RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23333 OBJECT ORIENTED PROGRAMING USING JAVA

Laboratory Record Note Book

Name : .VASANTHU K

Year / Branch / Section : . . II/IT/D. .. . .

College Roll No. : . . . . . 2116231001239. . . . . .

Semester : . . . . . III. . . . . .

Academic Year : . . . . . 2024-2025 . . . . . .

RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name : . .VASANTHU K. .. . .

Academic Year : 2024-2025 Semester:. III Branch : IT-D

Register No. 2116231001239

Certified that this is the bonafide record of work done by the above student in the CS23333 –Object Oriented Programming using JAVA during the year 2024 - 2025.

Signature of Faculty in-charge

Submitted for the Practical Examination held on . .27/11/2024. . . . .

Internal Examiner **External Examiner**

INDEX

1.

Write a program to find whether the given input number is Odd.

If the given number is odd, the program should return 2 else It should return 1.

Note: The number passed to the program can either be negative. positive or zero. Zero should NOT be treated as Odd.

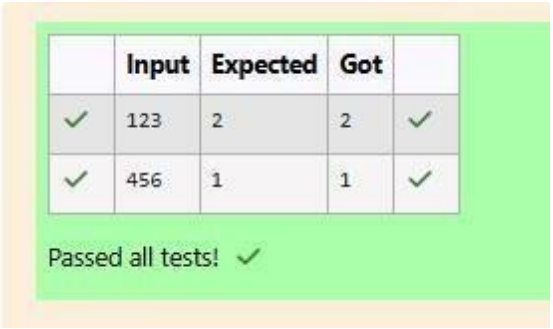
For example:

Input	Result
123	2
456	1

SOLUTION :

```
import java.util.Scanner;
public class oddorEven{
public static void
main(String[]args){ Scanner s=new
Scanner(System.in); int number =
s.nextInt(); if(number %2==0){
   System.out.println(1);
} else
{
   System.out.println(2);
}
}
}
```

OUTPUT :



	Input	Expected	Got	
✓	123	2	2	✓
✓	456	1	1	✓

Passed all tests! ✓

2.

Write a program that returns the last digit of the given number. Last digit is being referred to the least significant digit i.e. the digit in the ones (units) place in the given number.

The last digit should be returned as a positive number.

For example,

if the given number is 197, the last digit is 7

if the given number is -197, the last digit is 7

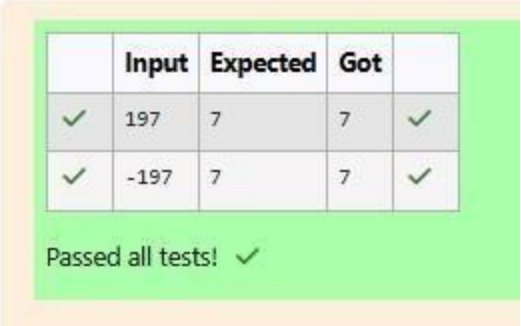
For example:

Input	Result
197	7
-197	7

SOLUTION :

```
import java.util.Scanner; import
java.lang.Math; public class LastDigit{
public static void main(String[]args){
Scanner s=new Scanner(System.in);
        int a = s.nextInt(); int
        lastDigit=Math.abs(a%10);
        System.out.println(lastDigit);
    }
    }
```

OUTPUT :



	Input	Expected	Got	
✓	197	7	7	✓
✓	-197	7	7	✓

Passed all tests! ✓

3.

Rohit wants to add the last digits of two given numbers.
For example,
If the given numbers are 267 and 154, the output should be 11.
Below is the explanation:
Last digit of the 267 is 7
Last digit of the 154 is 4
Sum of 7 and 4 = 11
Write a program to help Rohit achieve this for any given two numbers.
Note: Tile sign of the input numbers should be ignored.
i.e.
if the input numbers are 267 and 154, the sum of last two digits should be 11
if the input numbers are 267 and -154, the slim of last two digits should be 11
if the input numbers are -267 and 154, the sum of last two digits should be 11
if the input numbers are -267 and -154, the sum of last two digits should be 11

For example:

Input	Result
267 154	11
267 -154	11
-267 154	11
-267 -154	11

SOLUTION :

```
import java.util.Scanner;
import java.lang.Math;
 public class number{ public static void
    main(String[]args){ Scanner s= new
    Scanner(System.in);
        int a = s.nextInt();
        int b = s.nextInt();
        System.out.println(Math.abs(a)%10+Math.abs(b)%10);
    }
 }
```

OUTPUT:

	Input	Expected	Got	
✓	267 154	11	11	✓
✓	267 -154	11	11	✓
✓	-267 154	11	11	✓
✓	-267 -154	11	11	✓

Passed all tests! ✓

Lab-02-Flow Control Statements

1.

Consider the following sequence:

1st term: 1

2nd term: 1 2 1

3rd term: 1 2 1 3 1 2 1

4th term: 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

And so on. Write a program that takes as parameter an integer n and prints the nth terms of this sequence.

Example Input:

1

Output:

1

Example Input:

4

Output:

1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

For example:

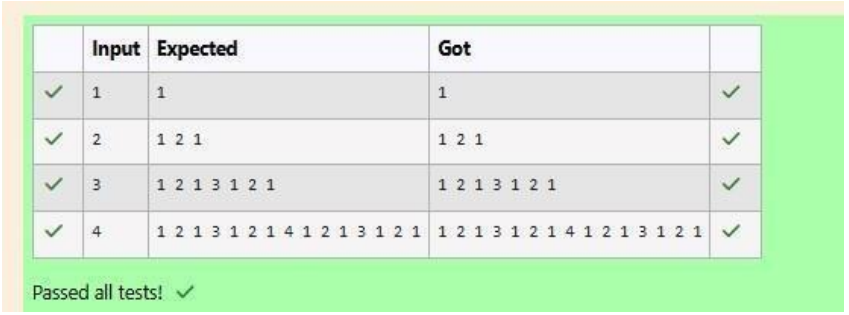
Input	Result
1	1
2	1 2 1
3	1 2 1 3 1 2 1
4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

SOLUTION :

```
import java.util.Scanner; public class
SequenceGenerator{ public static void
main(String[]args){ Scanner S = new
Scanner(System.in);
        int n = S.nextInt();
        String term = generateTerm(n);
        System.out.print(term);
    }
    private static String generateTerm(int n){
        if (n==1){ return "1";
        }
        String prevTerm = generateTerm (n-1);
        StringBuilder currentTerm = new StringBuilder(prevTerm);
```

```
        currentTerm.append(" " + n + " ");
        currentTerm.append(prevTerm);
        return currentTerm.toString();
    }
}
```

OUTPUT :



	Input	Expected	Got	
✓	1	1	1	✓
✓	2	1 2 1	1 2 1	✓
✓	3	1 2 1 3 1 2 1	1 2 1 3 1 2 1	✓
✓	4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1	✓

Passed all tests! ✓

2.

