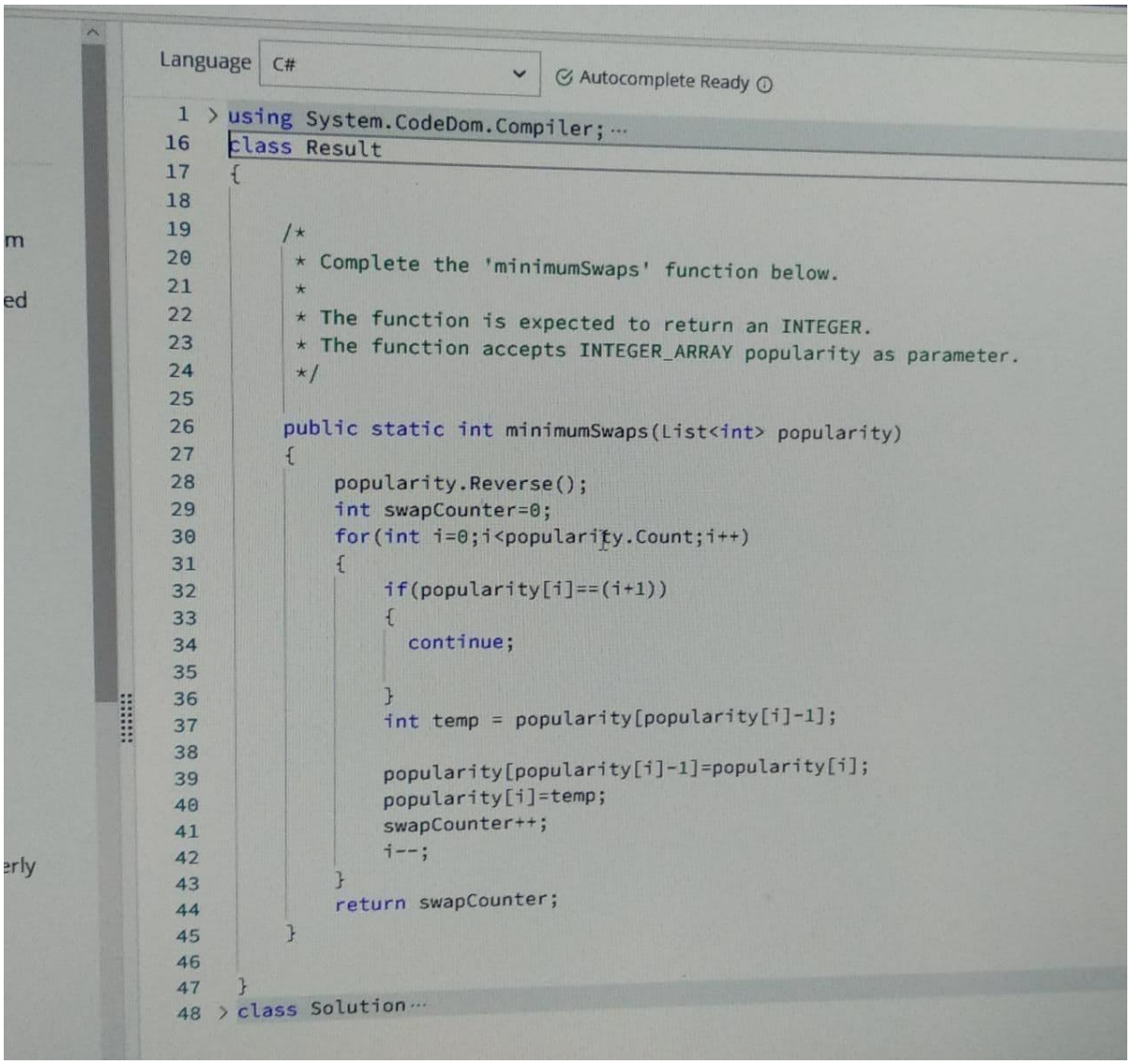
```
29
30
31      public static int finalInstances(int instances, List<Integer> averageUtil){
32
33          for(int i=0; i<averageUtil.size(); i++){
34
35              if(averageUtil.get(i) < 25){
36
37                  if(instances != 1 && (instances & 1) == 0){
38
39                      instances = instances / 2;
40                      i = i + 10;
41
42                  }else if(instances != 1 && (instances & 1) == 1){
43
44                      instances = (instances / 2) + 1;
45                      i = i + 10;
46                  }
47              }else if(averageUtil.get(i) > 60){
48
49                  if( instances <= 100000000 ){
50                      instances = 2 * instances;
51                      i = i + 10;
52                  }
53
54              }
55          }
56          return instances;
57      }
58
59
60  }
61
62
```

MINIMUM SWAPS



```
import java.io.*;
class Result {

    /*
     * Complete the 'minimumSwaps' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY popularity as parameter
     */

    public static int minimumSwaps(List<Integer> popularity) {
    // Write your code here
    Collections.reverse(popularity);
    int swapCounter=0;
    for(int i=0; i<popularity.size(); i++){
        if(popularity.get(i)==(i+1)){
            continue;
        }
        Collections.swap(popularity, i, popularity.get(i)-1);
        swapCounter++;
        i--;
    }
    return swapCounter;

    }

}
public class Solution {...
```

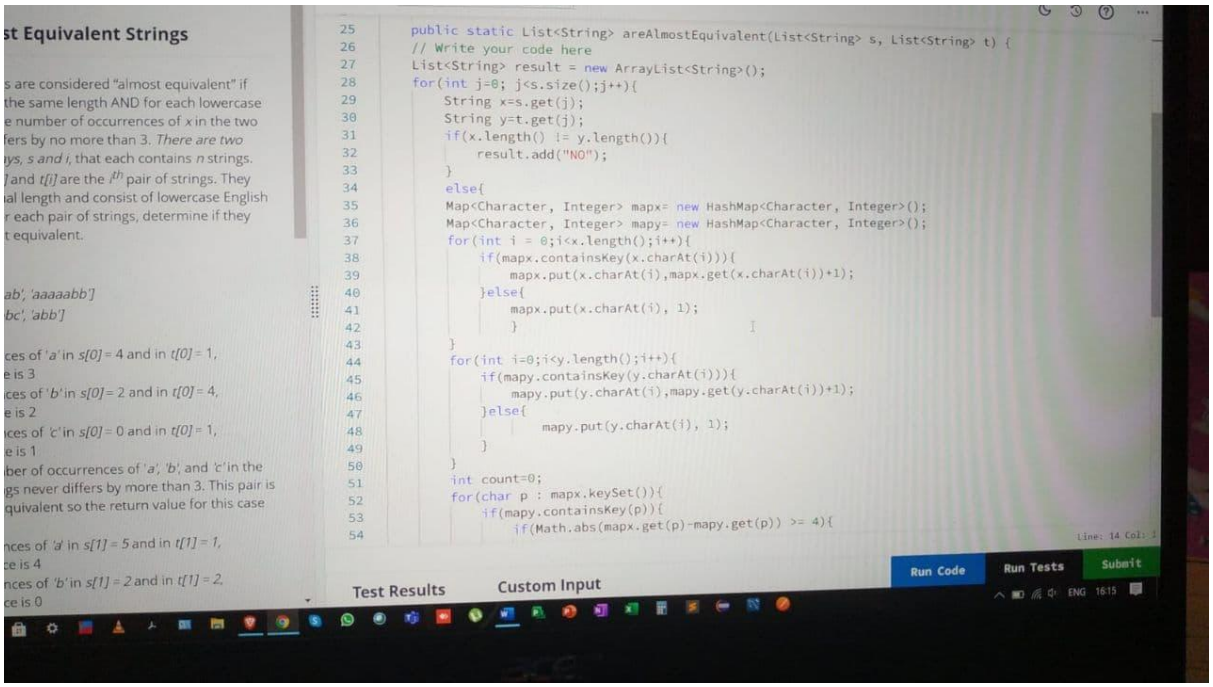
ANOTHER SOLUTION

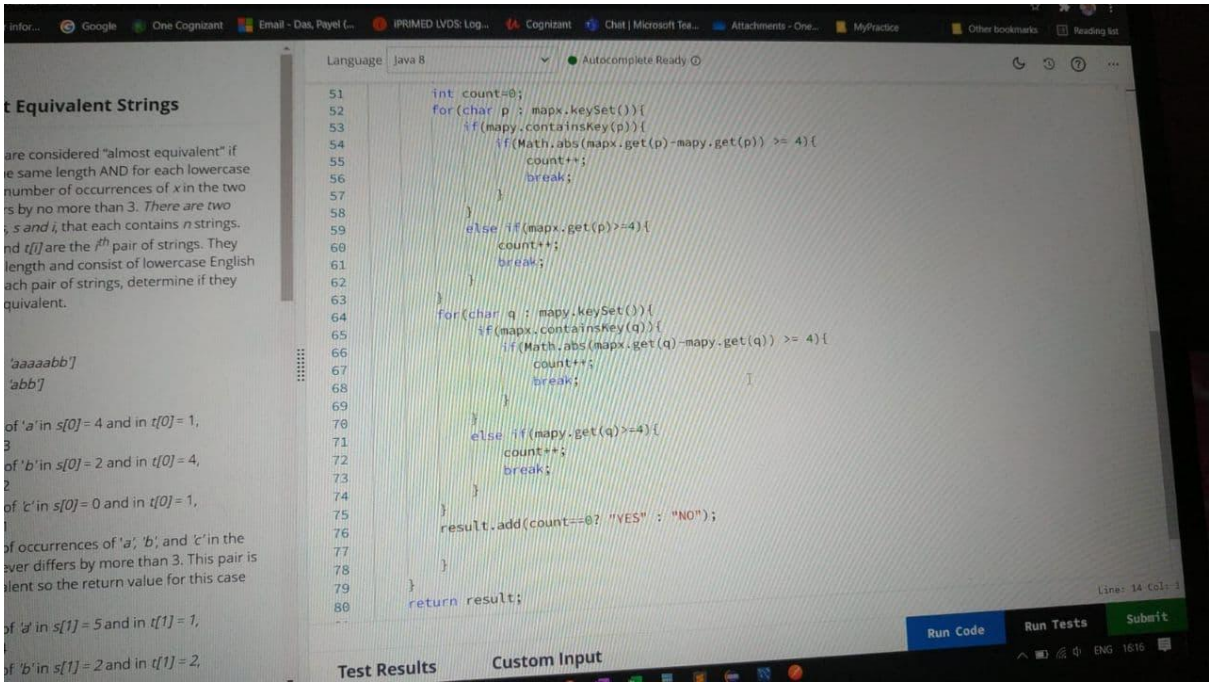


Language C# Autocomplete Ready

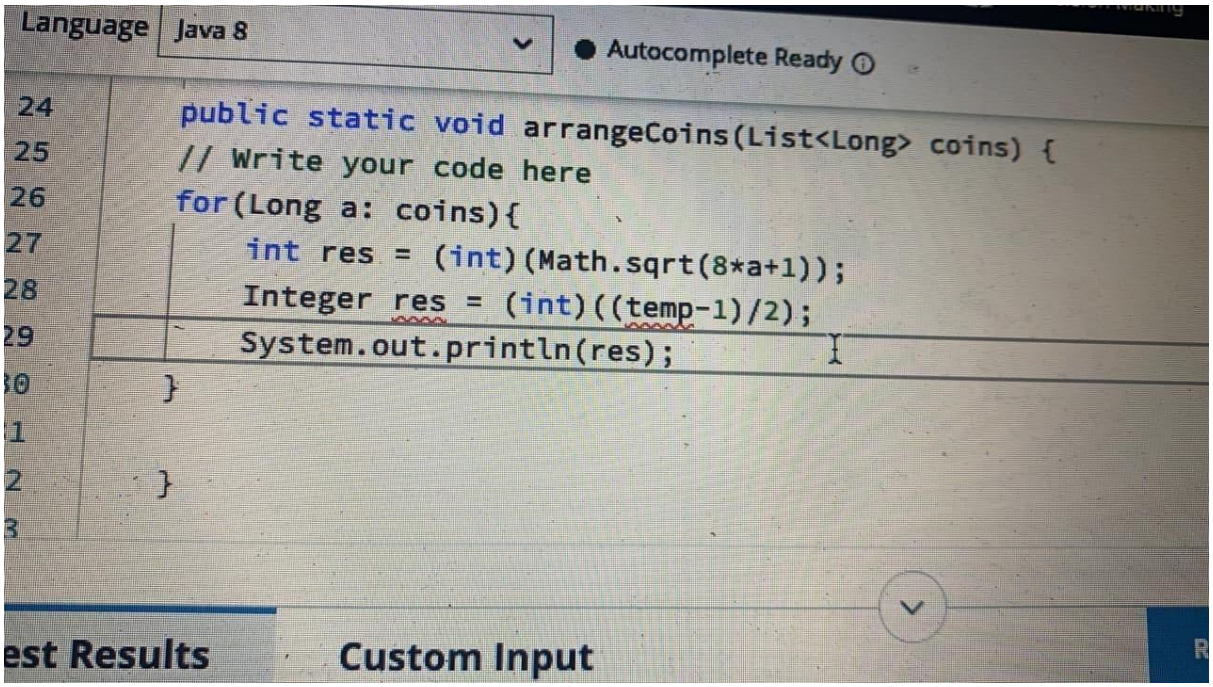
```
using System.CodeDom.Compiler;...
class Result
{

    /*
     * Complete the 'minimumSwaps' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY popularity as parameter.
     */

    public static int minimumSwaps(List<int> popularity)
    {
        popularity.Reverse();
        int swapCounter=0;
        for(int i=0;i<popularity.Count;i++)
        {
            if(popularity[i]==(i+1))
            {
              continue;

            }
            int temp = popularity[popularity[i]-1];

            popularity[popularity[i]-1]=popularity[i];
            popularity[i]=temp;
            swapCounter++;
            i--;
        }
        return swapCounter;
    }

}
class Solution...
```

