ANIMAL CODE

```
interface Animal{
   void eat();
   void makeSound();
}


interface Bird{
   static int legs = 2;
   void fly();
}


class Parrot implements Bird, Animal{
  public void eat(){
      System.out.println("Parrots can eat up to 100 gms in a day")
   }
   public void makeSound(){
      System.out.println("Parrots make sound of screech");
   }
 public void fly(){
      System.out.println("Parrots can fly up to 50 miles in a day");
   }
}
```

EMPLOYEE CODE

```java
import java.io.*;

abstract class Employee{
    abstract void setSalary(int salary);
    abstract int getSalary();
    abstract void setGrade(String grade);
    abstract String getGrade();
    void label(){
    System.out.print("Employee's data:\n");
    }
}

class Engineer extends Employee{
    private int salary;
    private String grade;
```

```java

class Engineer extends Employee{
    private int salary;
    private String grade;
    public void setSalary(int salary){
        this.salary = salary;
    }
    public int getSalary(){
        return salary;
    }
    public void setGrade(String grade){
        this.grade = grade;
    }
    public String getGrade(){
        return grade;
    }
}

class Manager extends Employee{
    private int salary;
    private String grade;
    public void setSalary(int salary){
        this.salary = salary;
    }
    public int getSalary(){
        return salary;
    }
    public void setGrade(String grade){
        this.grade = grade;
    }
    public String getGrade(){
        return grade;
    }
}
public class Solution {…
```

https://youtu.be/c1ANbuywmBo - Almost equivalent string CODE

Maximum cumulative hackos – SQL CODE

```
SELECT MAX(MONTHS * HACKOS) AS MAXIMUM_HACKOS, COUNT(*) AS NUMBER_OF_HACKERS
FROM HACKER
WHERE MONTHS * HACKOS = (SELECT MAX(MONTHS * HACKOS) FROM HACKER)
```