Write a program that takes as parameter an integer n.
You have to print the number of zeros at the end of the factorial of n.
For example, 3! = 6. The number of zeros are 0. 5! = 120. The number of zeros at the end are 1.
Note: n! < 10^5
Example Input:
3
Output:
0
Example Input:
60
Output:
14
Example Input:
100
Output:
24
Example Input:
1024
Output:
253

For example:

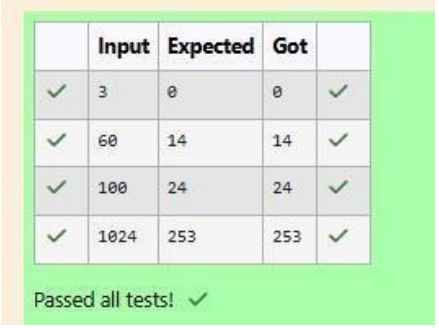
Input	Result
3	0
60	14
100	24
1024	253

SOLUTION :

```
// Java program to count trailing 0s in n!
import java.io.*; import
java.util.Scanner;
class prog {
    // Function to return trailing
    // 0s in factorial of n
    static int findTrailingZeros(int n)
    { if (n < 0) // Negative Number Edge Case
        return -1;
```

```
        // Initialize result

        int count=0;
        // Keep dividing n by powers //
        of 5 and update count for (int i =
        5; n / i >= 1; i*=5          ){ count
        += n / i;
} return count;
    }

    // Driver Code
    public static void main(String[] args)
    {
        Scanner sc= new Scanner(System.in);
        int n=sc.nextInt();
        int res=findTrailingZeros(n);
        System.out.println(res);
    }
}
```

OUTPUT :



	Input	Expected	Got	
✓	3	0	0	✓
✓	60	14	14	✓
✓	100	24	24	✓
✓	1024	253	253	✓

Passed all tests! ✓

3.

Consider a sequence of the form 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149...

Write a method program which takes as parameter an integer n and prints the nth term of the above sequence. The nth term will fit in an integer value.

Example Input:

5

Output:

4

Example Input:

8

Output:

24

Example Input:

11

Output:

149

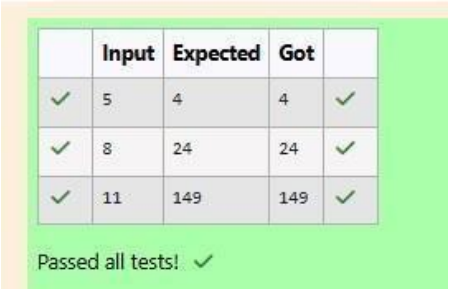
For example:

Input	Result
5	4
8	24
11	149

SOLUTION :

```
import java.util.Scanner;
class fibo3{ int a; int b;
int c;
   fibo3(int a,int b,int c){
      this.a = a; this.b =
      b; this.c = c;
   }
   int nth(int x){
      if (x == 1){
      return 0;
      }
      else if(x == 2 && x == 3)
         return 1;
      else{ int temp1,temp2,temp; int
         count = 4; while(x >=
         count){ temp =
         this.a+this.b+this.c;
            temp1 = this.c;
            this.c = temp;
            temp2 = this.b;
            this.b = temp1;
            this.a = temp2;
            count++;
         }
         return this.c;
      }
   }
}
public class Main{ public static void
   main(String[] args){ Scanner s = new
   Scanner(System.in);
      int t = s.nextInt(); fibo3 r
      = new fibo3(0,1,1);
      System.out.print(r.nth(t));
   }
}
```

OUTPUT :



	Input	Expected	Got	
✓	5	4	4	✓
✓	8	24	24	✓
✓	11	149	149	✓

Passed all tests! ✓

Lab-03-Arrays

1.

You are provided with a set of numbers (array of numbers).

You have to generate the sum of specific numbers based on its position in the array set provided to you.

This is explained below:

Example 1:

Let us assume the encoded set of numbers given to you is:

input1:5 and input2: {1, 51, 436, 7860, 41236}

Step 1:

Starting from the 0^{th} index of the array pick up digits as per below:

0^{th} index – pick up the units value of the number (in this case is 1).

1^{st} index - pick up the tens value of the number (in this case it is 5).

2^{nd} index - pick up the hundreds value of the number (in this case it is 4).

3^{rd} index - pick up the thousands value of the number (in this case it is 7).

4^{th} index - pick up the ten thousands value of the number (in this case it is 4).

(Continue this for all the elements of the input array).

The array generated from Step 1 will then be – {1, 5, 4, 7, 4}.

Step 2:

Square each number present in the array generated in Step 1.

{1, 25, 16, 49, 16}

Step 3:

Calculate the sum of all elements of the array generated in Step 2 to get the final result. The result will be = 107.

Note:

1) While picking up a number in Step1, if you observe that the number is smaller than the required position then use 0.

2) In the given function, input1[] is the array of numbers and input2 represents the number of elements in input1.

Example 2:

input1: 5 and input1: {1, 5, 423, 310, 61540}

Step 1:

Generating the new array based on position, we get the below array:

{1, 0, 4, 0, 6}

In this case, the value in input1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0.

Step 2:

{1, 0, 16, 0, 36}

Step 3:

The final result = 53.

For example:

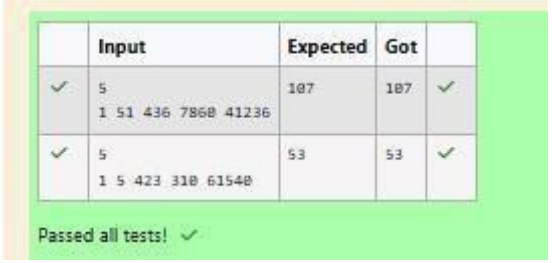
Input	Result
5 1 51 436 7860 41236	107
5 1 5 423 310 61540	53

SOLUTION :

```
import java.util.Scanner; public class
digit{ public static void
main(String[]args){
        Scanner scanner =new Scanner(System.in);
```

```
        int size =scanner.nextInt();
        int[]inpar=new int[size];
        for(int i=0;i<size;i++){
        inpar[i]=scanner.nextInt();
        }
        int[]dig=new int[size];
        for(int i=0;i<size;i++){
        int num=inpar[i];
        if(i==0){
        dig[i]=num%10;
                }
                else if (i==1){
                   dig[i]=(num/10)%10;
                }
                else if(i==2){
                   dig[i]=(num/100)%10;
                }
                else if(i==3){
                   dig[i]=(num/1000)%10;

                }
                else if(i==4){
                   dig[i]=(num/10000)%10;
                } else{
                dig[i]=0;
                }
        } int fin=0;
        for(int digi:dig){
        fin+=digi*digi;
        }
        System.out.print(fin);
    }
}
```

OUTPUT :



	Input	Expected	Got	
✓	5 1 51 436 7860 41236	107	107	✓
✓	5 1 5 423 310 61540	53	53	✓

Passed all tests! ✓

2.