ARE ALMOST EQUIVALENT

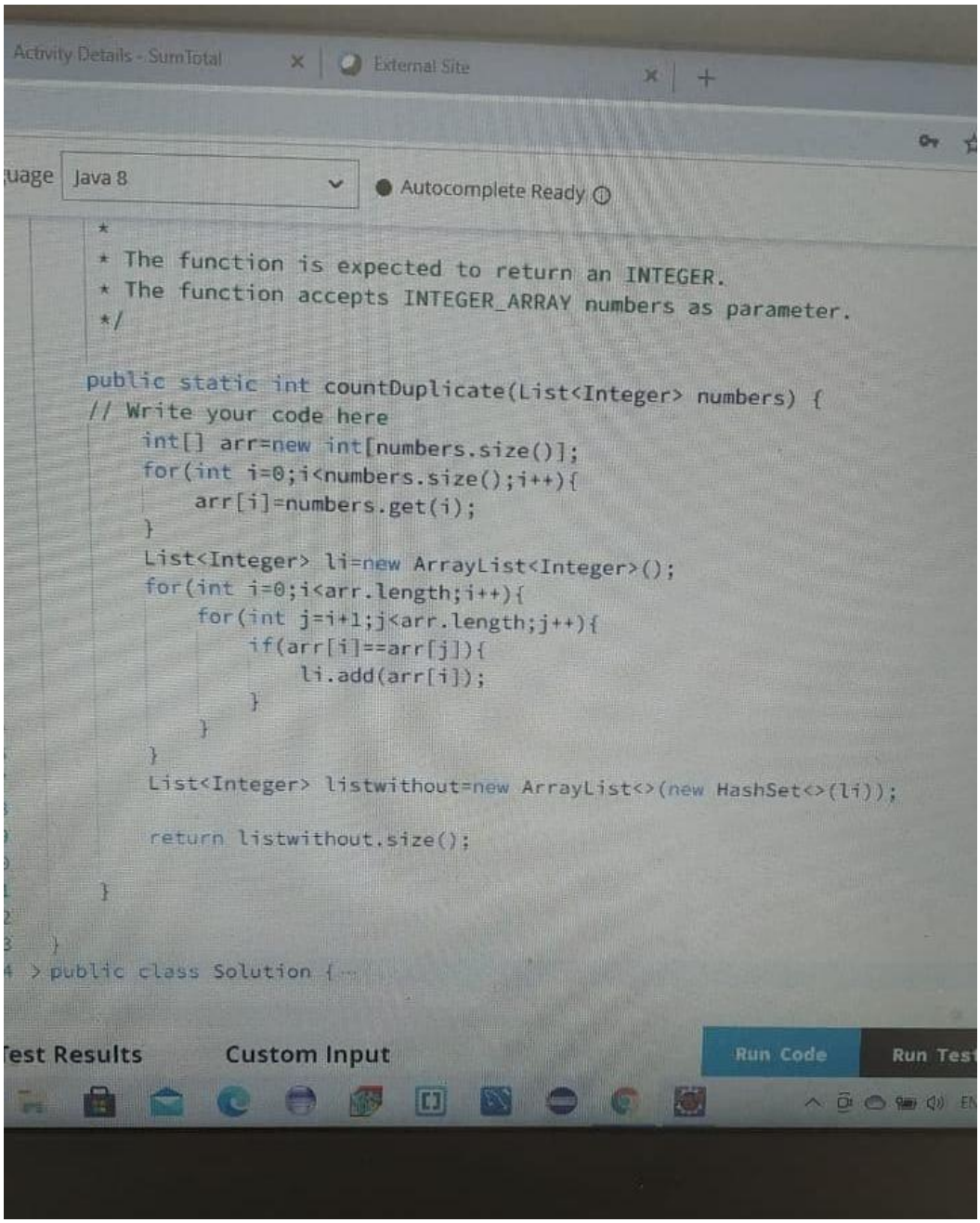




ARRANGING COINS



COUNT DUPLICATE ELEMENTS

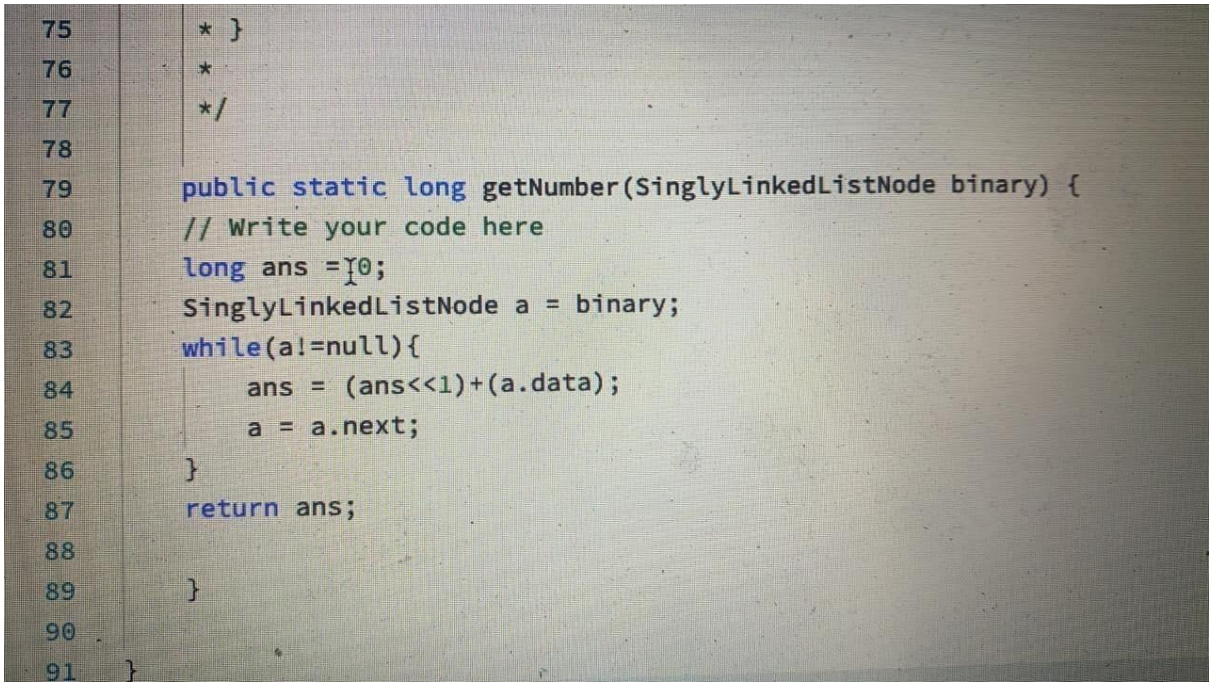


ANOTHER SOLUTION

```
public static int countDuplicate(List<int> numbers)
  {
    Dictionary<int,int> frequency = new Dictionary<int, int>();
    for(int i=0; i<numbers.Count;i++){
```

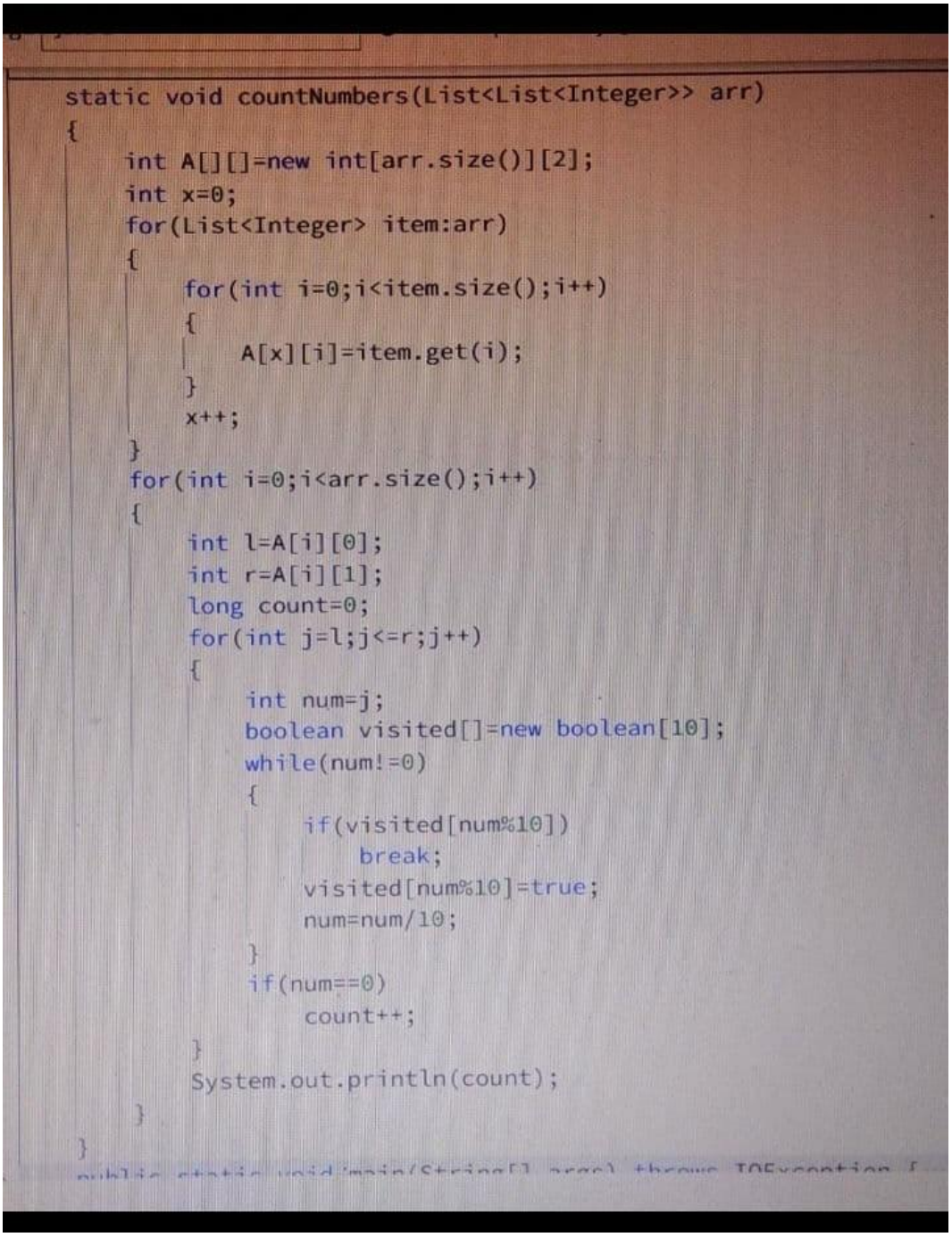
```
        if(frequency.ContainsKey(numbers[i])){
            frequency[numbers[i]] = frequency[numbers[i]]+1;
        }
        else{
            frequency.Add(numbers[i],1);
        }
    }
    List<int> result = new List<int>();
    foreach(var keys in frequency.Keys){
        if(frequency[keys]>1){
            result.Add(keys);
        }
    }
    return result.Count;
}
```

BINARY NUMBER IN A LINKED LIST



```
75      * }
76      *
77      */
78
79      public static long getNumber(SinglyLinkedListNode binary) {
80      // Write your code here
81      long ans =0;
82      SinglyLinkedListNode a = binary;
83      while(a!=null){
84          ans = (ans<<1)+(a.data);
85          a = a.next;
86      }
87      return ans;
88
89      }
90
91  }
```

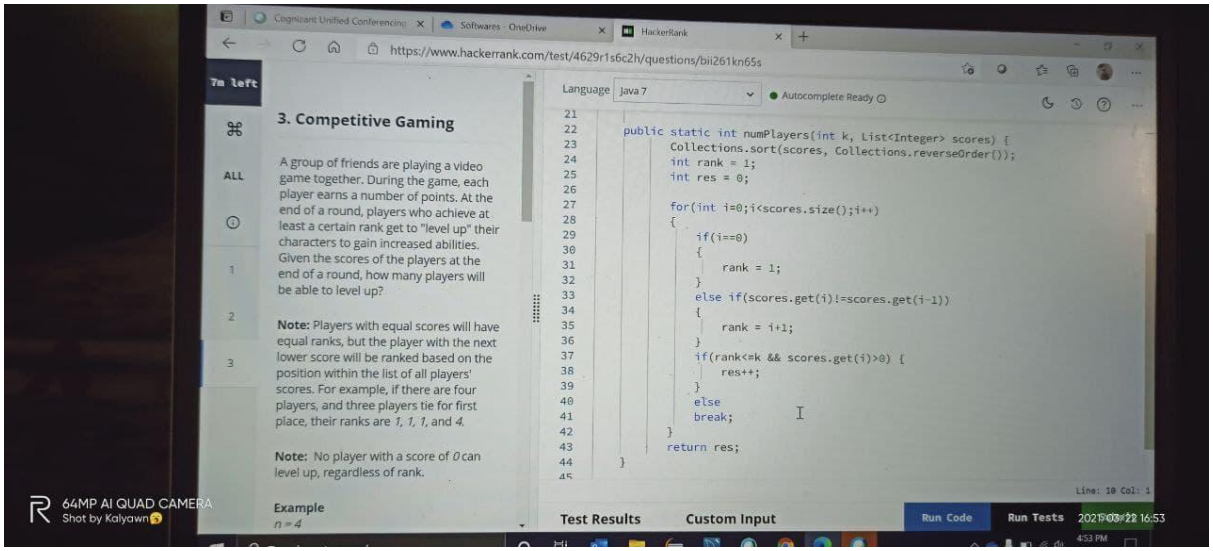
DISTINCT NUMBERS



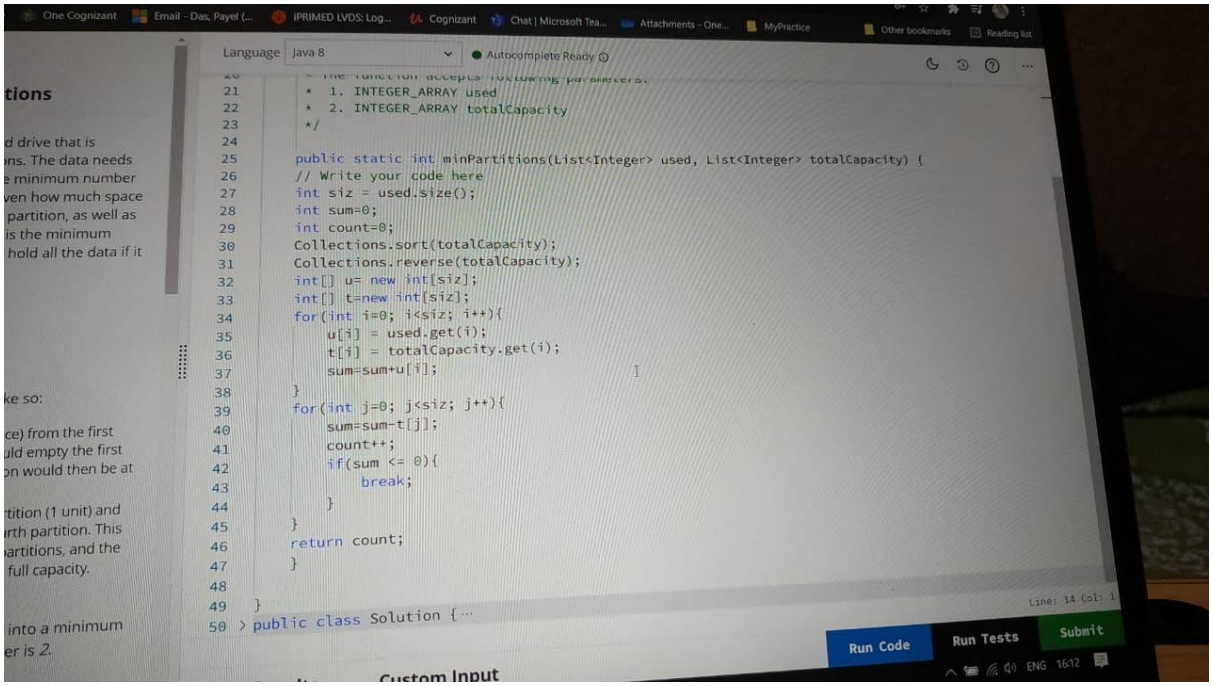
```
static void countNumbers(List<List<Integer>> arr)
{
    int A[][]=new int[arr.size()][2];
    int x=0;
    for(List<Integer> item:arr)
    {
        for(int i=0;i<item.size();i++)
        {
            A[x][i]=item.get(i);
        }
        x++;
    }
    for(int i=0;i<arr.size();i++)
    {
        int l=A[i][0];
        int r=A[i][1];
        long count=0;
        for(int j=l;j<=r;j++)
        {
            int num=j;
            boolean visited[]=new boolean[10];
            while(num!=0)
            {
                if(visited[num%10])
                    break;
                visited[num%10]=true;
                num=num/10;
            }
            if(num==0)
                count++;
        }
        System.out.println(count);
    }
}
```

```
25
26      public static int numPlayers(int k, List<Integer> scores) {
27      // Write your code here
28  if(k<=0)
29  return 0;
30  Collections.sort(scores, Collections.reverseOrder());
31  int rank=1;
32  int res=0;
33  for(int i=0;i<scores.size();i++){
34      if(i==0){
35          rank=1;
36      }
37      else if(scores.get(i)!=scores.get(i-1)){
38          rank=i+1;
39      }
40      if(rank<=k && scores.get(i)>0)
41      res++;
42      else break;
43  }
44  return res;
45      }
46
47  }
48
```