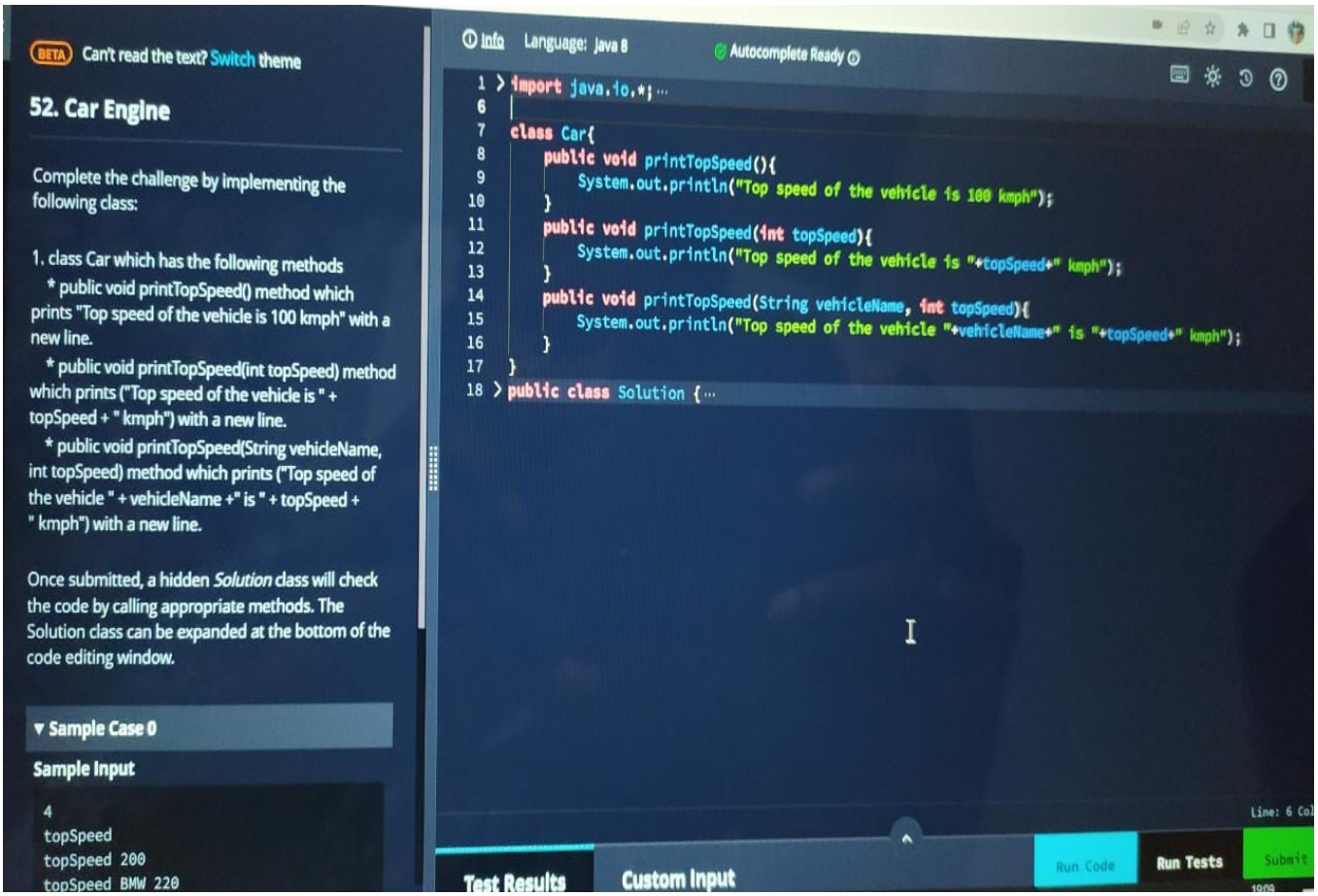
CAR CODE

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
class Car
{
   public void printTopSpeed()
   {
      System.out.println("Top speed of the vehicle is 100 kmph");
   }
   public void printTopSpeed(int topSpeed)
   {
      System.out.println("Top speed of the vehicle is " +topSpeed+ " kmph ");
   }
   public void printTopSpeed(String vehicleName,int topSpeed)
   {
```

```
        System.out.println("Top speed of the vehicle " +vehicleName+ " is " +topSpeed+ " kmph ");
    }
public void fuelType()
    {
        System.out.println("Car fuel type is Petrol");
    }
class SUV extends Car{
    public void fuelType()
    {
        System.out.println("SUV fuel type is Diesel");
    }
}


public class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        for(int i=0;i<2;i++) {
            String input = sc.nextLine();
            Car suv = new SUV();
            if(input.equals("topSpeed")){
                suv.topSpeed();
            }
            if(input.equals("fuelType")){
                suv.fuelType();
            }
```

```
        Car car = new Car();
        if(input.equals("topSpeed")){
            car.topSpeed();
        }
        if(input.equals("fuelType")){
            car.fuelType();
        }
}
}
}
```

CAR ENGINE CODE



Nutrition code

```
import java.util.*;

abstract class Food{

   double proteins;

   double fats;

   double carbs;

   double tastyScore;

   void getMacroNutrients(){}

}

class Bread extends Food{

   String type;

   public Bread(double proteins,double fats,double carbs) {

      this.proteins=proteins;

      this.fats=fats;

      this.carbs=carbs;

      this.tastyScore=8;

      this.type = "vegeterian";

   }

   void getMacroNutrients()

   {

      System.out.println("A slice of bread has "+String.valueOf(this.proteins)+" gms of protein,
"+String.valueOf(this.fats)+

            " gms of fats and "+String.valueOf(this.carbs)+" gms of carbohydrates.");

   }

}
```

```
class Egg extends Food{
   String type;
   public Egg(double proteins,double fats,double carbs) {
      this.proteins=proteins;
      this.fats=fats;
      this.carbs=carbs;
      this.tastyScore=8;
      this.type = "non-vegeterian";
   }
   void getMacroNutrients()
   {
      System.out.println("An egg has "+String.valueOf(this.proteins)+" gms of protein,
"+String.valueOf(this.fats)+
            " gms of fats and "+String.valueOf(this.carbs)+" gms of carbohydrates.");
   }
}
public class Solution {
   public static void main(String args[]) {


      Scanner sc = new Scanner(System.in);
      int cnt = Integer.parseInt(sc.nextLine());
      for(int i=0;i< cnt;i++)
      {
         String name = sc.nextLine();
         if(name.equals("Bread")) {
            Bread breadobj = new Bread(4, 1.1, 13.8);
```