Given an array of numbers, you are expected to return the sum of the longest sequence of POSITIVE numbers in the array.
If there are NO positive numbers in the array, you are expected to return -1.
In this question's scope, the number 0 should be considered as positive.
Note: If there are more than one group of elements in the array having the longest sequence of POSITIVE numbers, you are expected to return the total sum of all those POSITIVE numbers (see example 3 below).
input1 represents the number of elements in the array.
input2 represents the array of integers.
Example 1:
input1 = 16
input2 = {-12, -16, 12, 18, 18, 14, -4, -12, -13, 32, 34, -5, 66, 78, 78, -79}
Expected output = 62
Explanation:
The input array contains four sequences of POSITIVE numbers, i.e. "12, 18, 18, 14", "12", "32, 34", and "66, 78, 78". The first sequence "12, 18, 18, 14" is the longest of the four as it contains 4 elements. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = 12 + 18 + 18 + 14 = 63.
Example 2:
input1 = 11
input2 = {-22, -24, 16, -1, -17, -19, -37, -25, -19, -93, -61}
Expected output = -1
Explanation:
There are NO positive numbers in the input array. Therefore, the expected output for such cases = -1.
Example 3:
input1 = 16
input2 = {-58, 32, 26, 92, -10, -4, 12, 0, 12, -2, 4, 32, -9, -7, 78, -79}
Expected output = 174
Explanation:
The input array contains four sequences of POSITIVE numbers, i.e. "32, 26, 92", "12, 0, 12", "4, 32", and "78". The first and second sequences "32, 26, 92" and "12, 0, 12" are the longest of the four as they contain 4 elements each. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = (32 + 26 + 92) + (12 + 0 + 12) = 174.

For example:

Input	Result
16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62
11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1
16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174

SOLUTION :

```
import java.util.Scanner; public class
longdig{ public static void
main(String[]args){ Scanner sc=new
Scanner(System.in);
        int n=sc.nextInt();
        int c = 1,v,seqtemp = 0,seq = 0,countmax = 0;
        int count = 0; while(c <= n){ v = sc.nextInt();
        if(v >= 0){ countmax= countmax + v;
            seqtemp++;
            }
            else{
                seqtemp = 0;
                countmax = 0;
            }
            if(seqtemp > seq ){
                seq = seqtemp;
                count = countmax;
            }
            else if (seq == seqtemp){
                count = count + countmax;
            }
        c++; }
        if (count == 0)
            System.out.print(-1);
        else
            System.out.print(count);
```

```
}
```

```
}
```

OUTPUT :

	Input	Expected	Got	
✓	16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62	62	✓
✓	11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1	-1	✓
✓	16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174	174	✓

Passed all tests! ✓

3.

Given an integer array as input, perform the following operations on the array, in the below specified sequence.

1. Find the maximum number in the array.
2. Subtract the maximum number from each element of the array.
3. Multiply the maximum number (found in step 1) to each element of the resultant array.

After the operations are done, return the resultant array.

Example 1:

input1 = 4 (represents the number of elements in the input1 array)

input2 = {1, 5, 6, 9}

Expected Output = {-72, -36, 27, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(1 - 9), (5 - 9), (6 - 9), (9 - 9)} = {-8, -4, -3, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-8 x 9), (-4 x 9), (3 x 9), (0 x 9)} = {-72, -36, -27, 0}

So, the expected output is the resultant array {-72, -36, -27, 0}.

Example 2:

input1 = 5 (represents the number of elements in the input1 array)

input2 = {10, 87, 63, 42, 2}

Expected Output = {-6699, 0, -2088, -3915, -7395}

Explanation:

Step 1: The maximum number in the given array is 87.

Step 2: Subtracting the maximum number 87 from each element of the array:

{(10 - 87), (87 - 87), (63 - 87), (42 - 87), (2 - 87)} = {-77, 0, -24, -45, -85}

Step 3: Multiplying the maximum number 87 to each of the resultant array:

{(-77 x 87), (0 x 87), (-24 x 87), (-45 x 87), (-85 x 87)} = {-6699, 0, -2088, -3915, -7395}

So, the expected output is the resultant array {-6699, 0, -2088, -3915, -7395}.

Example 3:

input1 = 2 (represents the number of elements in the input1 array)

input2 = {-9, 9}

Expected Output = {-162, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(-9 - 9), (9 - 9)} = {-18, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-18 x 9), (0 x 9)} = {-162, 0}

So, the expected output is the resultant array {-162, 0}.

Note: The input array will contain not more than 100 elements

For example:

Input	Result
4 1 5 6 9	-72 -36 -27 0
5 10 87 63 42 2	-6699 0 -2088 -3915 -7395

SOLUTION :

```
import java.util.Scanner; public
class res{ public static
int[]pa(int[]arr){
        int maxs=Integer.MIN_VALUE;
        for (int num:arr){
            if(num>maxs){
            maxs=num;
            }
        }
        for(int i=0;i<arr.length;i++){ arr[i]=(arr[i]-
            maxs)*maxs;
        }
        return arr;
    }
    public static void main(String[]args){
        Scanner scanner =new Scanner (System.in);
        int n=scanner.nextInt();
        int[]arr=new int[n]; for(int
        i=0;i<n;i++){
        arr[i]=scanner.nextInt();
        }
        int[]res=pa(arr);
        for(int i=0;i<n;i++){
            System.out.print(res[i]+" ");
        }
        scanner.close();
    }
}
```

OUTPUT :

	Input	Expected	Got	
✓	4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0	✓
✓	5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395	✓
✓	2 -9 9	-162 0	-162 0	✓

Passed all tests! ✓

Lab-04-Classes and Objects

1.

Create a class called "Circle" with a radius attribute. You can access and modify this attribute using getter and setter methods. Calculate the area and circumference of the circle.

Area of Circle = πr^2

Circumference = $2\pi r$

Input:

2

Output:

Area = 12.57
Circumference = 12.57

For example:

Test	Input	Result
1	4	Area = 50.27 Circumference = 25.13

SOLUTION :

```
import java.io.*; import
java.util.Scanner;  class
Circle
{ private double radius; public
   Circle(double radius){
      // set the instance variable radius
     this.radius =radius;
        } public void setRadius(double
   radius){
      // set the radius
      this.radius=radius;

   }
public double getRadius()       {
      // return the radius
      return radius;

   }
   public double calculateArea() { // complete the below statement
      return Math.PI*radius*radius;

   }
public double calculateCircumference()       {
      // complete the statement return
      2*Math.PI*radius;
   }
} class prog{ public static void
main(String[] args) { int r;
       Scanner sc= new Scanner(System.in);
       r=sc.nextInt();
       Circle c= new Circle(r);
       System.out.println("Area = "+String.format("%.2f",
       c.calculateArea()));
       // invoke the calculatecircumference method
       System.out.println("Circumference = "+String.format("%.2f" ,
c.calculateCircumference()));

       sc.close();
   }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	4	Area = 50.27 Circumference = 25.13	Area = 50.27 Circumference = 25.13	✓
✓	2	6	Area = 113.10 Circumference = 37.70	Area = 113.10 Circumference = 37.70	✓
✓	3	2	Area = 12.57 Circumference = 12.57	Area = 12.57 Circumference = 12.57	✓

Passed all tests! ✓

2.

Create a Class Mobile with the attributes listed below,

private String manufacturer;
private String operating_system;
public String color;
private int cost;

Define a Parameterized constructor to initialize the above instance variables.

Define getter and setter methods for the attributes above.

for example : setter method for manufacturer is

void setManufacturer(String manufacturer){

this.manufacturer= manufacturer;

}

String getManufacturer(){

return manufacturer;}

Display the object details by overriding the toString() method.

For example:

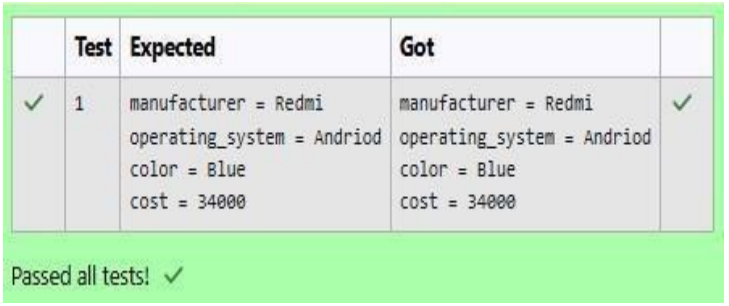
Test	Result
1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000

SOLUTION :