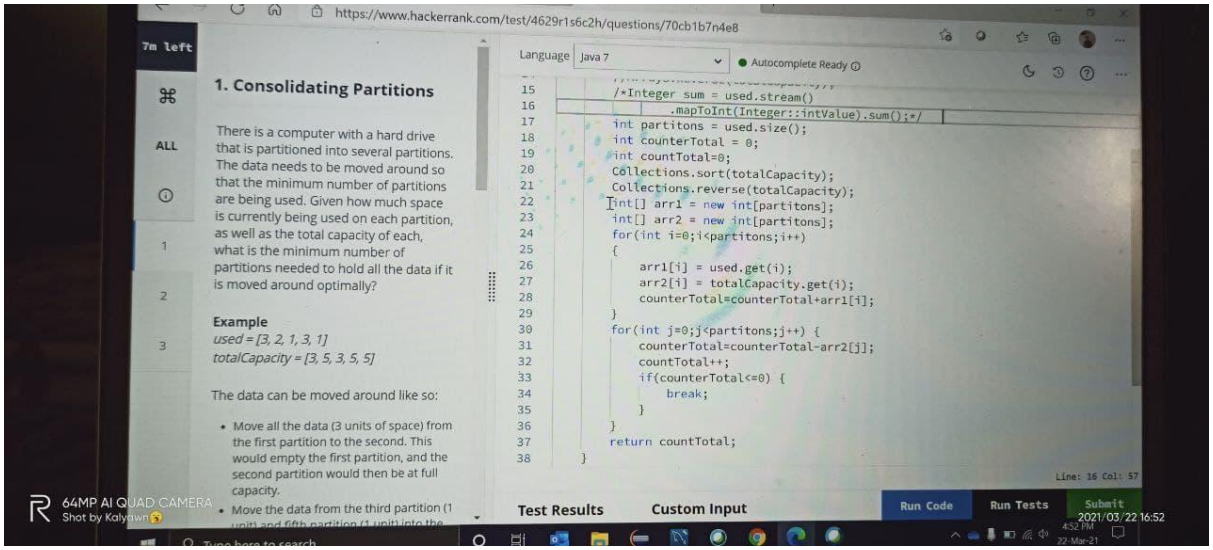
competetive gaming soln 2



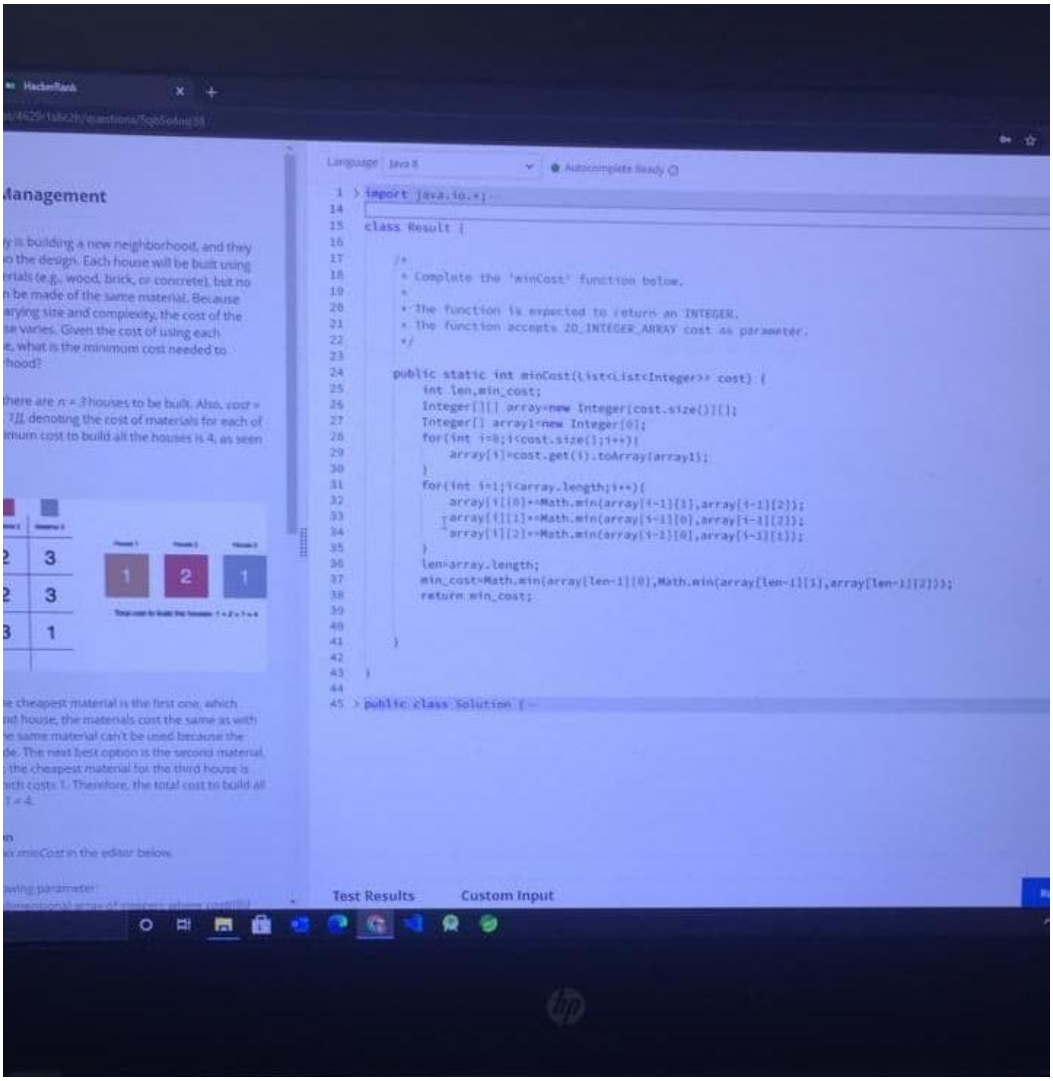
Consolidated partitions



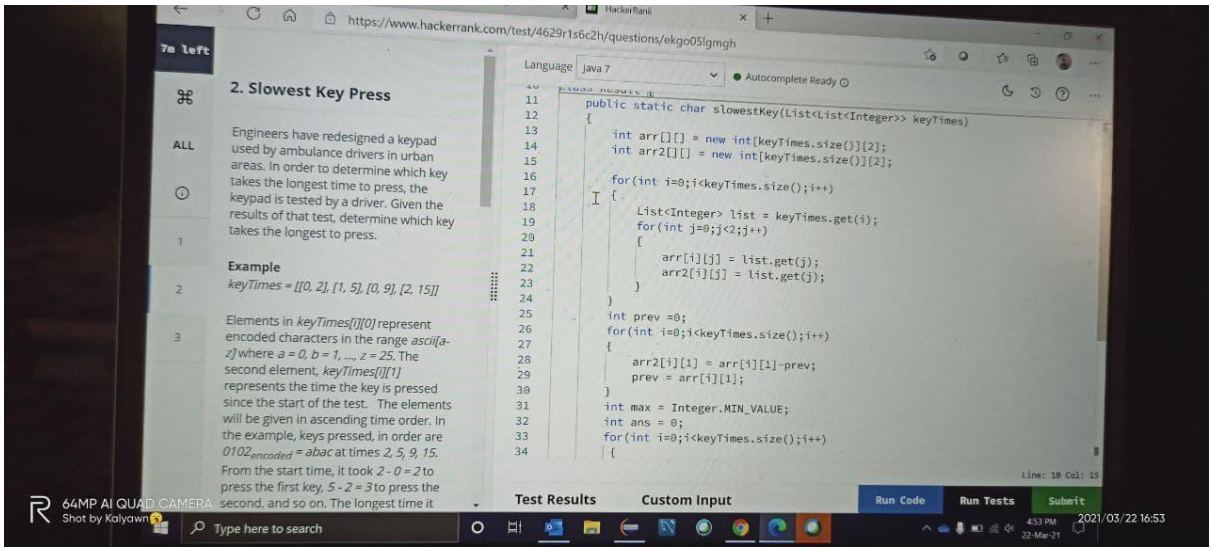
consolicadate partitions soln 2

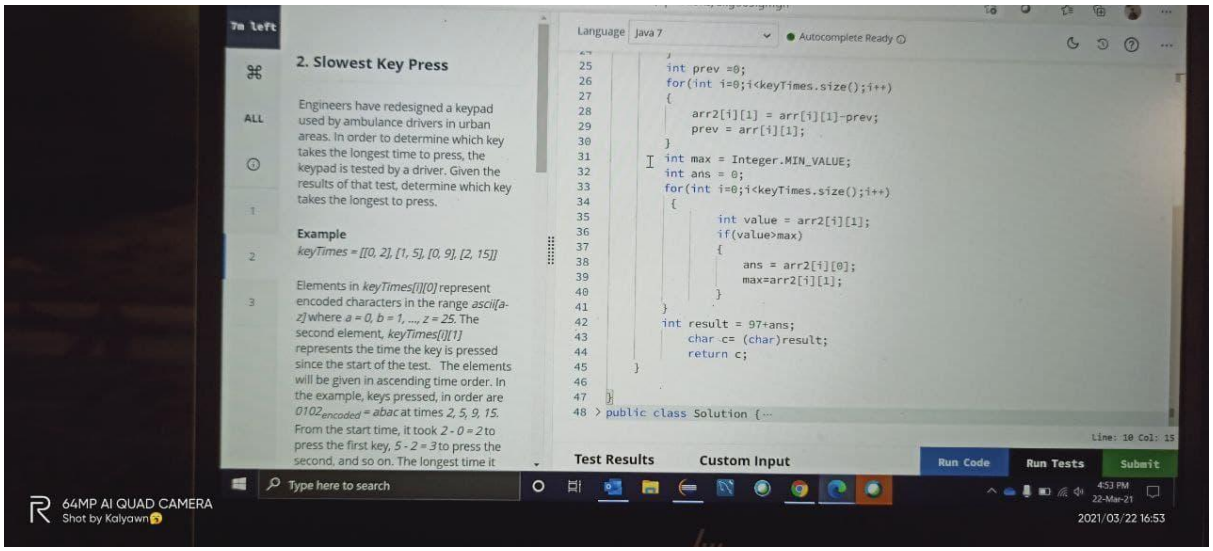


Construction management

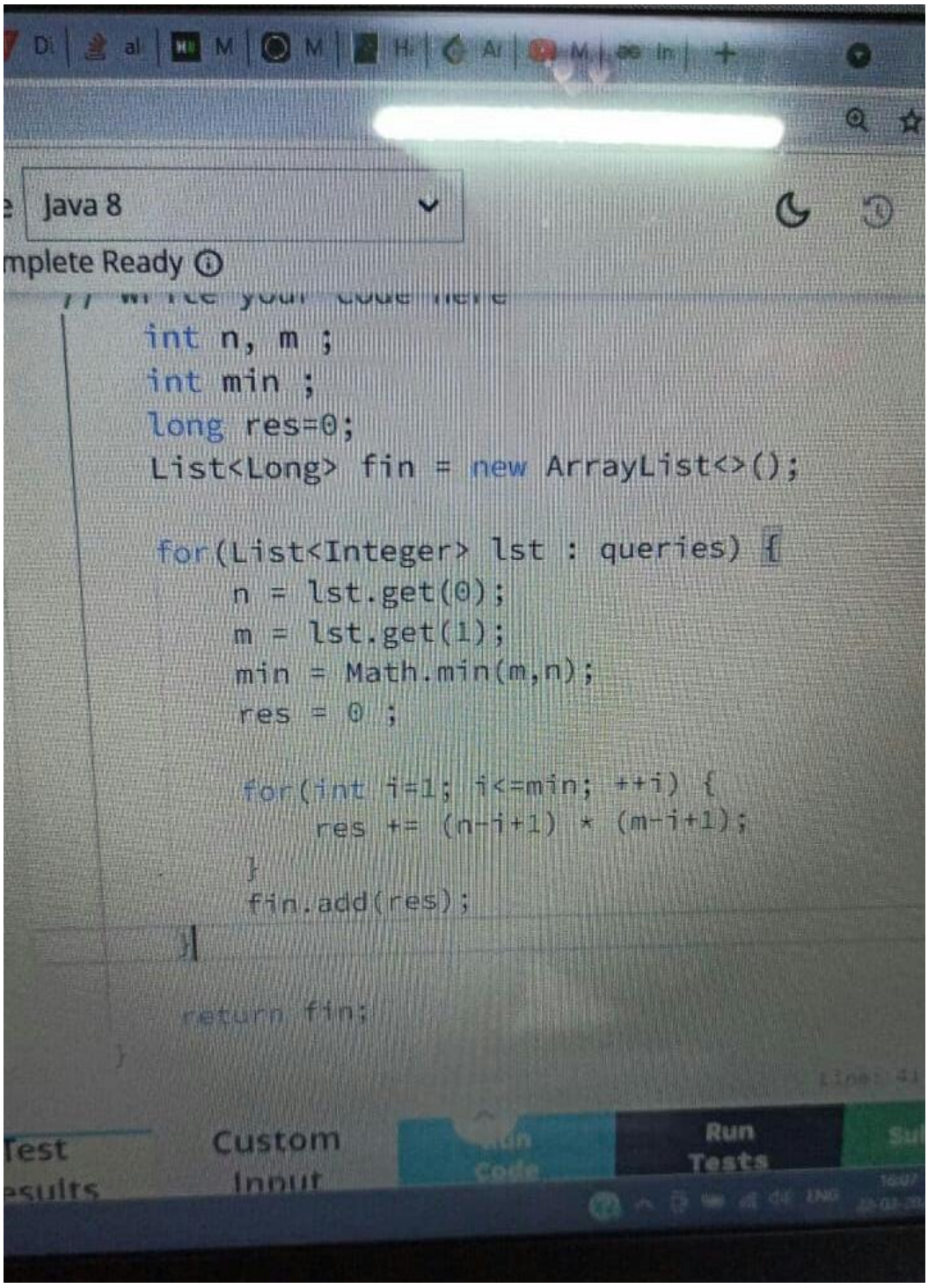


SLOWEST KEY PRESS


7m left
2. Slowest Key Press
Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.
Example
keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]
Elements in keyTimes[i][0] represent encoded characters in the range ascii[a-z] where a = 0, b = 1, ..., z = 25. The second element, keyTimes[i][1] represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are 0102encoded = abac at times 2, 5, 9, 15. From the start time, it took 2 - 0 = 2 to press the first key, 5 - 2 = 3 to press the second, and so on. The longest time it
Language Java 7
Autocomplete Ready
public static char slowestKey(List<List<Integer>> keyTimes)
{
int arr[][] = new int[keyTimes.size()][2];
int arr2[][] = new int[keyTimes.size()][2];
for(int i=0;i<keyTimes.size();i++)
{
List<Integer> list = keyTimes.get(i);
for(int j=0;j<2;j++)
{
arr[i][j] = list.get(j);
arr2[i][j] = list.get(j);
}
}
int prev =0;
for(int i=0;i<keyTimes.size();i++)
{
arr2[i][1] = arr[i][1]-prev;
prev = arr[i][1];
}
int max = Integer.MIN_VALUE;
int ans = 0;
for(int i=0;i<keyTimes.size();i++)
{
Line: 10 Col: 15
Test Results
Custom Input
Run Code
Run Tests
Submit
Type here to search
64MP AI QUAD CAMERA
Shot by Kalyawn
2021/03/22 16:53


7m left
2. Slowest Key Press
Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.
Example
keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]
Elements in keyTimes[i][0] represent encoded characters in the range ascii[a-z] where a = 0, b = 1, ..., z = 25. The second element, keyTimes[i][1] represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are 0102encoded = abac at times 2, 5, 9, 15. From the start time, it took 2 - 0 = 2 to press the first key, 5 - 2 = 3 to press the second, and so on. The longest time it
Language Java 7
Autocomplete Ready
int prev =0;
for(int i=0;i<keyTimes.size();i++)
{
arr2[i][1] = arr[i][1]-prev;
prev = arr[i][1];
}
int max = Integer.MIN_VALUE;
int ans = 0;
for(int i=0;i<keyTimes.size();i++)
{
int value = arr2[i][1];
if(value>max)
{
ans = arr2[i][0];
max=arr2[i][1];
}
}
int result = 97+ans;
char c= (char)result;
return c;
}
}
public class Solution {
Line: 10 Col: 15
Test Results
Custom Input
Run Code
Run Tests
Submit
Type here to search
64MP AI QUAD CAMERA
Shot by Kalyawn
2021/03/22 16:53