```
            for (int j = 0; j < 3; j++) {
                String command = sc.nextLine();
                if (command.equals("getMacros"))
                    breadobj.getMacroNutrients();
                else if (command.equals("getTaste"))
                    System.out.println("Taste: " + breadobj.tastyScore);
                else if (command.equals("getType"))
                    System.out.println("Bread is " + breadobj.type);
            }
        }
        if(name.equals("Egg")) {
            Egg eggobj = new Egg(6.3, 5.3, 0.6);
            for (int j = 0; j < 3; j++) {
                String command = sc.nextLine();
                if (command.equals("getMacros"))
                    eggobj.getMacroNutrients();
                else if (command.equals("getTaste"))
                    System.out.println("Taste: " + eggobj.tastyScore);
                else if (command.equals("getType"))
                    System.out.println("Bread is " + eggobj.type);
            }
        }
    }
  }
}
```

CAR FUELING

```java
class Car {
    public void topSpeed() {
        System.out.println("Top Speed of the vehicle is 100 kmph");
    }

    public void fuelType() {
        System.out.println("Car fuel type is Petrol");
    }
}

class SUV extends Car {
    public void fuelType() {
        System.out.println("SUV fuel type is Diesel");
    }
}

```

BOTANY QUERY

```
/*
Enter your query here.
Please append a semicolon ";" at the end of the query
*/
select plant.name,a.type from plant
join(select plant_species,type,count(plant_species)
over(partition by plant_species)as count from weather)
a
on a.plant_species=plant.species and a.count=1;
```

Line: 7 Col: 48

Test Results

Run Query Submit

Compiled successfully. Correct answer.

Test case 0

Your Output (stdout)

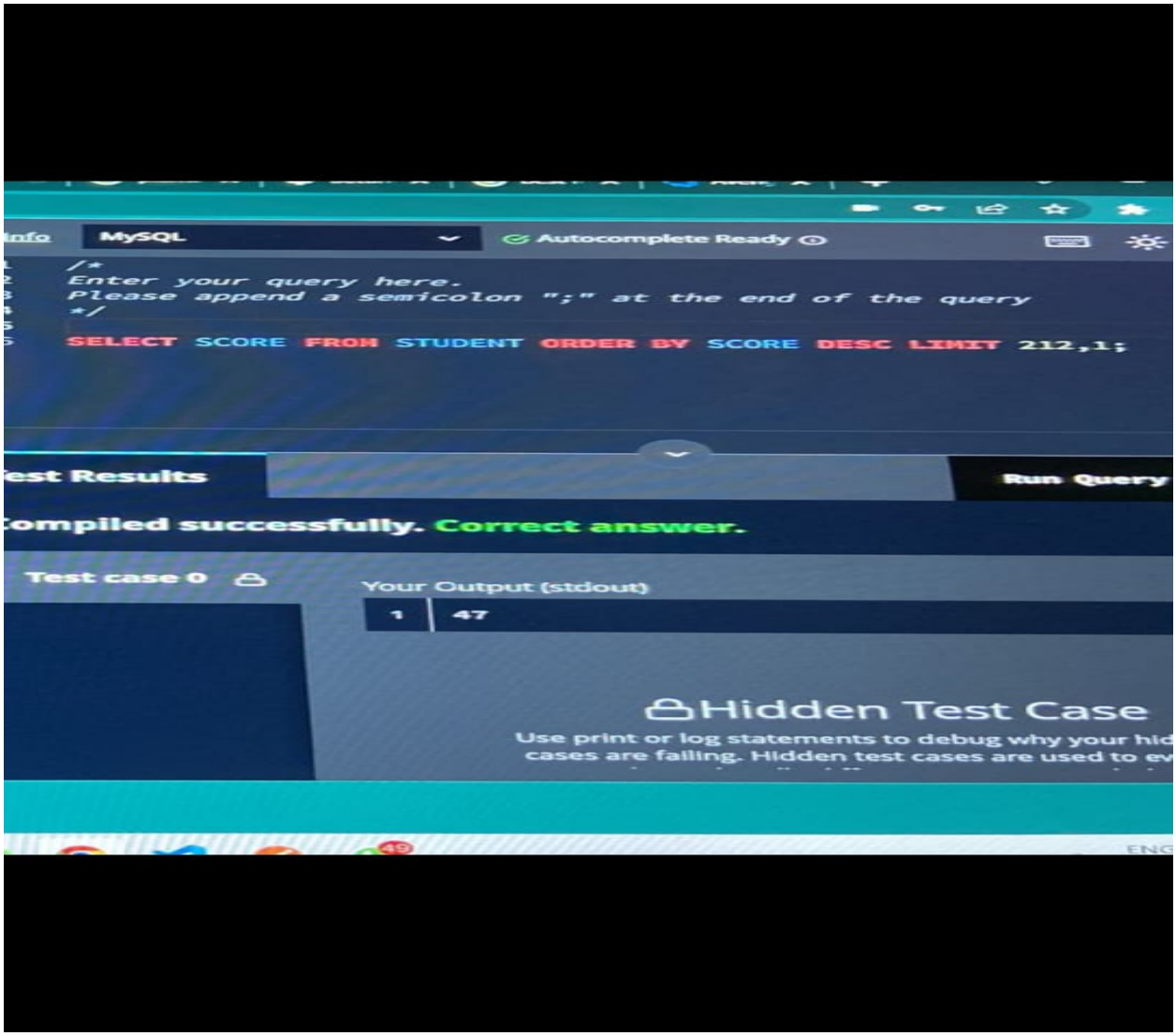
```
Curtisia Hail
```

Library structure code