```
public class mobile{
   private String man;
   private String os;
   public String clr;
   private int cost;
   public mobile(String man,String os,String clr,int cost){
      this.man=man; this.os=os; this.clr=clr;
      this.cost=cost;
      }
      public String toString(){ return "manufacturer = "+man+"\n"+"operating_system =
"+os+"\n"+"color = "+ clr+"\n"+"cost = "+cost;
      }
      public static void main(String[]args){
```

```
        mobile mobile=new mobile("Redmi","Andriod","Blue",34000);
        System.out.println(mobile);
    }
}
```

OUTPUT :



	Test	Expected	Got	
✓	1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	✓

Passed all tests! ✓

3.

Create a class Student with two private attributes, name and roll number. Create three objects by invoking different constructors available in the class Student.

Student()

Student(String name)

Student(String name, int rollno)

Input:

No input

Output:

No-arg constructor is invoked
1 arg constructor is invoked
2 arg constructor is invoked
Name =null , Roll no = 0
Name =Rajalakshmi , Roll no = 0
Name =Lakshmi , Roll no = 101

For example:

Test	Result
1	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101

SOLUTION :

```
public class stud{ private String name; private int roll;
  public stud(){
     System.out.println("No-arg constructor is invoked"); name=null; roll=0;

  }
  public stud(String name){
     System.out.println("1 arg constructor is invoked"); this.name=name; roll=0;
```

```
}
public stud(String name,int roll){
   System.out.println("2 arg constructor is invoked"); this.name=name;
   this.roll=roll;

   }

public static void main (String[]args){
        stud s1=new stud(); stud s2=new
        stud("Rajalakshmi"); stud s3=new
        stud("Lakshmi",101);
        System.out.println("Name ="+s1.name+" , Roll no = "+s2.roll);
        System.out.println("Name ="+s2.name+" , Roll no = "+s2.roll);
        System.out.println("Name ="+s3.name+" , Roll no = "+s3.roll);
     }
   }
```

OUTPUT :

	Test	Expected	Got	
✓	1	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101	✓

Passed all tests! ✓

Lab-05-Inheritance

1.

Create a class known as "BankAccount" with methods called deposit() and withdraw().

Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.

For example:

Result
Create a Bank Account object (A/c No. BA1234) with initial balance of $500: Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300: Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0

SOLUTION :

```
class BankAccount {
// Private field to store the account number
private String accountNumber;
```

```
// Private field to store the balance
```

```
    private double balance;

    // Constructor to initialize account number and balance
    public BankAccount(String accountNumber,double balance){
    this.accountNumber=accountNumber;
        this.balance=balance;
    }




    // Method to deposit an amount into the account
    public void deposit(double amount) {
        // Increase the balance by the deposit amount
     balance+=amount;
    }

    // Method to withdraw an amount from the account
    public void withdraw(double amount) {
        // Check if the balance is sufficient for the withdrawal
        if (balance >= amount) {
            // Decrease the balance by the withdrawal amount
            balance -= amount;
        } else {
            // Print a message if the balance is
        insufficient System.out.println("Insufficient
        balance"); }
    }

    // Method to get the current balance
    public double getBalance() { //
    Return the current balance
        return balance;
    }
    public String getAccountNumber(){
        return accountNumber;
    }
}
class SavingsAccount extends BankAccount {
    // Constructor to initialize account number and balance
    public SavingsAccount(String accountNumber, double balance) {
        // Call the parent class constructor
        super(accountNumber,balance);
    }

    // Override the withdraw method from the parent class
    @Override
```

```
public void withdraw(double amount) {
    // Check if the withdrawal would cause the balance to drop below $100
```

```
        if (getBalance() - amount < 100) {
            // Print a message if the minimum balance requirement is not met
            System.out.println("Minimum balance of $100 required!");
        } else {
            // Call the parent class withdraw method
            super.withdraw(amount);
        }
    }
} public class Main {

    public static void main(String[] args) {
        // Print message to indicate creation of a BankAccount object
        System.out.println("Create a Bank Account object (A/c No. BA1234) with initial balance of $500:");
        // Create a BankAccount object (A/c No. "BA1234") with initial balance of $500
        BankAccount BA1234 = new BankAccount("BA1234", 500);
        // Print message to indicate deposit action
        System.out.println("Deposit $1000 into account BA1234:");
        // Deposit $1000 into account BA1234
       BA1234.deposit(1000);
        // Print the new balance after deposit
         System.out.println("New balance after depositing $1000: $"+BA1234.getBalance());

        // Print message to indicate withdrawal action
        System.out.println("Withdraw $600 from account BA1234:");
        // Withdraw $600 from account BA1234
      BA1234.withdraw(600);
        // Print the new balance after withdrawal
        System.out.println("New balance after withdrawing $600: $" + BA1234.getBalance());

        // Print message to indicate creation of another SavingsAccount object
        System.out.println("Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300:");
        // Create a SavingsAccount object (A/c No. "SA1000") with initial balance of $300
        SavingsAccount SA1000 = new SavingsAccount("SA1000", 300);

        // Print message to indicate withdrawal action
        System.out.println("Try to withdraw $250 from SA1000!");
        // Withdraw $250 from SA1000 (balance falls below $100)
        SA1000.withdraw(250);
        // Print the balance after attempting to withdraw $250
        System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance()); }
}
```

OUTPUT :

	Expected	Got	
✓	Create a Bank Account object (A/c No. BA1234) with initial balance of $500: Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300: Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0	Create a Bank Account object (A/c No. BA1234) with initial balance of $500: Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300: Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0	✓

Passed all tests! ✓

2.

create a class called College with attribute String name, constructor to initialize the name attribute , a method called Admitted(). Create a subclass called CSE that extends Student class, with department attribute , Course() method to sub class. Print the details of the Student.

College:

String collegeName;

public College() { }

public admitted() { }

Student:

String studentName;

String department;

public Student(String collegeName, String studentName,String depart) { }

public toString()

Expected Output:

A student admitted in REC
CollegeName : REC
StudentName : Venkatesh
Department : CSE

For example:

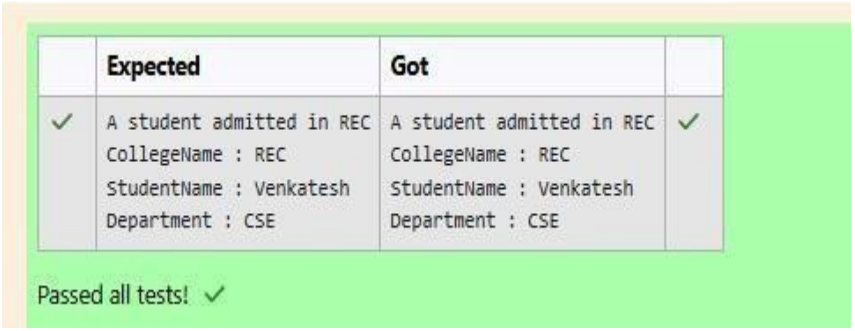
Result
A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE

SOLUTION :