AREA OF THE BOX

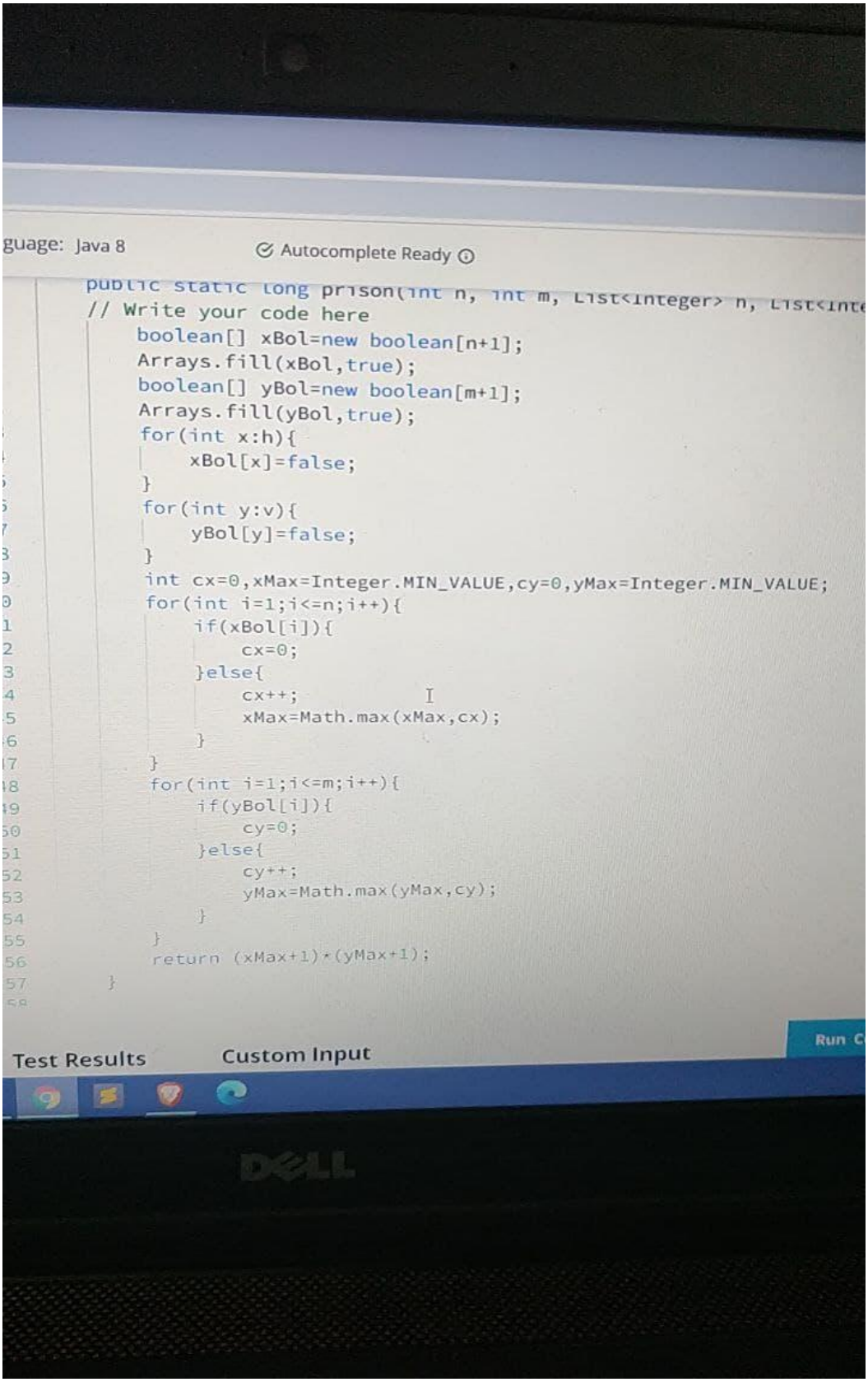


CAR INHERITANCE

```
class WagonR extends Car{
   int mileage;
   public WagonR(Integer mileage){
      super(false, "4");
      this.mileage = mileage;
   }

   @Override
   public String getMileage(){
      String mil=Integer.toString(mileage);
      return mileage+" kmpl";
   }
}


class HondaCity extends Car{
   int mileage;
   public HondaCity(Integer mileage){
     super(true, "4");
      this.mileage = mileage;

   }

   @Override
   public String getMileage(){
      String mil=Integer.toString(mileage);
      return mil+" kmpl";
   }
}
```

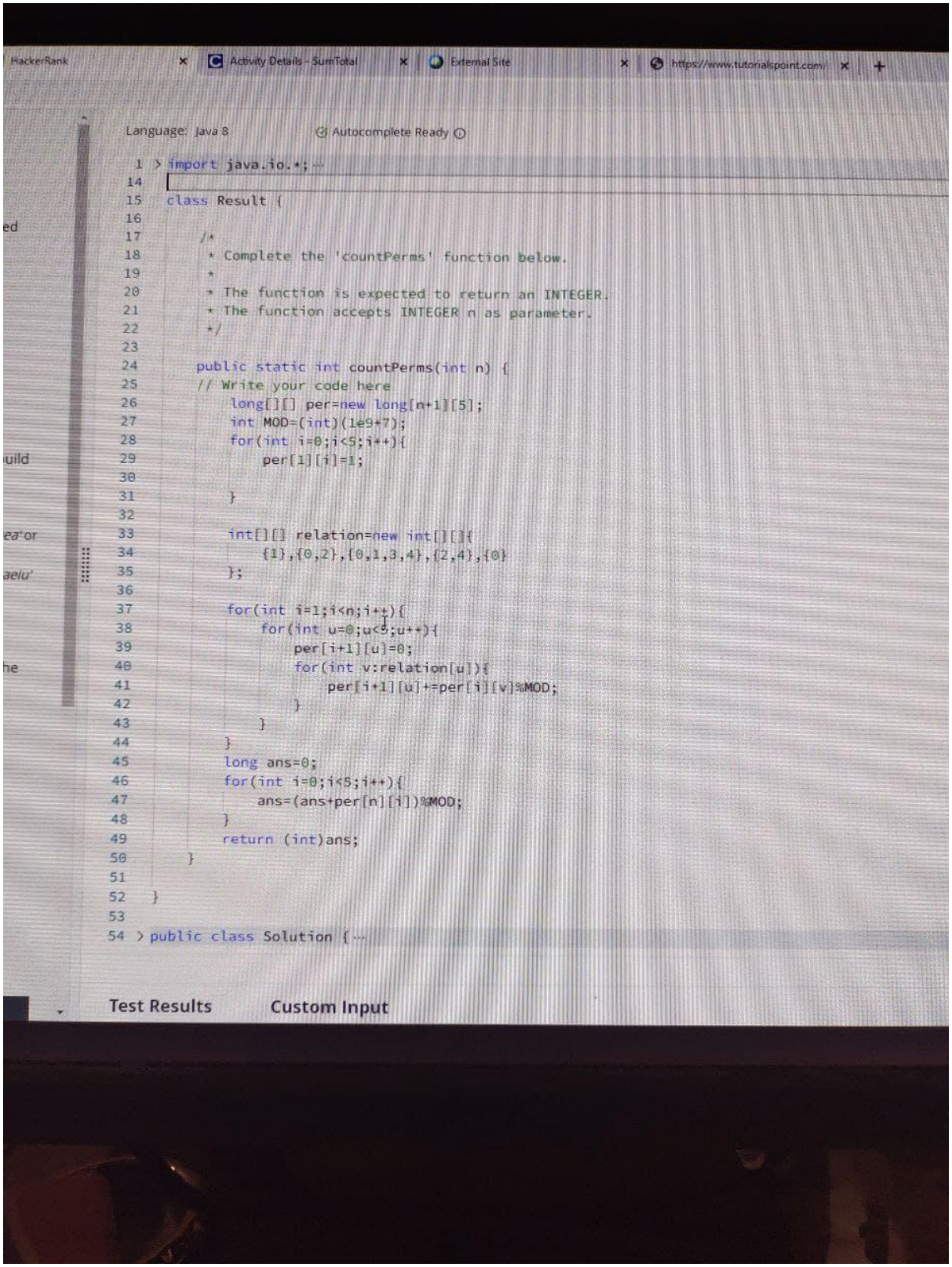
```
class InnovaCrysta extends Car{
    int mileage;
    public InnovaCrysta(Integer mileage){

       super(false, "6");
        this.mileage = mileage;

    }

    @Override
    public String getMileage(){
        String mil=Integer.toString(mileage);
        return mil+" kmpl";
    }
}
```

PRISON BREAK



Check this also: https://leetcode.com/discuss/interview-question/1002082/twillio-oa-prison-break

COUNT STRING PERMUTATIONS



LIST INHERITANCE

https://allhackerranksolutionsbykaira.blogspot.com/2020/08/list-inheritance-hackerrank-solution.html

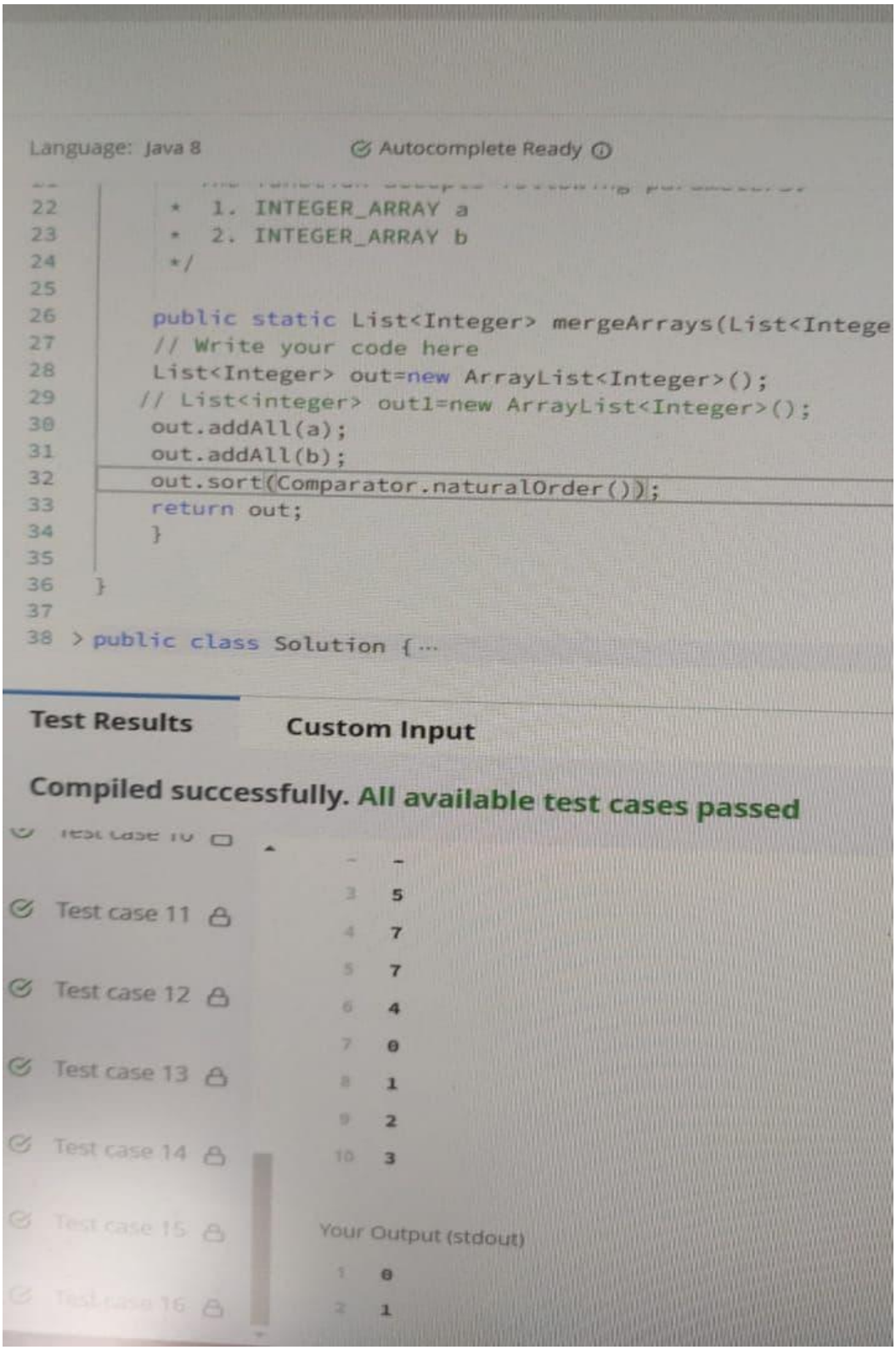
DEVICE NAME SYSTEM

https://www.homeworklib.com/question/1486956/in-java-6-device-name-system-suggested-problem

SPORT INHERITANCE

https://www.goeduhub.com/2885/program-inherit-cricketplayer-footballplayer-hockeyplayer

MERGE 2 ARRAYS



Language: Java 8 Autocomplete Ready

```
22      *  1. INTEGER_ARRAY a
23      *  2. INTEGER_ARRAY b
24      */
25
26      public static List<Integer> mergeArrays(List<Intege
27      // Write your code here
28      List<Integer> out=new ArrayList<Integer>();
29      // List<integer> out1=new ArrayList<Integer>();
30      out.addAll(a);
31      out.addAll(b);
32      out.sort(Comparator.naturalOrder());
33      return out;
34      }
35
36  }
37
38  > public class Solution {...
```