```
class Library{
    private int number_of_books;
    private String name;
    private Map<String,Integer> bookGeneres = new HashMap<>();

    public int getNumber_of_books() {
        return number_of_books;
    }

    public void setNumber_of_books(int number_of_books) {
        this.number_of_books = number_of_books;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Map<String, Integer> getBookGeneres() {
        return bookGeneres;
    }
```

```
    public void setBookGeneres(Map<String, Integer> bookGeneres) {

        this.bookGeneres = bookGeneres;

    }




}
```

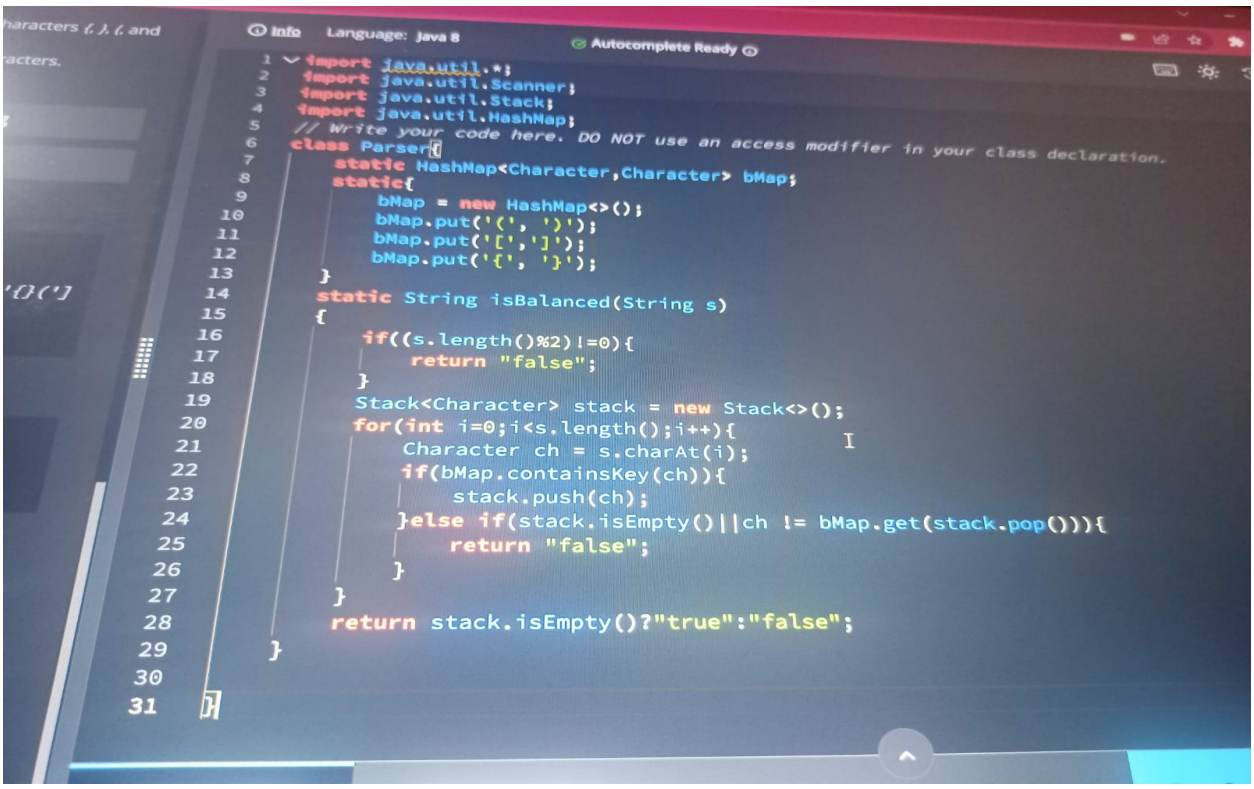
STUDENT RANK CODE



Examination data management CODE- SQL