```
class College
{
public  String collegeName;

public College(String collegeName)
    { // initialize the instance variables
    this.collegeName=collegeName; }

public void admitted() {
    System.out.println("A student admitted in "+collegeName);
} } class Student extends
College{

String studentName;
String department;

public Student(String collegeName, String studentName,String department) {
   // initialize the instance variables
   super(collegeName);
   this.studentName=studentName;
   this.department=department;
```

```
}

public String toString(){
   // return the details of the student return "CollegeName :
"+collegeName+"\n"+"StudentName :
"+studentName+"\n"+"Department : "+department;
}
}
public class Main {
public static void main (String[] args) {
      Student s1 = new Student("REC","Venkatesh","CSE");
      s1.admitted();                        // invoke the admitted() method
      System.out.println(s1.toString());
}
}
```

OUTPUT :



	Expected	Got	
✓	A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	✓

Passed all tests! ✓

3.

Create a class Mobile with constructor and a method basicMobile().

Create a subclass CameraMobile which extends Mobile class , with constructor and a method newFeature().

Create a subclass AndroidMobile which extends CameraMobile, with constructor and a method androidMobile().

display the details of the Android Mobile class by creating the instance. .

```
class Mobile{

}
class CameraMobile  extends Mobile {
}
class AndroidMobile extends CameraMobile {
}
```

expected output:

```
Basic Mobile is Manufactured
Camera Mobile is Manufactured
Android Mobile is Manufactured
Camera Mobile with 5MG px
Touch Screen Mobile is Manufactured
```

For example:

Result
Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured

SOLUTION :

```
class mob{
   mob(){
      System.out.println("Basic Mobile is Manufactured");
```

```
    }
    void basmob(){
        System.out.println("Basic Mobile is Manufactured");
    }
}
class cam extends
    mob{ cam(){
    super();
        System.out.println("Camera Mobile is Manufactured");
    }
    void newm(){
        System.out.println("Camera Mobile with 5MG px");

    }
}
class and extends
    cam{ and(){
    super();
    System.out.println("Android Mobile is Manufactured");
    }
    void andmob(){
        System.out.println("Touch Screen Mobile is Manufactured");
    }
    } public class Main{ public static
void main(String[]args){ and
andmob=new and(); andmob.newm();
andmob.andmob();
    }

}
```

OUTPUT :

	Expected	Got	
✓	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	✓

Passed all tests! ✓

Lab-06-String, StringBuffer

1.

You are provided a string of words and a 2-digit number. The two digits of the number represent the two words that are to be processed.

For example:

If the string is "Today is a Nice Day" and the 2-digit number is 41, then you are expected to process the 4th word ("Nice") and the 1st word ("Today").

The processing of each word is to be done as follows:

Extract the Middle-to-Begin part: Starting from the middle of the word, extract the characters till the beginning of the word.

Extract the Middle-to-End part: Starting from the middle of the word, extract the characters till the end of the word.

If the word to be processed is "Nice":

Its Middle-to-Begin part will be "iN".

Its Middle-to-End part will be "ce".

So, merged together these two parts would form "iNce".

Similarly, if the word to be processed is "Today":

Its Middle-to-Begin part will be "doT".

Its Middle-to-End part will be "day".

So, merged together these two parts would form "doTday".

Note: Note that the middle letter 'd' is part of both the extracted parts. So, for words whose length is odd, the middle letter should be included in both the extracted parts.

Expected output:

The expected output is a string containing both the processed words separated by a space "iNce doTday"

Example 1:

input1 = "Today is a Nice Day"

input2 = 41

output = "iNce doTday"

Example 2:

input1 = "Fruits like Mango and Apple are common but Grapes are rare"

input2 = 39

output = "naMngo arGpes"

Note: The input string input1 will contain only alphabets and a single space character separating each word in the string.

Note: The input string input1 will NOT contain any other special characters.

Note: The input number input2 will always be a 2-digit number (>=11 and <=99). One of its digits will never be 0. Both the digits of the number will always point to a valid word in the input1 string.

For example:

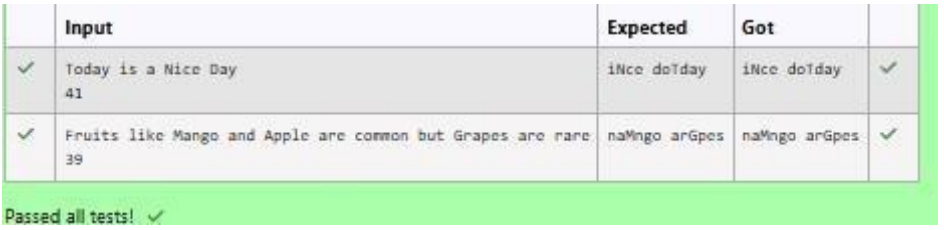
Input	Result
Today is a Nice Day 41	iNce doTday
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes

SOLUTION :

```
import java.util.*; public class mix{
public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String g = scan.nextLine(); int n =
        scan.nextInt(),ones,flag = 0; StringBuffer
        temp = new StringBuffer(); StringBuffer
        temp1 = new StringBuffer(); int space =
        0; while (n > 0){ ones = (n %10) - 1;
             for(int i = 0; i < g.length();i++){
                if (g.charAt(i) == ' '){ space
                = space + 1;
                }
                else if(space == ones && flag == 0){
                     temp.append(Character.toString(g.charAt(i)));
                }
                else if(space == ones && flag == 1){
                     temp1.append(Character.toString(g.charAt(i)));
                }
```

```
            } space =
            0 ; flag =
            1; n = n
            /10;
        }
        rew m = new rew();
        System.out.println(m.r(temp1.toString()) + " " + m.r(temp.toString()));
    }
}
class rew{
    String r(String a){ int le
        = a.length(),n,q;
        StringBuffer temp3 = new StringBuffer();
        if(le % 2 == 1){
            n = ((int)(le/2));
            q = ((int)(le/2));
        } else{ n =
        ((int)(le/2)) - 1;
            q = ((int)(le/2));
        }
        for(int i = n;i >= 0;i--){ temp3.append(Character.toString(a.charAt(i)));
            } for(int i = q;i < le;i++){
        temp3.append(Character.toString(a.charAt(i)));
        }
        return temp3.toString();
    }
}
```

OUTPUT :



	Input	Expected	Got	
✓	Today is a Nice Day 41	iNce doTday	iNce doTday	✓
✓	Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes	naMngo arGpes	✓

Passed all tests! ✓

2.

Given a String input1, which contains many number of words separated by : and each word contains exactly two lower case alphabets, generate an output based upon the below 2 cases.

Note:

1. All the characters in input 1 are lowercase alphabets.
2. input 1 will always contain more than one word separated by :
3. Output should be returned in uppercase.

Case 1:

Check whether the two alphabets are same.

If yes, then take one alphabet from it and add it to the output.

Example 1:

input1 = ww:ii:pp:rr:oo

output = WIPRO

Explanation:

word1 is ww, both are same hence take w

word2 is ii, both are same hence take i

word3 is pp, both are same hence take p

word4 is rr, both are same hence take r

word5 is oo, both are same hence take o

Hence the output is WIPRO

Case 2:

If the two alphabets are not same, then find the position value of them and find maximum value – minimum value.

Take the alphabet which comes at this (maximum value - minimum value) position in the alphabet series.

Example 2"

input1 = zx:za:ee

output = BYE

Explanation

word1 is zx, both are not same alphabets

position value of z is 26

position value of x is 24

max – min will be 26 – 24 = 2

Alphabet which comes in 2nd position is b

Word2 is za, both are not same alphabets

position value of z is 26

position value of a is 1

max – min will be 26 – 1 = 25

Alphabet which comes in 25th position is y

word3 is ee, both are same hence take e

Hence the output is BYE

For example:

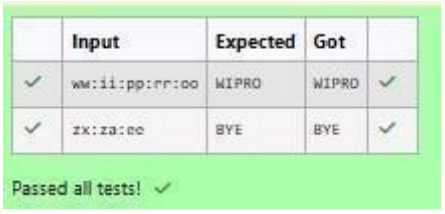
Input	Result
ww:ii:pp:rr:oo	WIPRO
zx:za:ee	BYE

SOLUTION :