Test Results Custom Input

Compiled successfully. All available test cases passed

Test case 11

Test case 12

Test case 13

Test case 14

Test case 15

Test case 16

```
3  5
4  7
5  7
6  4
7  0
8  1
9  2
10 3
```

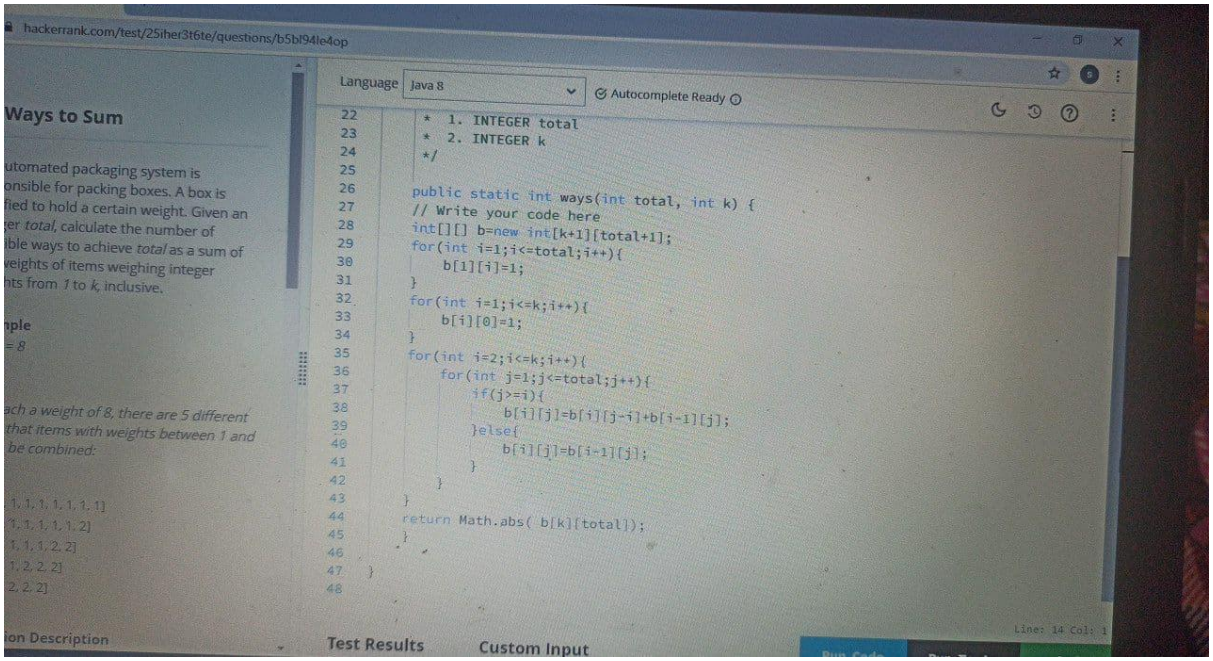
Your Output (stdout)

```
1  0
2  1
```

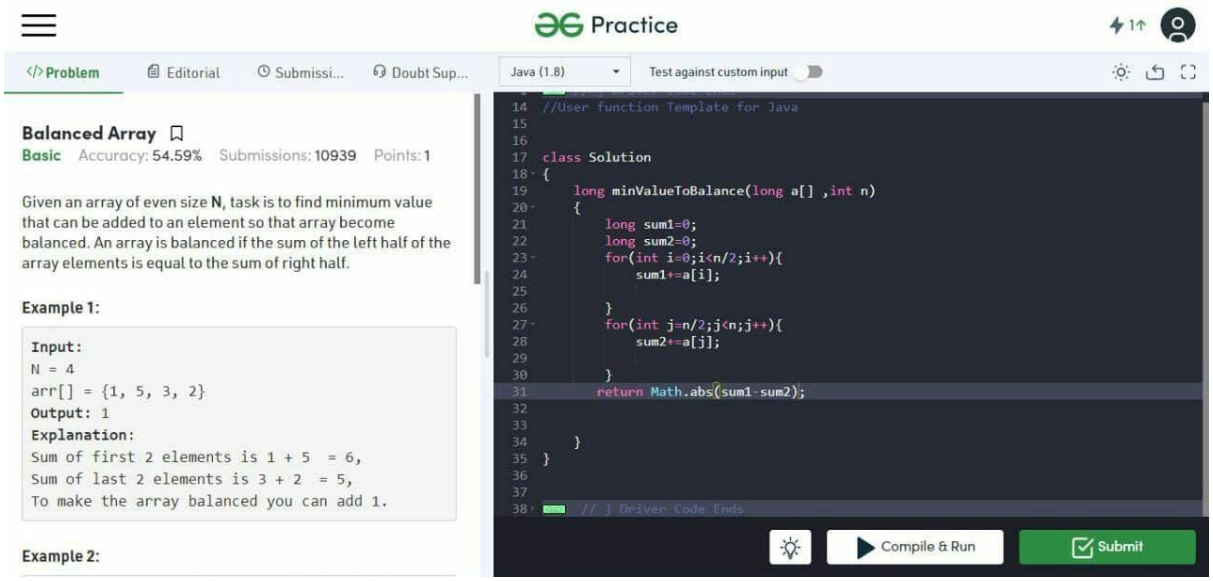
CONDENSED LIST

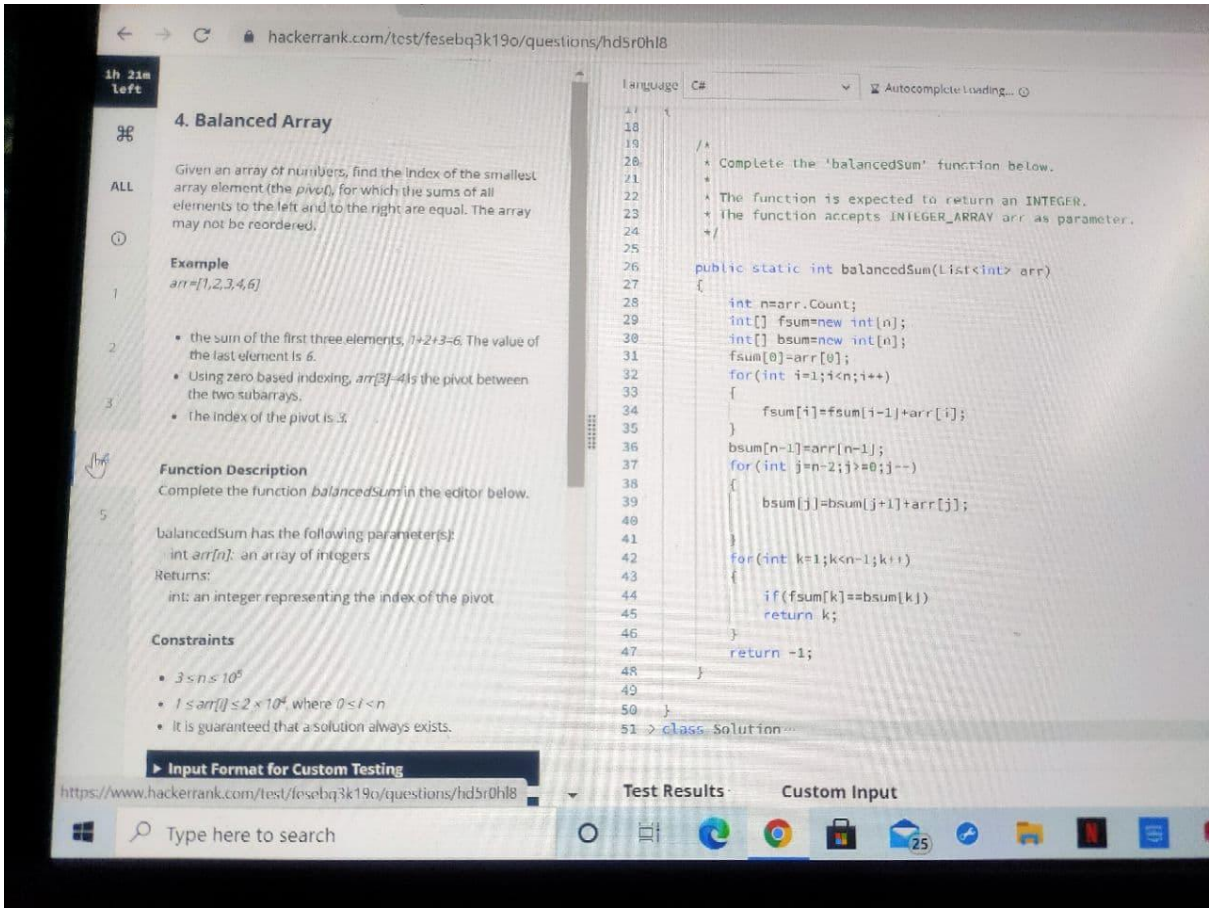
```
77      */
78
79      public static SinglyLinkedListNode condense(SinglyLinkedListNode head) {
80      // Write your code here
81      HashSet<Integer> hSet = new HashSet<>();
82      SinglyLinkedListNode ptr1;
83      SinglyLinkedListNode ptr2;
84      ptr1 = head;
85      ptr2 = null;
86      while(ptr1!=null){
87          int  val = ptr1.data;
88          if(hSet.contains(val))ptr2.next = ptr1.next;
89          else{
90              hSet.add(val);
91              ptr2 = ptr1;
92          }
93          ptr1 = ptr1.next;
94      }
95      return head;
96
```

WAYS TO SUM



BALANCED ARRAY

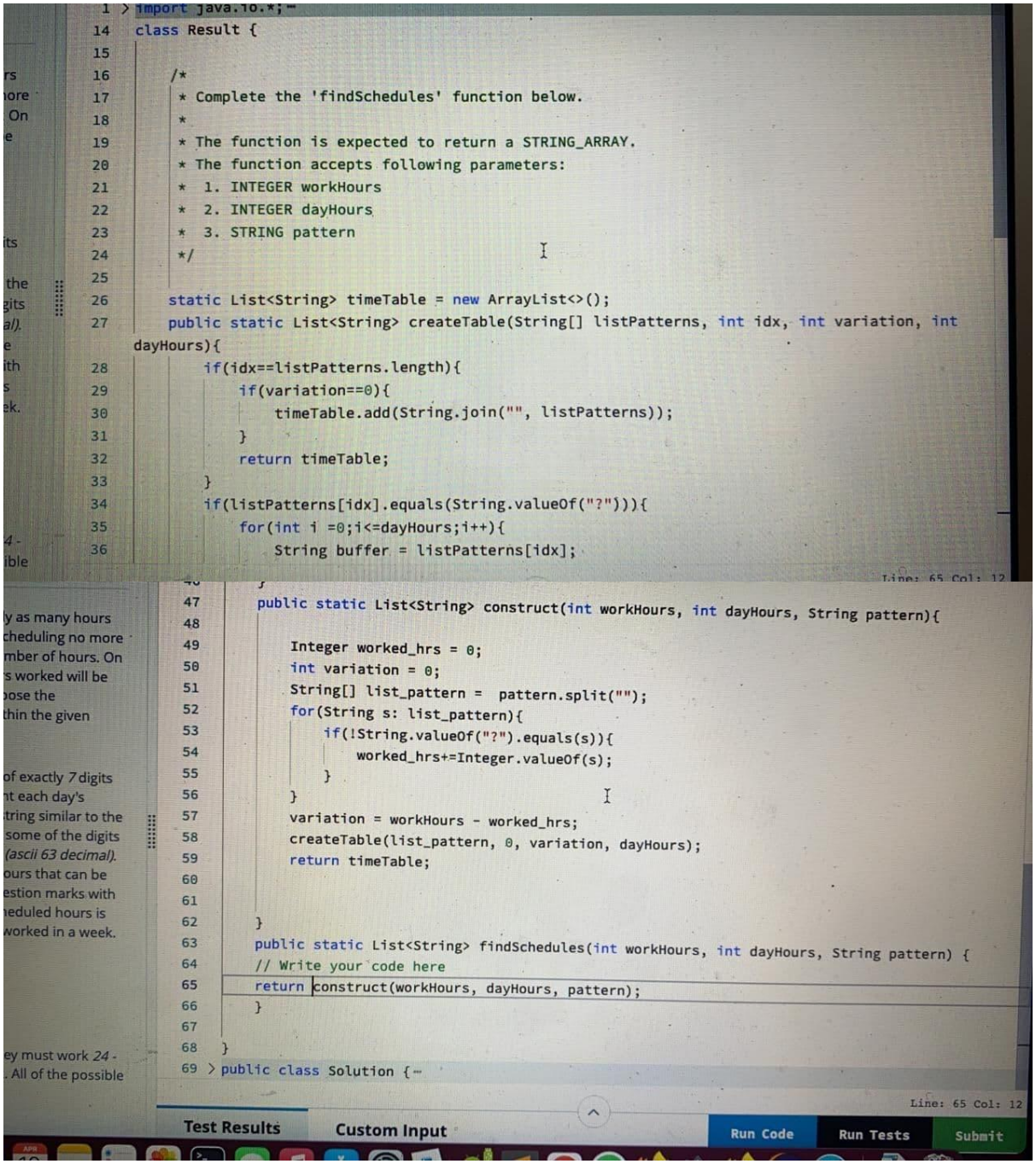




EMPLOYEE IMPLEMENTATION

https://github.com/vadym-usatiuk/Hackerrank-Employee-Implementation/blob/master/src/Solution.java

WORK SCHEDULE



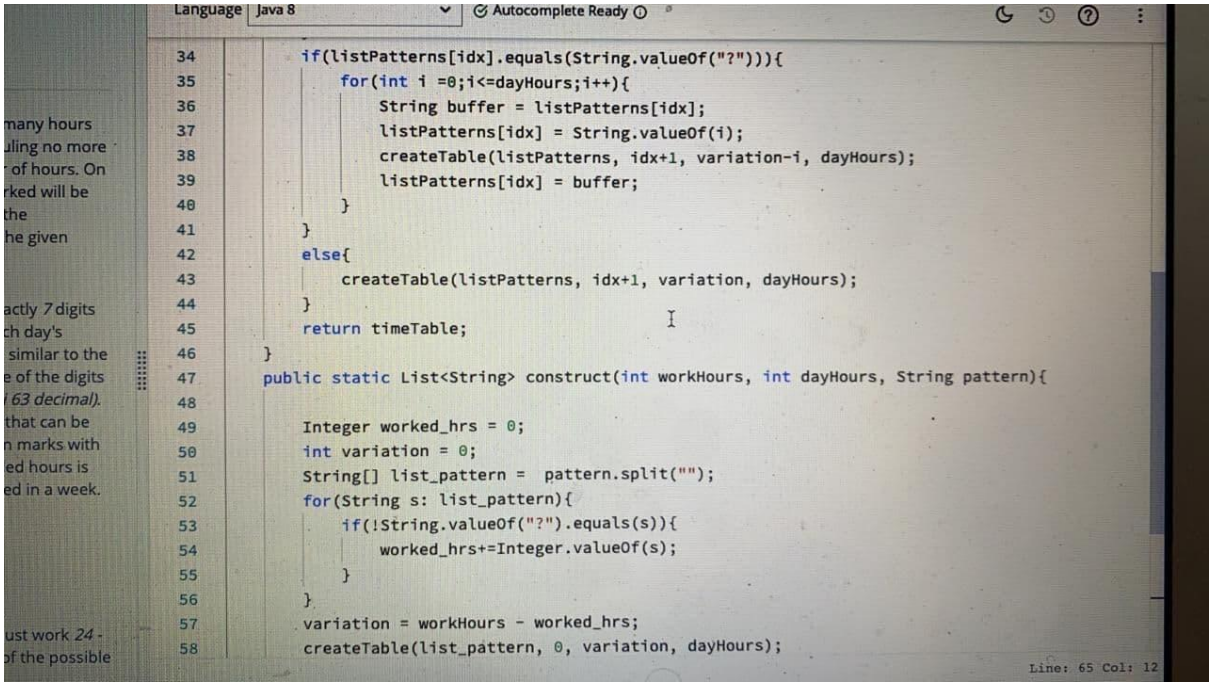
```
import java.io.*;
class Result {

    /*
     * Complete the 'findSchedules' function below.
     *
     * The function is expected to return a STRING_ARRAY.
     * The function accepts following parameters:
     *  1. INTEGER workHours
     *  2. INTEGER dayHours
     *  3. STRING pattern
     */

    static List<String> timeTable = new ArrayList<>();
    public static List<String> createTable(String[] listPatterns, int idx, int variation, int dayHours){
        if(idx==listPatterns.length){
            if(variation==0){
                timeTable.add(String.join("", listPatterns));
            }
            return timeTable;
        }
        if(listPatterns[idx].equals(String.valueOf("?"))){
            for(int i =0;i<=dayHours;i++){
                String buffer = listPatterns[idx];
```