```
SELECT student_id, subject, COUNT(*) AS no_of_times

FROM Examination

GROUP BY student_id, subject;
```

Java Braces CODE



Student class

```
class Student

{

   private String myName;

   private static int myRegNum = 0;

   Student(String name)

   {

      myName = name;

      myRegNum += 1;
```

```
    }

     @Override
    public String toString() {
        return this.myRegNum + ": " + this.myName;
    }
}
```

Substring CODE

```
class Solution {
    public int countBinarySubstrings(String s) {
        int[] groups = new int[s.length()];
        int t = 0;
        groups[0] = 1;
        for (int i = 1; i < s.length(); i++) {
            if (s.charAt(i-1) != s.charAt(i)) {
                groups[++t] = 1;
            } else {
                groups[t]++;
            }
        }

        int ans = 0;
        for (int i = 1; i <= t; i++) {
            ans += Math.min(groups[i-1], groups[i]);
        }
```

```
        return ans;
    }
}
```

Selling products CODE

```java
import java.util.Map.Entry;
import java.util.Map;
import java.util.HashMap;
public class MinimumDistinctId
{
    public static int getMinimumDistinctIds(int arr[], int n, int m)
    {
        Map<Integer, Integer> MAP = new HashMap<Integer, Integer>();
        int temp = 0;
        int cap = 0;
        for (int i = 0; i < n; i++)
        {
            if (MAP.containsKey(arr[i]) == false)
            {
                MAP.put(arr[i], 1);
                cap++;
            }
            else
                MAP.put(arr[i], MAP.get(arr[i]) + 1);
        }
        for (Entry<Integer, Integer> entry:MAP.entrySet())
        {
            if (entry.getKey() <= m)
            {
                m = m - entry.getKey();
                temp++;
            }
            else
                return cap - temp;
        }
        return cap - temp;
    }
    public static void main(String[] args)
    {
        int arr[] = {1, 1,1,2,3,2};
        int m = 2;
        System.out.println(getMinimumDistinctIds(arr, arr.length, m));
    }
}
```

Execute Mode, Version, Inputs & Arguments

JDK 17.0.1

Interactive

Stdin Inputs

CommandLine Arguments

Execute

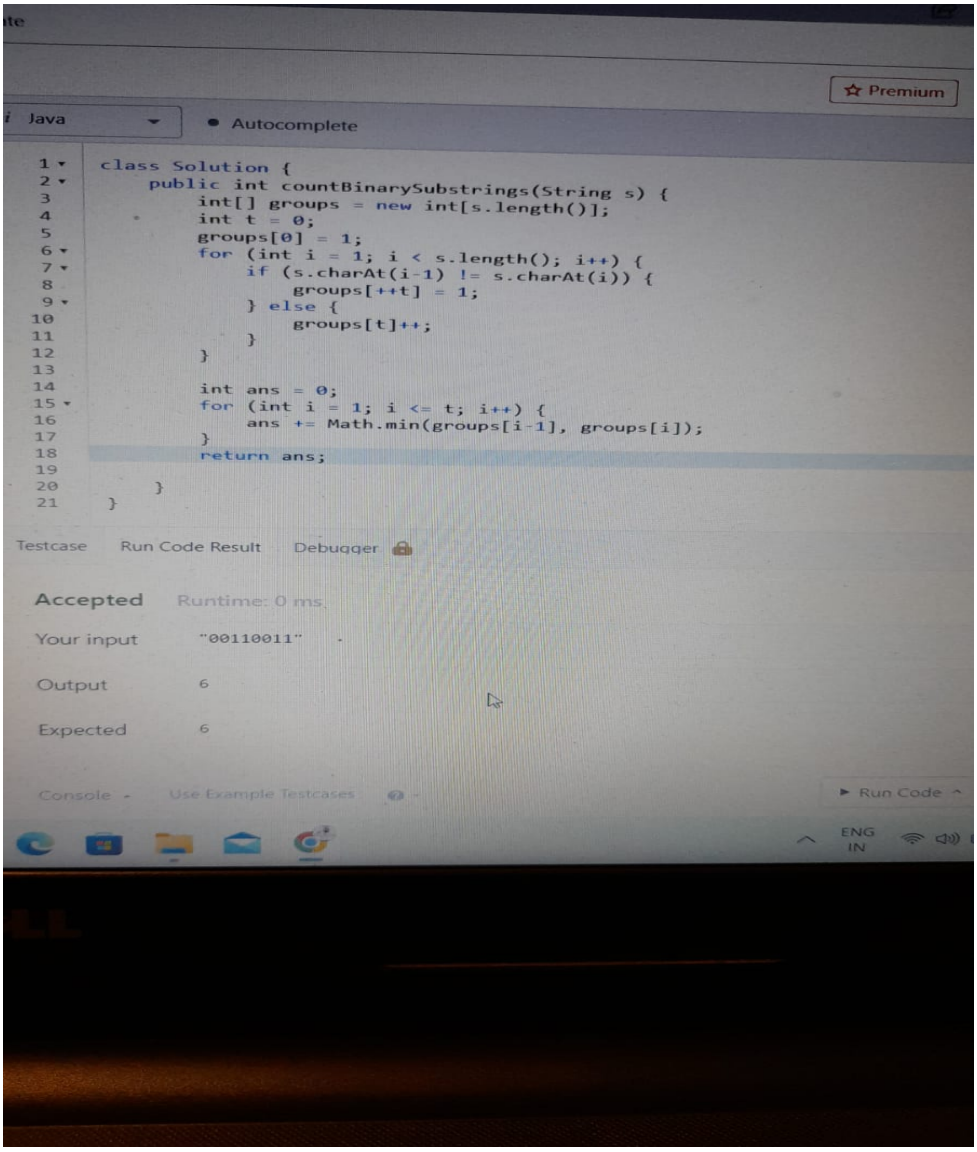
Result

CPU Time: 0.07 sec(s), Memory: 31952 kilobyte(s)