```
import java.util.*; class diff{ char different(char
a, char b){ if ((int)a != (int)b) return
(char)((int)'a' + ((int)a-(int)b) - 1);
        return a;
        }
}
public class Main{ public static void
    main(String[] args){ Scanner scan = new
    Scanner(System.in);
        diff z = new diff();
        String q = scan.nextLine();
        StringBuffer ans = new StringBuffer();
        StringBuffer temp = new
        StringBuffer(); for(int i = 0;i <
        q.length();i++){ if(q.charAt(i) == ':'){
        temp.append(" ");
            } else{
            temp.append(Character.toString(q.charAt(i))); }
```

```
        }
        String h = temp.toString(); for(int i
        = 0;i < temp.length();i++){ if(i%3
        == 0){
              ans.append(Character.toString(z.different(h.charAt(i),h.charAt(i+1))));
           }
        }
        System.out.print(ans.toString().toUpperCase());
    }
}
```

OUTPUT :



	Input	Expected	Got	
✓	ww:ii:pp:rr:oo	WIPRO	WIPRO	✓
✓	zx:za:ee	BYE	BYE	✓

Passed all tests! ✓

3.

Given 2 strings input1 & input2.

- Concatenate both the strings.
- Remove duplicate alphabets & white spaces.
- Arrange the alphabets in descending order.

Assumption 1:

There will either be alphabets, white spaces or null in both the inputs.

Assumption 2:

Both inputs will be in lower case.

Example 1:

Input 1: apple

Input 2: orange

Output: rponlgea

Example 2:

Input 1: fruits

Input 2: are good

Output: utsroigfeda

Example 3:

Input 1: ""

Input 2: ""

Output: null

For example:

Test	Input	Result
1	apple orange	rponlgea
2	fruits are good	utsroigfeda

SOLUTION :

```
import java.util.*;


public class HelloWorld { public static
   void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String a = scan.nextLine();
      String b = scan.nextLine();
      StringBuffer ab = new StringBuffer();
      if(a.trim().isEmpty() && b.trim().isEmpty()){
      System.out.print("null");
      }
      else{
      for(int i = 0;i < a.length();i++){ if (a.charAt(i)
         != ' ') {
         ab.append(Character.toString(a.charAt(i))); }
      }
      for(int i = 0;i < b.length();i++){ if (b.charAt(i)
         != ' '){
         ab.append(Character.toString(b.charAt(i))); }
      }
      char[] d = ab.toString().toCharArray();
      Arrays.sort(d);
      for(int i = d.length - 1;i >= 1;i--){
         if(d[i] != d[i-1])
         System.out.print(d[i]);
      }
      System.out.print(d[0]);
      }

   }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	apple orange	rponlgea	rponlgea	✓
✓	2	fruits are good	utsroigfeda	utsroigfeda	✓
✓	3		null	null	✓

Passed all tests! ✓

Lab-07-Interfaces

1.

RBI issues all national banks to collect interest on all customer loans.

Create an RBI interface with a variable String parentBank="RBI" and abstract method rateOfInterest().

RBI interface has two more methods default and static method.

default void policyNote() {

System.out.println("RBI has a new Policy issued in 2023.");

}

static void regulations(){

System.out.println("RBI has updated new regulations on 2024.");

}

Create two subclasses SBI and Karur which implements the RBI interface.

Provide the necessary code for the abstract method in two sub-classes.

Sample Input/Output:

RBI has a new Policy issued in 2023
RBI has updated new regulations in 2024.
SBI rate of interest: 7.6 per annum.
Karur rate of interest: 7.4 per annum.

For example:

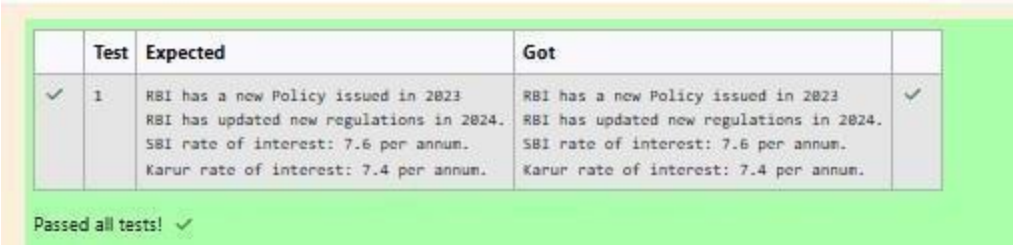
Test	Result
1	RBI has a new Policy issued in 2023 RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.

SOLUTION :

```
// Define the RBI interface
interface RBI {
    // Variable declaration
    String parentBank = "RBI";

    // Abstract method
    double rateOfInterest();

    // Default method
    default void policyNote() {
        System.out.println("RBI has a new Policy issued in 2023");
    }

    // Static method
    static void regulations() {
        System.out.println("RBI has updated new regulations in 2024.");
    }
}

// SBI class implementing RBI interface
class SBI implements RBI {
    // Implementing the abstract method
    public double rateOfInterest() {
```

```
        return 7.6;
    }
}

// Karur class implementing RBI
interface class Karur implements RBI { //
Implementing the abstract method public
double rateOfInterest() { return 7.4;
    }
}

// Main class to test the functionality
public class Main { public static void
main(String[] args) {
        // RBI policies and regulations
        RBI rbi = new SBI(); // Can be any class implementing RBI
        rbi.policyNote(); // Default method RBI.regulations();
            // Static method

        // SBI bank details
        SBI sbi = new SBI();
        System.out.println("SBI rate of interest: " + sbi.rateOfInterest() + " per annum.");

        // Karur bank details
        Karur karur = new Karur();
        System.out.println("Karur rate of interest: " + karur.rateOfInterest() + " per annum.");
    }
}
```

OUTPUT :



	Test	Expected	Got	
✓	1	RBI has a new Policy issued in 2023 RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.	RBI has a new Policy issued in 2023 RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.	✓

Passed all tests! ✓

2.

Create interfaces shown below.

```
interface Sports {
public void setHomeTeam(String name);
public void setVisitingTeam(String name);
}
interface Football extends Sports {
public void homeTeamScored(int points);
public void visitingTeamScored(int points);}
```

create a class College that implements the Football interface and provides the necessary functionality to the abstract methods.

sample Input:

```
Rajalakshmi
Saveetha
22
21
```

Output:

```
Rajalakshmi 22 scored
Saveetha 21 scored
Rajalakshmi is the Winner!
```

For example:

Test	Input	Result
1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!