Line: 65 Col: 12

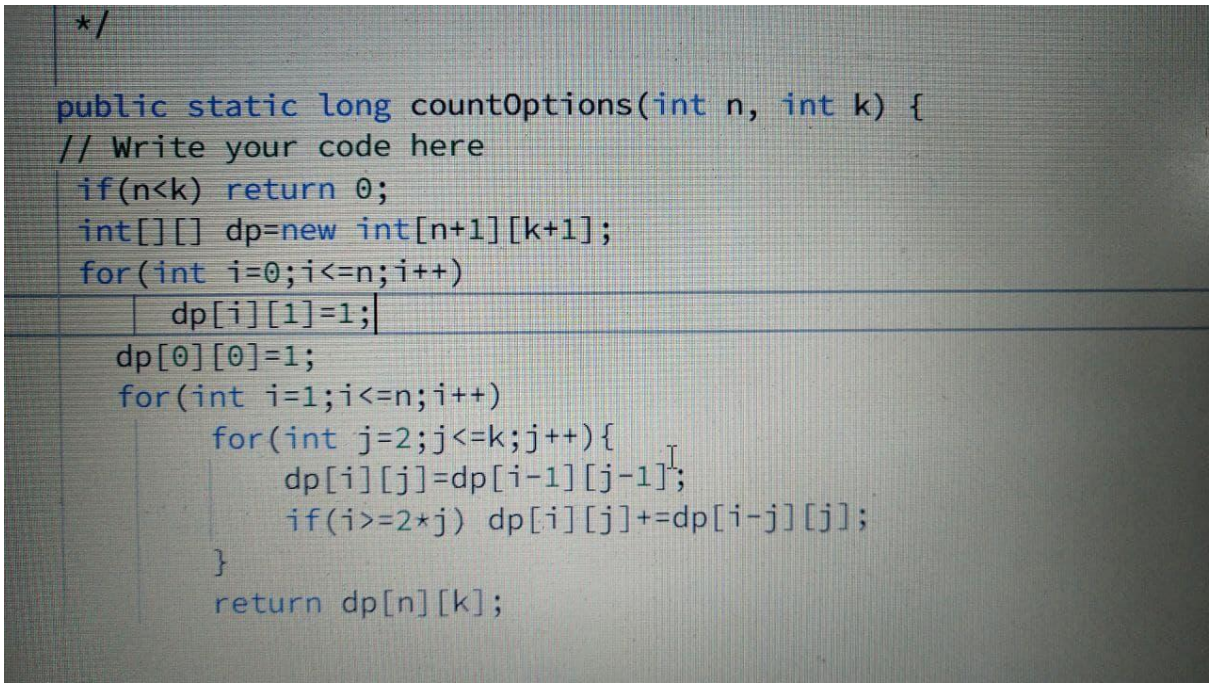
```
    public static List<String> construct(int workHours, int dayHours, String pattern){

        Integer worked_hrs = 0;
        int variation = 0;
        String[] list_pattern =  pattern.split("");
        for(String s: list_pattern){
            if(!String.valueOf("?").equals(s)){
                worked_hrs+=Integer.valueOf(s);
            }
        }
        variation = workHours - worked_hrs;
        createTable(list_pattern, 0, variation, dayHours);
        return timeTable;


    }
    public static List<String> findSchedules(int workHours, int dayHours, String pattern) {
    // Write your code here
    return construct(workHours, dayHours, pattern);
    }

}
public class Solution {
```

Line: 65 Col: 12

Test Results | Custom Input | Run Code | Run Tests | Submit



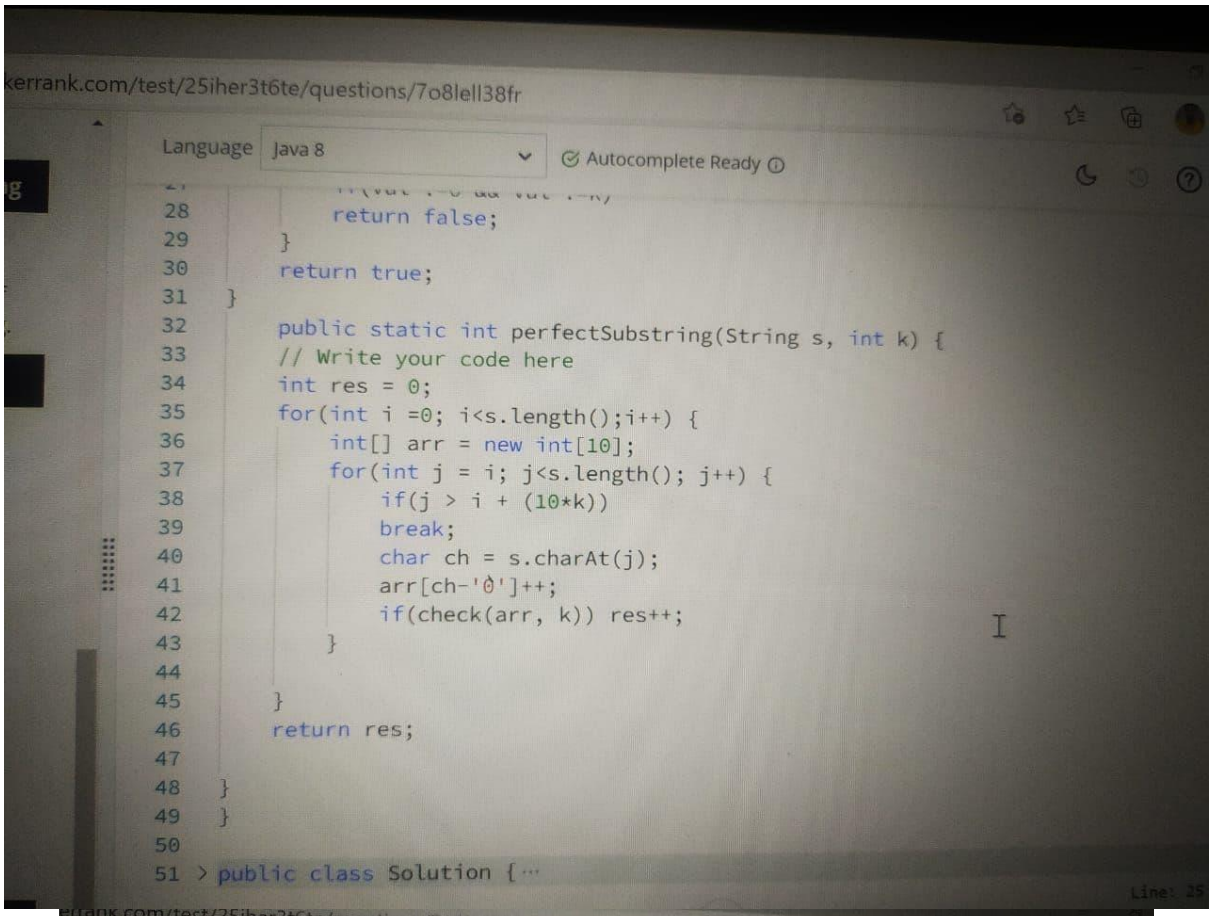
COUNT OPTIONS



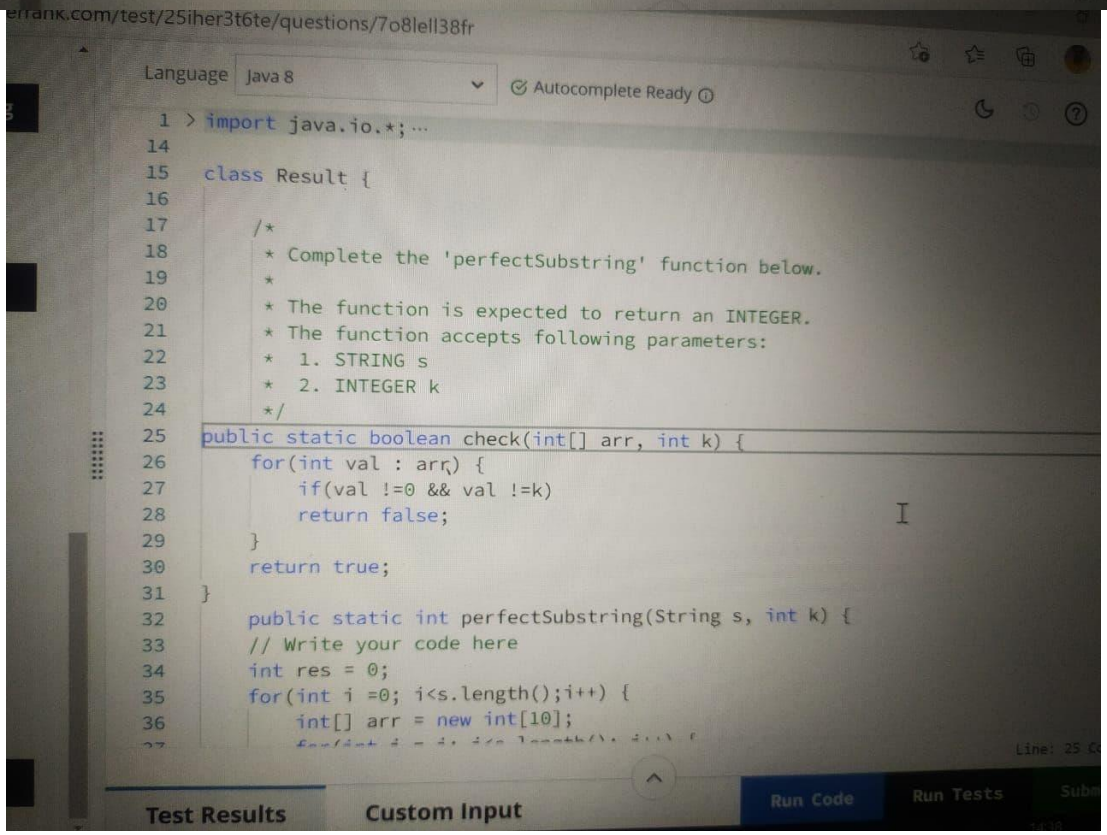
Count the number of ways to divide N in k s incrementally

https://www.geeksforgeeks.org/count-the-number-of-ways-to-divide-n-in-k-groups-incrementally/

PERFECT SUBSTRING



```
28          return false;
29      }
30      return true;
31  }
32      public static int perfectSubstring(String s, int k) {
33      // Write your code here
34      int res = 0;
35      for(int i =0; i<s.length();i++) {
36          int[] arr = new int[10];
37          for(int j = i; j<s.length(); j++) {
38              if(j > i + (10*k))
39              break;
40              char ch = s.charAt(j);
41              arr[ch-'0']++;
42              if(check(arr, k)) res++;
43          }
44
45      }
46      return res;
47
48  }
49  }
50
51  > public class Solution {···
```



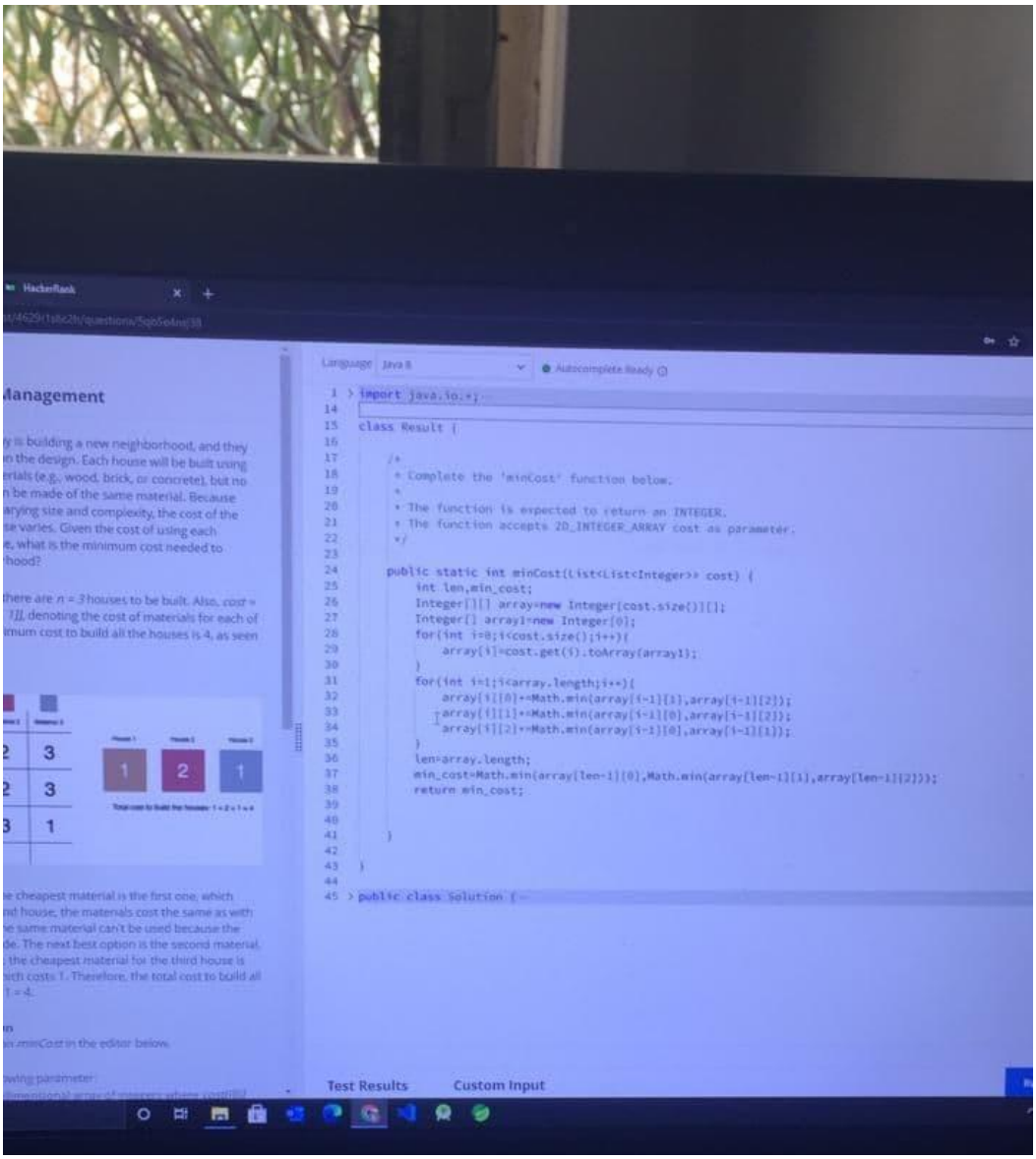
```
1  > import java.io.*;···
14
15  class Result {
16
17      /*
18       * Complete the 'perfectSubstring' function below.
19       *
20       * The function is expected to return an INTEGER.
21       * The function accepts following parameters:
22       *  1. STRING s
23       *  2. INTEGER k
24       */
25  public static boolean check(int[] arr, int k) {
26      for(int val : arr) {
27          if(val !=0 && val !=k)
28          return false;
29      }
30      return true;
31  }
32      public static int perfectSubstring(String s, int k) {
33      // Write your code here
34      int res = 0;
35      for(int i =0; i<s.length();i++) {
36          int[] arr = new int[10];
```

Test Results | Custom Input | Run Code | Run Tests

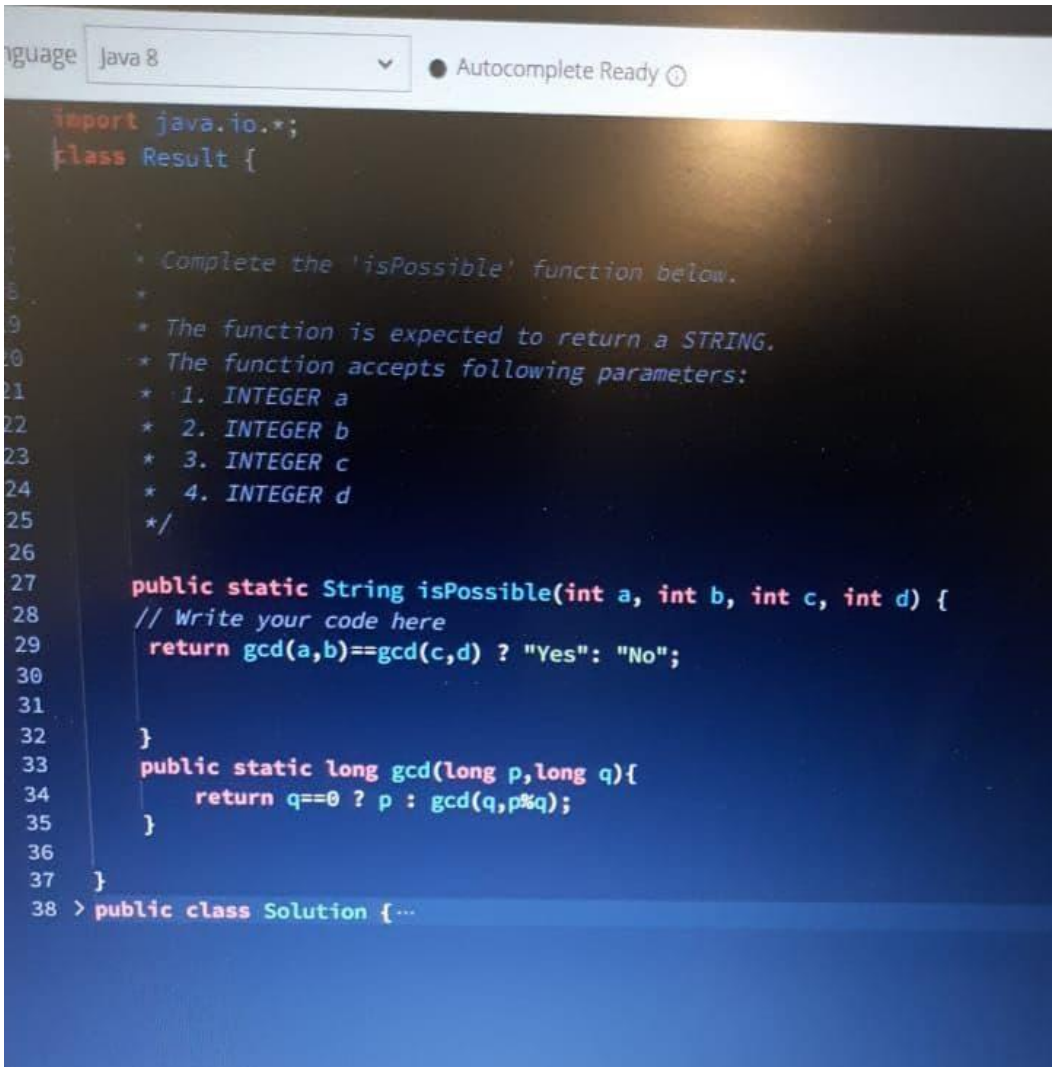
LOAD BALANCING

https://leetcode.com/discuss/interview-question/719447/wish-online-assessment-question