```
2
```

Note: Please check our documentation, or Youtube channel. for more details

Binary string CODE

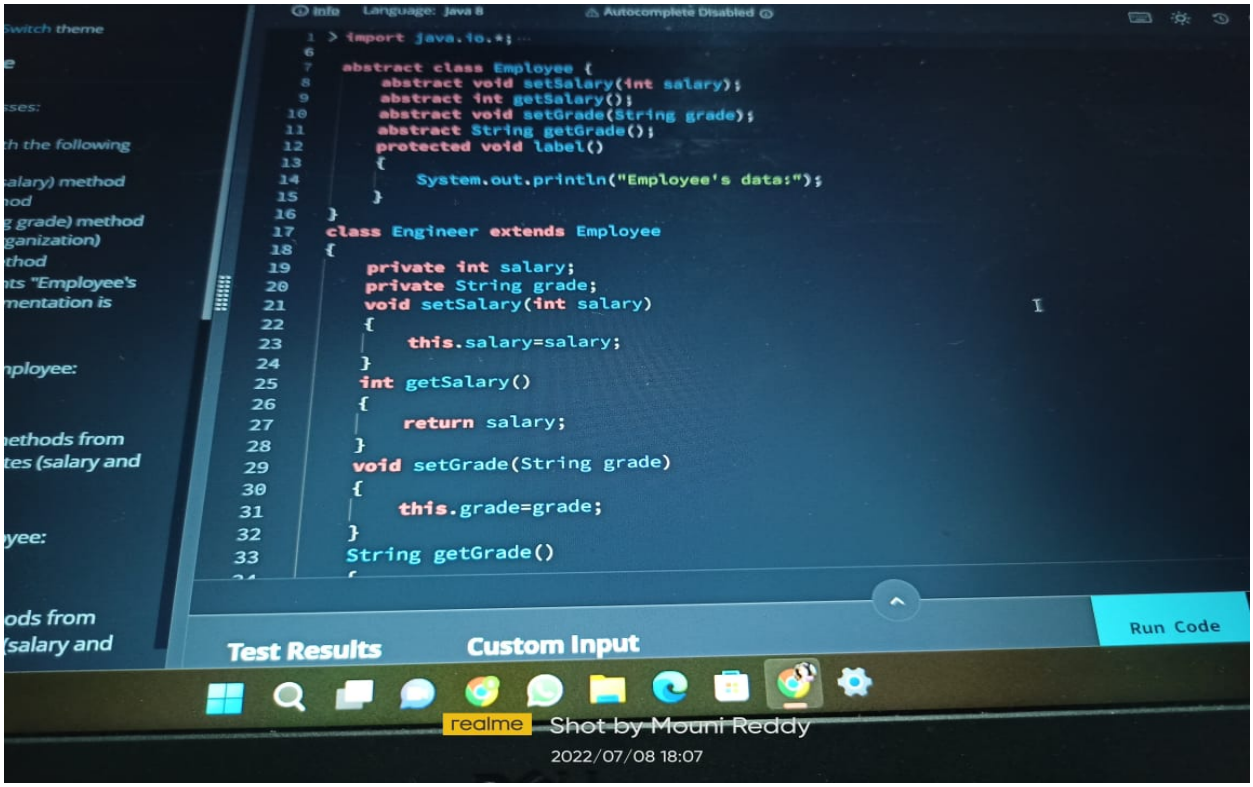


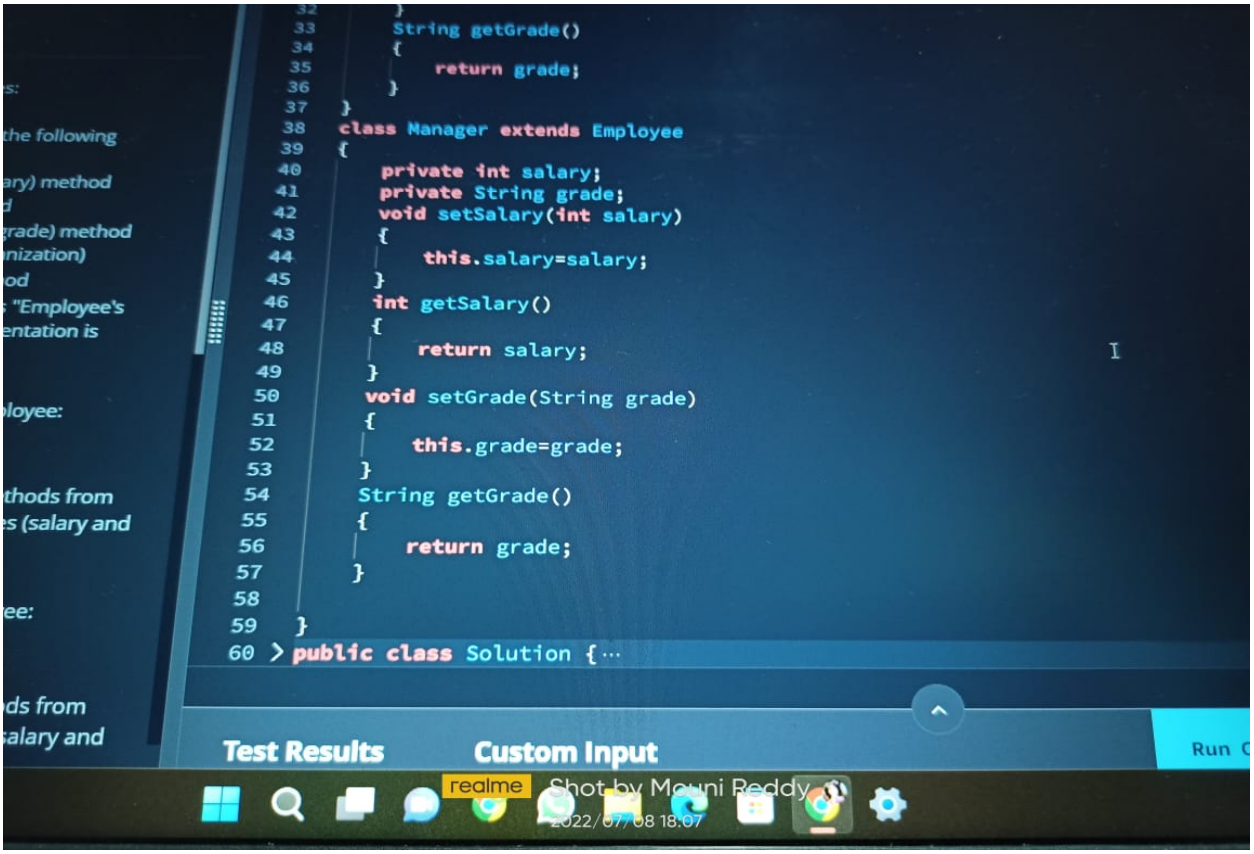
Car fueling