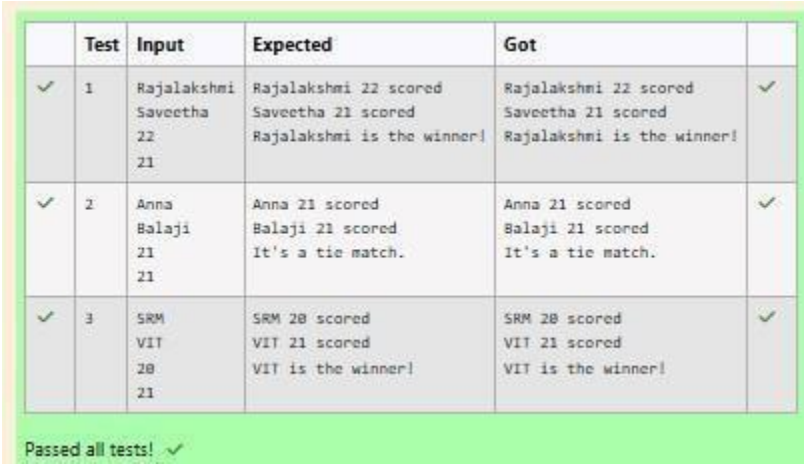
SOLUTION :

```
import java.util.Scanner;

interface Sports { void
    setHomeTeam(String name); void
    setVisitingTeam(String name);
}

interface Football extends Sports {
    void homeTeamScored(int points);
    void visitingTeamScored(int points);
}

class College implements Football {
    private String homeTeam; private
    String visitingTeam; private int
    homeTeamPoints = 0; private int
    visitingTeamPoints = 0;

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

    public void setVisitingTeam(String name) {
        this.visitingTeam = name;
    } public void homeTeamScored(int points)
    {
```

```
        homeTeamPoints += points;
        System.out.println(homeTeam + " " + points + " scored");
    }

    public void visitingTeamScored(int points) {
        visitingTeamPoints += points;
        System.out.println(visitingTeam + " " + points + " scored");
    }

    public void winningTeam() { if
        (homeTeamPoints > visitingTeamPoints) {
            System.out.println(homeTeam + " is the winner!");
        } else if (homeTeamPoints < visitingTeamPoints) {
            System.out.println(visitingTeam + " is the winner!");
        } else {
            System.out.println("It's a tie match.");
        }
    }
}

public class Main { public static void
    main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Get home team name
        String hname = sc.nextLine();

        // Get visiting team name
        String vteam = sc.nextLine();

        // Create College object College
        match = new College();
        match.setHomeTeam(hname);
        match.setVisitingTeam(vteam);

        // Get points scored by home team
        int htpoints = sc.nextInt();
        match.homeTeamScored(htpoints);

        // Get points scored by visiting team
        int vtpoints = sc.nextInt();
        match.visitingTeamScored(vtpoints);

        // Determine and print the winning team
        match.winningTeam();

        sc.close();
    }
```

```
}
```

OUTPUT :



	Test	Input	Expected	Got	
✓	1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	✓
✓	2	Anna Balaji 21 21	Anna 21 scored Balaji 21 scored It's a tie match.	Anna 21 scored Balaji 21 scored It's a tie match.	✓
✓	3	SRM VIT 20 21	SRM 20 scored VIT 21 scored VIT is the winner!	SRM 20 scored VIT 21 scored VIT is the winner!	✓

Passed all tests! ✓

3.

create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

```
interface Playable {
    void play();
}

class Football implements Playable {
    String name;
    public Football(String name){
        this.name=name;
    }
  public void play() {
    System.out.println(name+" is Playing football");
  }
}
```

Similarly, create Volleyball and Basketball classes.

Sample output:

```
Sadhvin is Playing football
Sanjay is Playing volleyball
Sruthi is Playing basketball
```

For example:

Test	Input	Result
1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball
2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball

SOLUTION :

```
import java.util.Scanner;

// Define the Playable interface
interface Playable {
    // Abstract method to play the respective sport
    void play();
}

// Football class implementing Playable interface
class Football implements Playable {
    String name;

    // Constructor
    public Football(String name) {
        this.name = name;
    }

    // Override the play method
```

```
    public void play() {
        System.out.println(name + " is Playing football");
    }
}

// Volleyball class implementing Playable interface
class Volleyball implements Playable {
    String name;

    // Constructor
    public Volleyball(String name) {
        this.name = name;
    }

    // Override the play method
    public void play() {
        System.out.println(name + " is Playing volleyball");
    }
}

// Basketball class implementing Playable interface
class Basketball implements Playable {
    String name;

    // Constructor
    public Basketball(String name) {
        this.name = name;
    }

    // Override the play method
    public void play() {
        System.out.println(name + " is Playing basketball");
    }
}

// Main class to test the functionality
public class Main { public static void
main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for Football player

        String footballPlayerName = scanner.nextLine();
        Football footballPlayer = new Football(footballPlayerName);

        // Input for Volleyball player
```

```
String volleyballPlayerName = scanner.nextLine();
Volleyball volleyballPlayer = new Volleyball(volleyballPlayerName);
```

```
        // Input for Basketball player

        String basketballPlayerName = scanner.nextLine();
        Basketball basketballPlayer = new Basketball(basketballPlayerName);

        // Call the play method for each player
        footballPlayer.play();
        volleyballPlayer.play();
        basketballPlayer.play();

        scanner.close();
    }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	✓
✓	2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	✓

Passed all tests! ✓

Lab-08 - Polymorphism, Abstract Classes, final Keyword

1.

As a logic building learner you are given the task to extract the string which has vowel as the first and last characters from the given array of Strings.

Step1: Scan through the array of Strings, extract the Strings with first and last characters as vowels; these strings should be concatenated.

Step2: Convert the concatenated string to lowercase and return it.

If none of the strings in the array has first and last character as vowel, then return no matches found

input1: an integer representing the number of elements in the array.

input2: String array.

Example 1:

input1: 3

input2: {"oreo", "sirish", "apple"}

output: oreoapple

Example 2:

input1: 2

input2: {"Mango", "banana"}

output: no matches found

Explanation:

None of the strings has first and last character as vowel.

Hence the output is no matches found.

Example 3:

input1: 3

input2: {"Ate", "Ace", "Girl"}

output: ateace

For example:

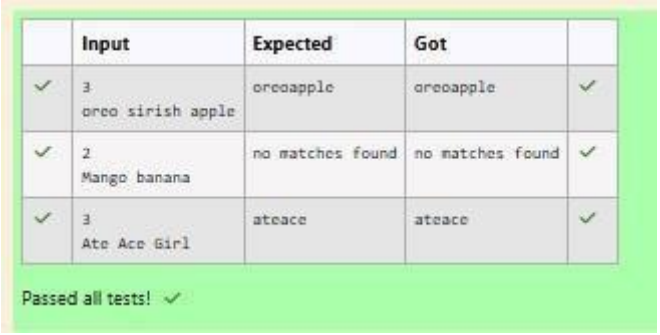
Input	Result
3 oreo sirish apple	oreoapple
2 Mango banana	no matches found
3 Ate Ace Girl	ateace

SOLUTION :