CONSTRUCTION MANAGEMENT



IS POSSIBLE

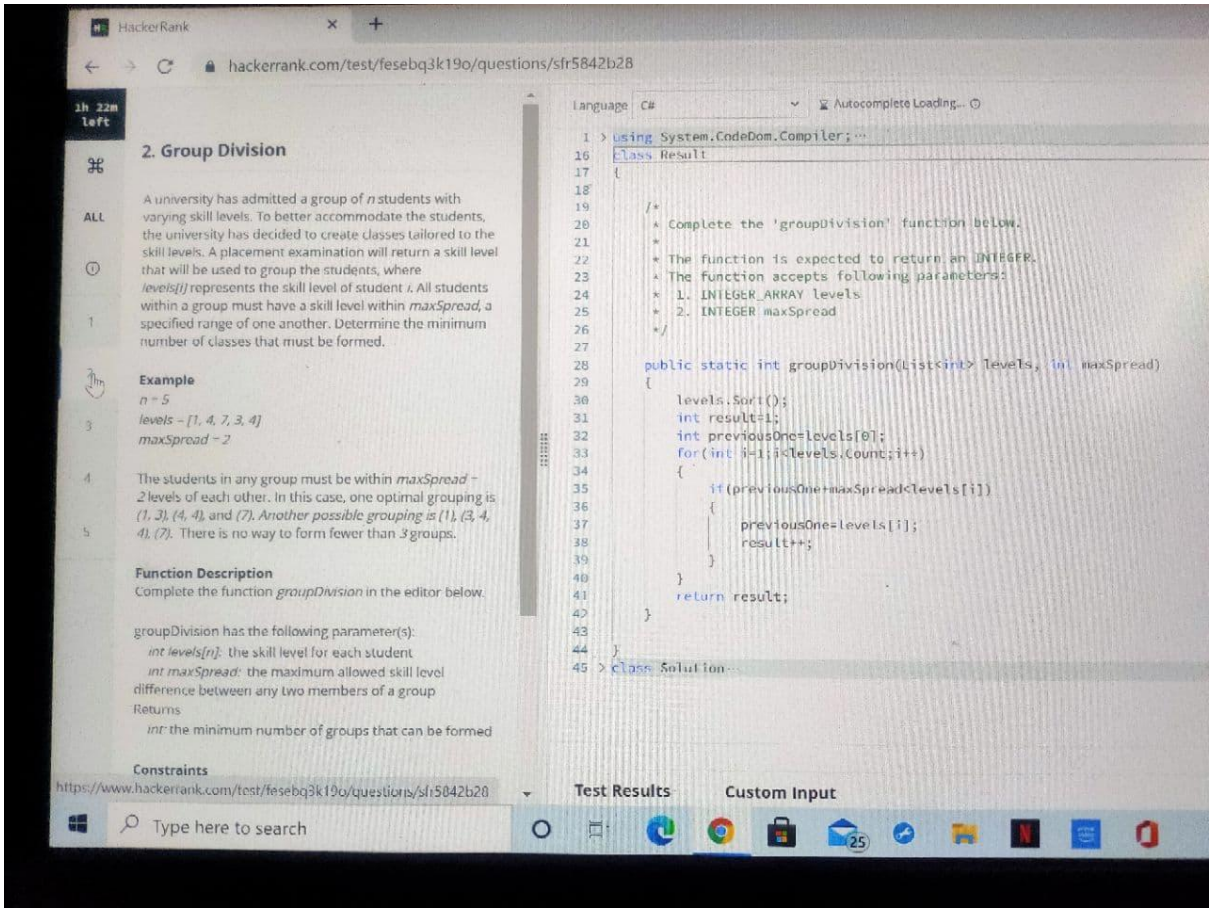


ANOTHER SOLUTION

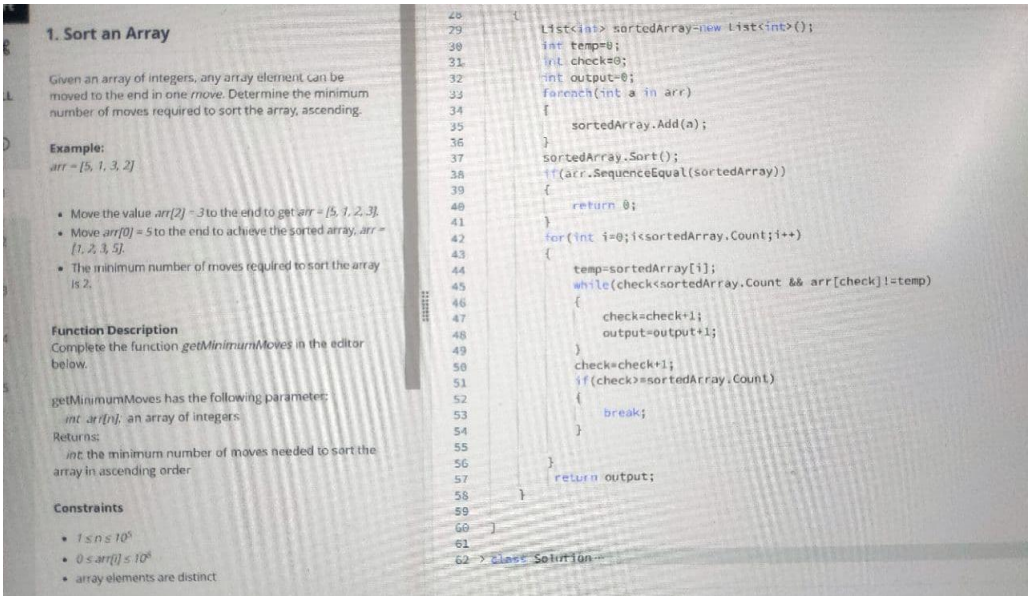
```
static LinkedList<Pair<Integer,Integer>> pairs = new LinkedList<Pair<Integer, Integer>>();


    public static String isItPossible(Integer a, Integer b, Integer c, Integer d){
        pairs.addLast(new Pair<Integer, Integer>(a,b));
        while (!pairs.isEmpty()){
            Pair<Integer,Integer> pair = pairs.poll();
            Integer key = pair.getKey();
            Integer value = pair.getValue();
```

```
        if(key.equals(c) &&
                value.equals(d)){
            return "YES";
        }
        int sum=key+value;
        if (sum<=c){
            pairs.addLast(new Pair<Integer, Integer>(sum,value));
        }
        if (sum<=d){
            pairs.addLast(new Pair<Integer, Integer>(key,sum));
        }

    }

    return "NO";
}
```

GROUP DIVISION



SORT AN ARRAY



1. Sort an Array

Given an array of integers, any array element can be moved to the end in one *move*. Determine the minimum number of moves required to sort the array, ascending.

Example:
arr = [5, 1, 3, 2]

- Move the value *arr[2] = 3* to the end to get *arr = [5, 1, 2, 3]*.
- Move *arr[0] = 5* to the end to achieve the sorted array, *arr = [1, 2, 3, 5]*.
- The minimum number of moves required to sort the array is 2.

Function Description
Complete the function *getMinimumMoves* in the editor below.

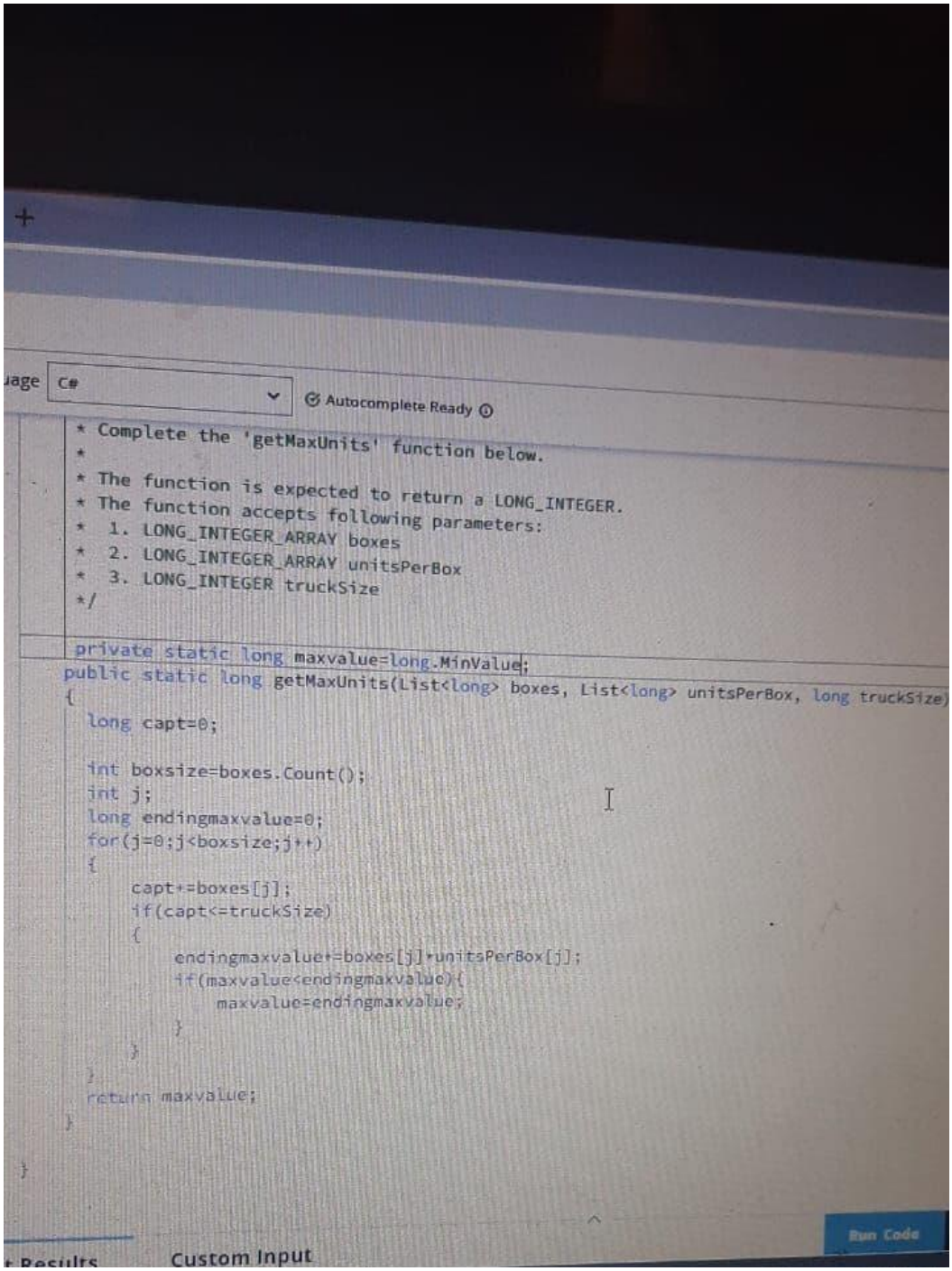
getMinimumMoves has the following parameter:
int arr[n]: an array of integers
Returns:
int: the minimum number of moves needed to sort the array in ascending order

Constraints

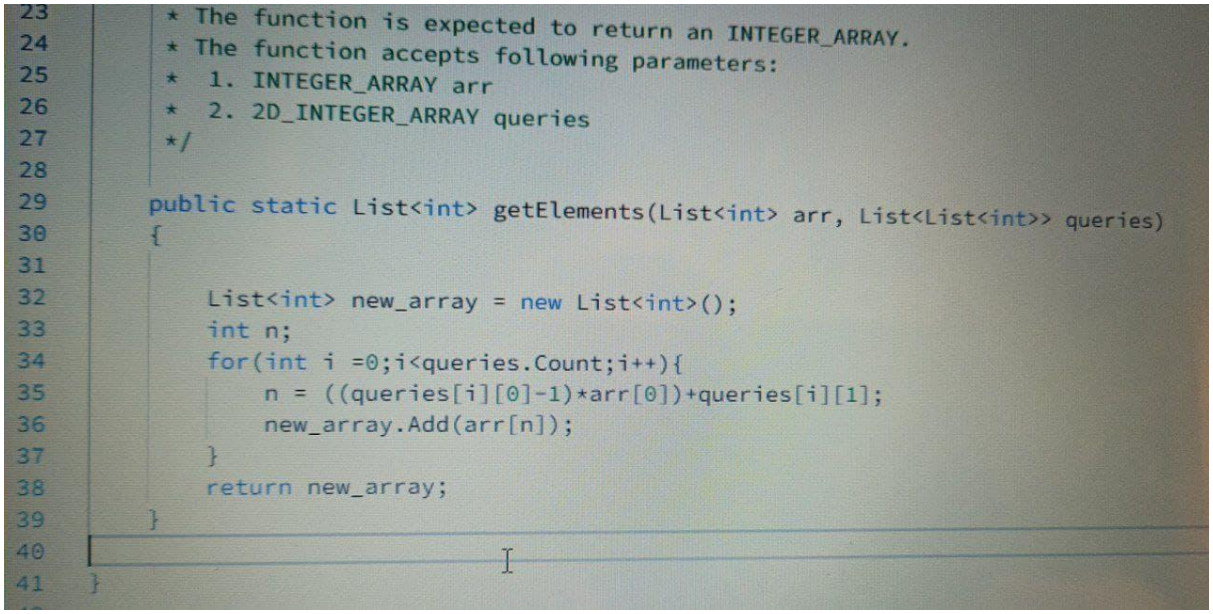
- $1 \leq n \leq 10^5$
- $0 \leq arr[i] \leq 10^6$
- array elements are distinct

```
78      {
79          List<int> sortedArray=new List<int>();
80          int temp=0;
81          int check=0;
82          int output=0;
83          foreach(int a in arr)
84          {
85              sortedArray.Add(a);
86          }
87          sortedArray.Sort();
88          if(arr.SequenceEqual(sortedArray))
89          {
90              return 0;
91          }
92          for(int i=0;i<sortedArray.Count;i++)
93          {
94              temp=sortedArray[i];
95              while(check<sortedArray.Count && arr[check]!=temp)
96              {
97                  check=check+1;
98                  output=output+1;
99              }
100             check=check+1;
101             if(check>=sortedArray.Count)
102             {
103                 break;
104             }
105
106         }
107         return output;
108     }
109
110 }
111
112 > class Solution ...
```