```
class Car
{
   public void topSpeed()
   {
      System.out.println("Top speed of the vehicle is 100 kmph");
   }
```

```
public void fuelType()
   {
      System.out.println("Car fuel type is Petrol");
   }
  }
class SUV extends Car{
   public void fuelType()
   {
      System.out.println("SUV fuel type is Diesel");
   }
}
```

EMPLOYEE PROFILE CODE





Count Binary string code

```
class Solution {

    public int countBinarySubstrings(String s) {

        int curr = 1, prev = 0, ans = 0;

        for (int i = 1; i < s.length(); i++)

            if (s.charAt(i) == s.charAt(i-1)) curr++;

            else {

                ans += Math.min(curr, prev);

                prev = curr;

                curr = 1;

            }

        return ans + Math.min(curr, prev);

    }
```

```
}
```

Serial multiplier CODE

```
import java.util.*;
public class SerialMultiplier
{
   static int first=1, second=1, third=1, fourth=1, fifth=1;
   static int result=0;
   public SerialMultiplier(int first)
   {
      this.first=first;
      result=first;
   }
   public SerialMultiplier(int first, int second)
   {
      this.first=first;
      this.second=second;
      result=first*second;
   }
   public SerialMultiplier(int first, int second, int third)
   {
      this.first=first;
      this.second=second;
      this.third=third;
      result=first*second*third;
```

```
}
 public SerialMultiplier(int first, int second, int third, int fourth)
{
   this.first=first;
   this.second=second;
   this.third=third;
   this.fourth=fourth;
   result=first*second*third*fourth;
}
 public SerialMultiplier(int first, int second, int third, int fourth, int fifth)
{
   this.first=first;
   this.second=second;
   this.third=third;
   this.fourth=fourth;
   this.fifth=fifth;
   result=first*second*third*fourth*fifth;

}

public static void main( String args[])
{
   Scanner sc= new Scanner(System.in);
   int n=sc.nextInt();
   int a[]=new int[n];
```

```
for(int i=0;i<n;i++)
{
a[i]=sc.nextInt();
}

if(n==1)
{
   SerialMultiplier obj=new SerialMultiplier(a[0]);
}
if(n==2)
{
   SerialMultiplier obj=new SerialMultiplier(a[0],a[1]);
}
if(n==3)
{
   SerialMultiplier obj=new SerialMultiplier(a[0],a[1],a[2]);
}
if(n==4)
{
   SerialMultiplier obj=new SerialMultiplier(a[0],a[1],a[2],a[3]);
}
if(n==5)
{
   SerialMultiplier obj=new SerialMultiplier(a[0],a[1],a[2],a[3],a[4]);
}
```

```
        System.out.println(result);
    }


}
```

Count Binary Substrings CODE

```
class Solution {
    public  int countBinarySubstrings(String s) {
        int count=0;
        for(int i=0;i<s.length()-1;i++){


            count+=doCount(s,i,i+1,0);


        }
        return count;
    }
    private int doCount(String s,int start,int end,int count){
        if(s.charAt(start)=='0'&&s.charAt(end)=='1'){
        while(start>=0&&end<s.length()&&s.charAt(start)=='0'&&s.charAt(end)=='1'){
            count++;
            start--;
            end++;


        }
        }else if(s.charAt(start)=='1'&&s.charAt(end)=='0'){
```

```
        while(start>=0&&end<s.length()&&s.charAt(start)=='1'&&s.charAt(end)=='0'){
            count++;
            start--;
            end++;
        }
        }
        return count;
    }
}
```