EFFICIENT SHIPPING

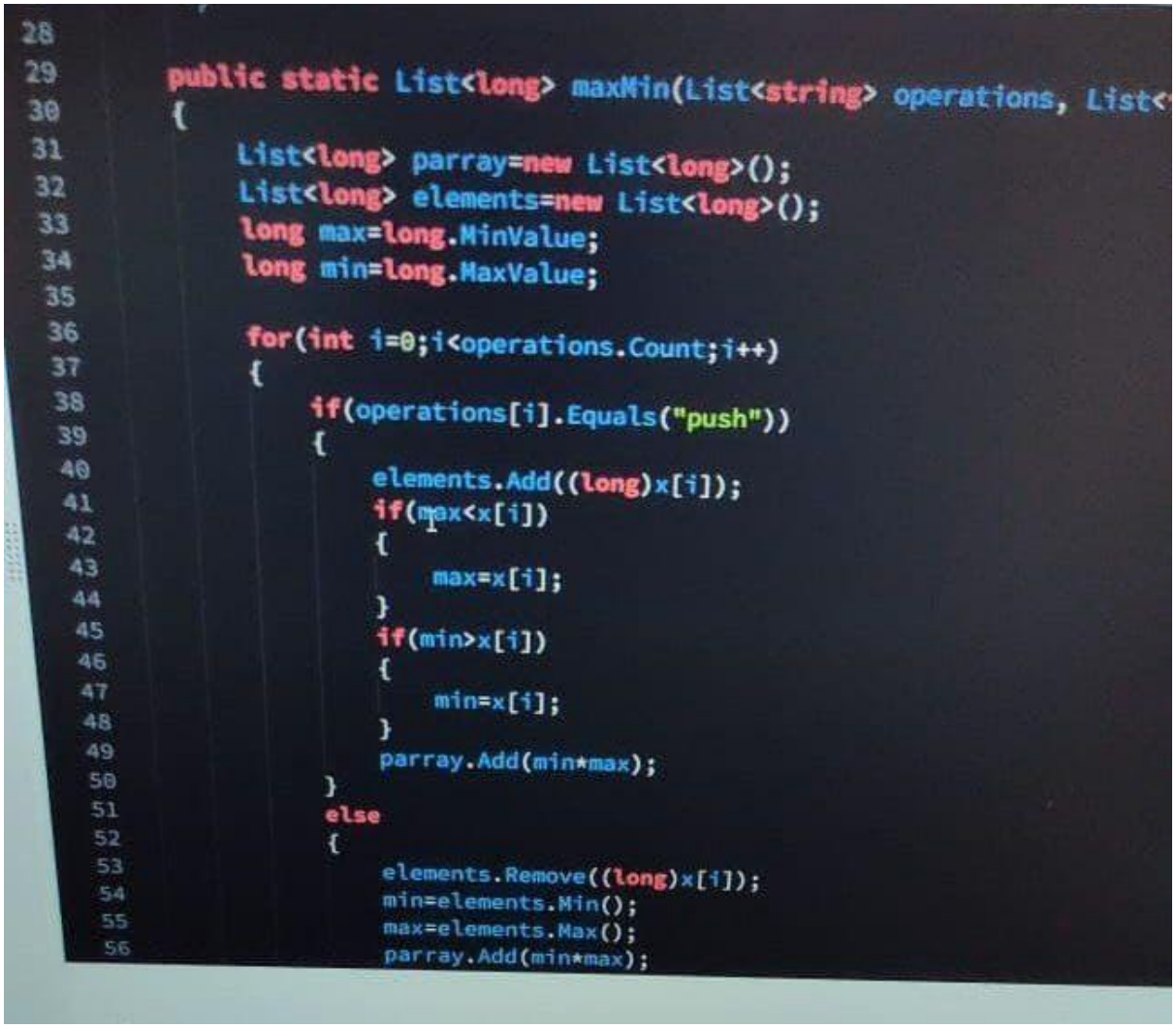


THE RESTRUCTURED ARRAY

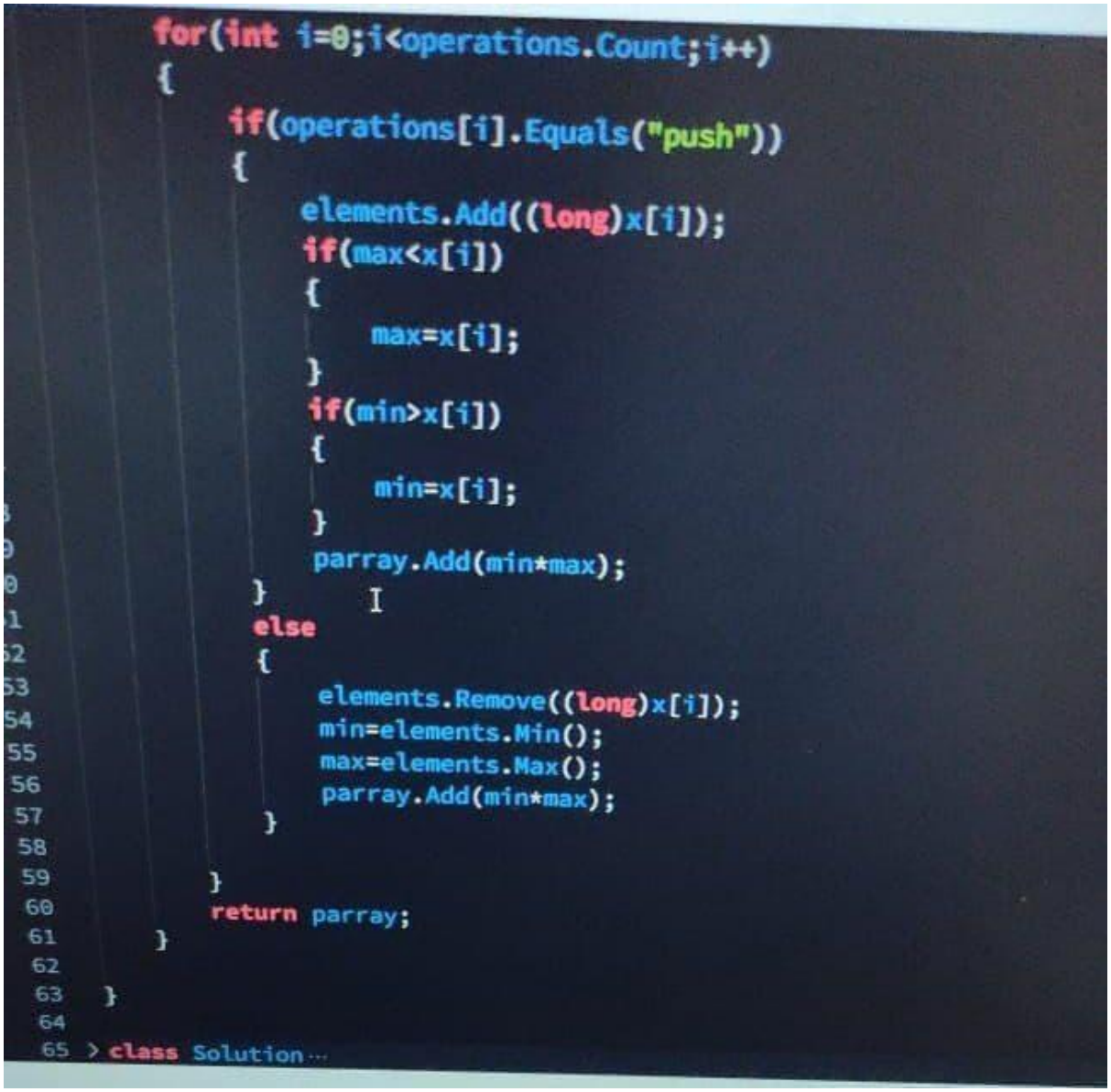


```
23      * The function is expected to return an INTEGER_ARRAY.
24      * The function accepts following parameters:
25      *  1. INTEGER_ARRAY arr
26      *  2. 2D_INTEGER_ARRAY queries
27      */
28
29     public static List<int> getElements(List<int> arr, List<List<int>> queries)
30     {
31
32         List<int> new_array = new List<int>();
33         int n;
34         for(int i =0;i<queries.Count;i++){
35             n = ((queries[i][0]-1)*arr[0])+queries[i][1];
36             new_array.Add(arr[n]);
37         }
38         return new_array;
39     }
40
41 }
```

PRODUCT OF MAXIMUM AND MINIMUM IN A DATA SET

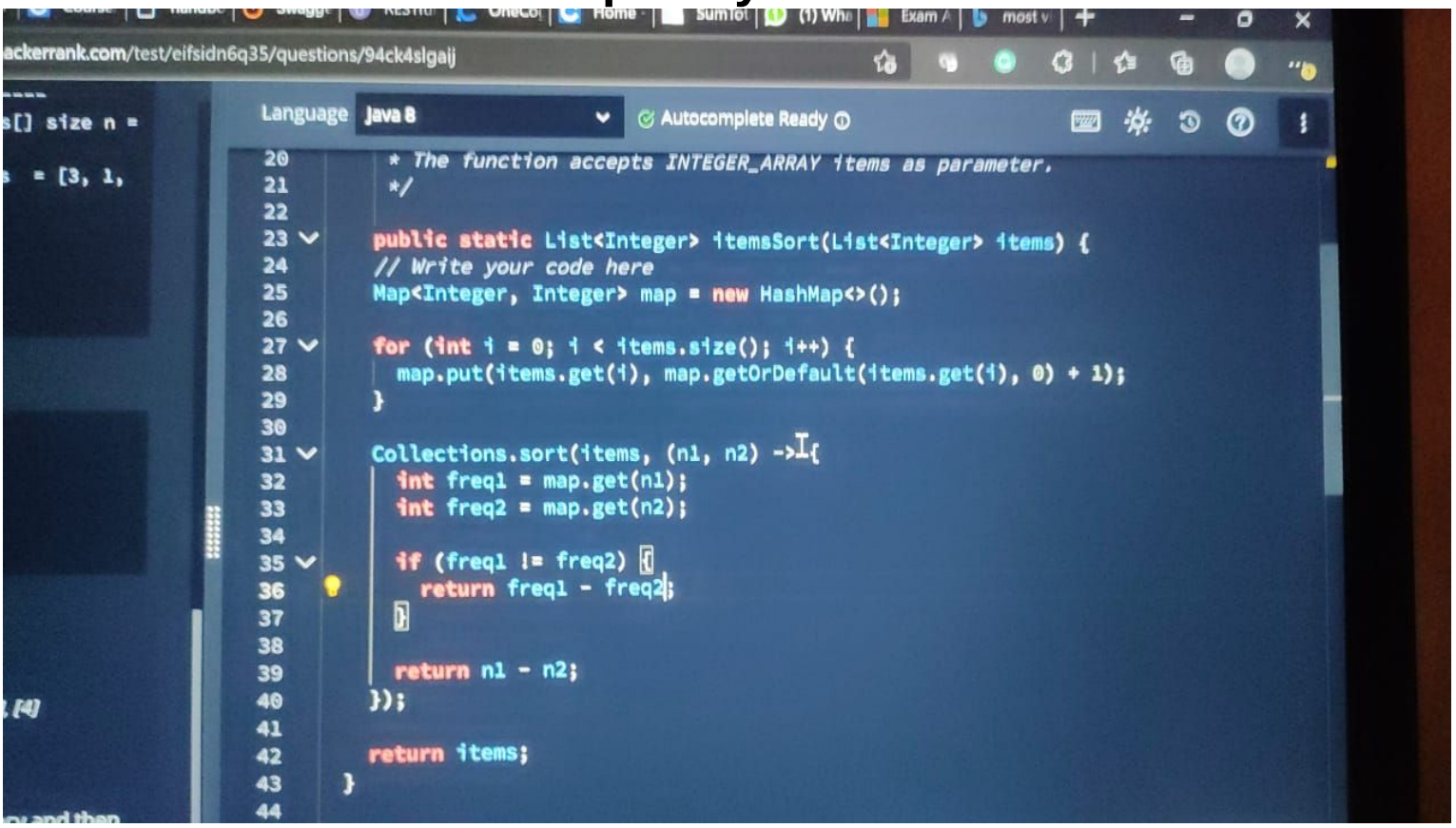


```
28
29      public static List<long> maxMin(List<string> operations, List<
30      {
31          List<long> parray=new List<long>();
32          List<long> elements=new List<long>();
33          long max=long.MinValue;
34          long min=long.MaxValue;
35
36          for(int i=0;i<operations.Count;i++)
37          {
38              if(operations[i].Equals("push"))
39              {
40                  elements.Add((long)x[i]);
41                  if(max<x[i])
42                  {
43                      max=x[i];
44                  }
45                  if(min>x[i])
46                  {
47                      min=x[i];
48                  }
49                  parray.Add(min*max);
50              }
51              else
52              {
53                  elements.Remove((long)x[i]);
54                  min=elements.Min();
55                  max=elements.Max();
56                  parray.Add(min*max);
```



```
        for(int i=0;i<operations.Count;i++)
        {
            if(operations[i].Equals("push"))
            {
                elements.Add((long)x[i]);
                if(max<x[i])
                {
                    max=x[i];
                }
                if(min>x[i])
                {
                    min=x[i];
                }
                parray.Add(min*max);
            }
            else
            {
                elements.Remove((long)x[i]);
                min=elements.Min();
                max=elements.Max();
                parray.Add(min*max);
            }

        }
        return parray;
    }

}
> class Solution...
```

Frequency sort



Twin Strings

```
public static List<string> twins(List<string> a, List<string> b)
    {
        var res = new string[a.Count];
        for(var i=-1; ++i <a.Count;)
        {
            string aitem = a[i],bitem=b[i];
            var twinis = aitem.Length==bitem.Length;
            if(twinis)
            {
                var tomap = new int['z' - 'a' +1,2];
                for(var j=-1; ++j<bitem.Length;)
                {
                    ++tomap[bitem[j] -'a',j&1];
                }
                for(var j=-1; ++j<bitem.Length;)
                {
                    if(--tomap[aitem[j]-'a',j&1]<0)
                    {
                        twinis = false;
                        break;
                    }
                }
            }
            res[i]=twinis ? "Yes" : "No";
        }
        return res.ToList();

    }
```

Competitve gaming

```
public static int numPlayers(int k, List<int> scores)
 {
     var ranks = new List<int>();
     var groups = scores.Where(f => f != 0).OrderByDescending(f => f).GroupBy(g => g).ToList();
     var rank = 1;
     foreach (var g in groups)
     {
         var count = g.Count();
         for (var j = 0; j < count; j++)
         {
             ranks.Add(rank);
         }
         rank = rank + count;
     }
     return ranks.Count(r => r <= k);
 }
```

1.construction management

```
  public static int minCost(List<List<int>> cost)
     {
         for(int i=1;i<cost.Count;i++){
         cost[i][0]+=Math.Min(cost[i-1][1],cost[i-1][2]);
         cost[i][1]+=Math.Min(cost[i-1][0],cost[i-1][2]);
         cost[i][2]+=Math.Min(cost[i-1][0],cost[i-1][1]);
     }
     int len=cost.Count;
     int min_cost=Math.Min(cost[len-1][0],Math.Min(cost[len-1][1],cost[len-1][2]));
     return min_cost;
     }
```

2.Area of the box

```
 class Result {

     public static List<Long> numberOfWays(List<List<Integer>> queries) {
     // Write your code herei
     int n,m;
     int min;
     long res=0;
     List<Long>fin=new ArrayList<>();
     for(List<Integer> lst:queries){
         n=lst.get(0);
         m=lst.get(1);
         min=Math.min(m,n);
         res=0;
         for(int i=1;i<=min;++i){
             res+=(n-i+1)*(m-i+1);
         }
         fin.add(res);
     }
 return fin;
     }

 }
```