```
import java.util.Scanner; public

class VowelStringExtractor {

    // Method to extract strings with vowels as first and last characters
    public static String extractVowelStrings(String[] stringArray) {
        StringBuilder result = new StringBuilder();
        String vowels = "aeiouAEIOU"; // String containing all vowels

        // Iterate through the array of strings
        for (String s : stringArray) {
            // Check if the string is not empty and if both the first and last characters are vowels
if (s.length() > 0 && vowels.indexOf(s.charAt(0)) != -1 &&
vowels.indexOf(s.charAt(s.length() - 1)) != -1) { result.append(s); // Append matching
string to the result }
        }

        // Return the concatenated string in lowercase or "no matches found"
        return result.length() > 0 ? result.toString().toLowerCase() : "no matches found"; }
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for the number of strings

        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        // Input for the strings in one line

        String input = scanner.nextLine();
        String[] strings = input.split(" "); // Split input into an array

        // Process and output the result
        String result = extractVowelStrings(strings);
        System.out.println(result);

        scanner.close(); // Close the scanner
    }
}
```

OUTPUT :



	Input	Expected	Got	
✓	3 oreo sirish apple	oreoapple	oreoapple	✓
✓	2 Mango banana	no matches found	no matches found	✓
✓	3 Ate Ace Girl	ateace	ateace	✓

Passed all tests! ✓

2.

1. Final Variable:

- Once a variable is declared `final`, its value cannot be changed after it is initialized.
- It must be initialized when it is declared or in the constructor if it's not initialized at declaration.
- It can be used to define constants

```
final int MAX_SPEED = 120;  // Constant value, cannot be changed
```

2. Final Method:

- A method declared `final` cannot be overridden by subclasses.
- It is used to prevent modification of the method's behavior in derived classes.

```
public final void display() {
   System.out.println("This is a final method.");
}
```

3. Final Class:

- A class declared as `final` cannot be subclassed (i.e., no other class can inherit from it).
- It is used to prevent a class from being extended and modified.
- public final class Vehicle {
  // class code
  }

Given a Java Program that contains the bug in it, your task is to clear the bug to the output.

you should delete any piece of code.

For example:

Test	Result
1	The maximum speed is: 120 km/h This is a subclass of FinalExample.

SOLUTION :

```
// Final class definition
final class FinalExample {
    // Final variable
    final int MAX_SPEED = 120; // Constant value

    // Final method
    public final void display() {
        System.out.println("The maximum speed is: " + MAX_SPEED + " km/h");
    }
}

// Main class to test the final class public
class Test { public static void
main(String[] args) {
        // Create an instance of FinalExample
        FinalExample example = new FinalExample();
        example.display();

        // Uncommenting the following line will result in a compile-time error
        // because FinalExample is a final class and cannot be subclassed. //
        class SubclassExample extends FinalExample { }

        System.out.println("This is a subclass of FinalExample.");
    }
}
```

OUTPUT :

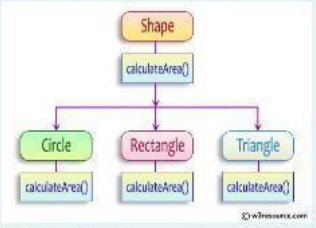
	Test	Expected	Got	
✓	1	The maximum speed is: 120 km/h This is a subclass of FinalExample.	The maximum speed is: 120 km/h This is a subclass of FinalExample.	✓

Passed all tests! ✓

3.

Create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.

In the given exercise, here is a simple diagram illustrating polymorphism implementation:




```
abstract class Shape {
  public abstract double calculateArea() ;
  }
}
```

System.out.printf("Area of a Triangle :%.2f%n",((0.5)*base*height)); // use this statement

sample Input :

4 // radius of the circle to calculate area PI*r*r

5 // length of the rectangle

6 // breadth of the rectangle to calculate the area of a rectangle

4 // base of the triangle

3 // height of the triangle

OUTPUT:

Area of a circle :50.27
Area of a Rectangle :30.00
Area of a Triangle :6.00

For example:

Test	Input	Result
1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00
2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32

SOLUTION :

```
import java.util.Scanner;

// Abstract class Shape abstract class
Shape { public abstract double
calculateArea();
}

// Circle class
class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
```

```
    public double calculateArea() { return Math.PI * radius
    * radius; // Area of circle: πr² }
}

// Rectangle class
class Rectangle extends Shape {
    private double length; private
    double breadth;

    public Rectangle(double length, double breadth) {
        this.length = length; this.breadth = breadth;
    }

    @Override
    public double calculateArea() { return length * breadth; // Area
        of rectangle: length * breadth
    }
}

// Triangle class
class Triangle extends Shape {
    private double base; private
    double height;

    public Triangle(double base, double height) {
        this.base = base; this.height = height;
    }

    @Override
    public double calculateArea() { return 0.5 * base * height; // Area
        of triangle: 0.5 * base * height
    }
}

// Main class to test the shapes public
class ShapeTest { public static void
main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for Circle

        double radius = scanner.nextDouble();
        Circle circle = new Circle(radius);
        System.out.printf("Area of a circle: %.2f%n", circle.calculateArea());

        // Input for Rectangle
```

```
        double length = scanner.nextDouble();

        double breadth = scanner.nextDouble();
        Rectangle rectangle = new Rectangle(length, breadth);
        System.out.printf("Area of a Rectangle: %.2f%n", rectangle.calculateArea());

        // Input for Triangle double base =

        scanner.nextDouble();

        double height = scanner.nextDouble();
        Triangle triangle = new Triangle(base, height);
        System.out.printf("Area of a Triangle: %.2f%n", triangle.calculateArea());

        scanner.close();
    }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	✓
✓	2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	✓

Passed all tests! ✓

Lab-09-Exception Handling

1.

Write a Java program to create a method that takes an integer as a parameter and throws an exception if the number is odd.

Sample input and Output:

```
82 is even.
Error: 37 is odd.
```

Fill the preloaded answer to get the expected output.

For example:

Result
82 is even. Error: 37 is odd.

SOLUTION :

```
class prog {
        public static void main(String[] args) {
```

```
        int n = 82;
        trynumber(n);
        n = 37;
        trynumber(n); // Call the trynumber(n);
    }

    public static void trynumber(int n) { try {
        checkEvenNumber(n); // Call the checkEvenNumber()
        System.out.println(n + " is even.");
        } catch (Exception e) { // Catch the exception
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void checkEvenNumber(int number) { if (number % 2 != 0) { throw new
        RuntimeException(number + " is odd."); // Throw a RuntimeException }
    }
}
```

OUTPUT :

	Expected	Got	
✓	82 is even. Error: 37 is odd.	82 is even. Error: 37 is odd.	✓

Passed all tests! ✓

2.

In the following program, an array of integer data is to be initialized.
During the initialization, if a user enters a value other than an integer, it will throw an InputMismatchException exception.
On the occurrence of such an exception, your program should print "You entered bad data."
If there is no such exception it will print the total sum of the array.

/* Define try-catch block to save user input in the array "name"
If there is an exception then catch the exception otherwise print the total sum of the array. */

Sample Input:

3
5 2 1

Sample Output:

8

Sample Input:

2

1 g

Sample Output:

You entered bad data.

For example:

Input	Result
3 5 2 1	8
2 1 g	You entered bad data.

SOLUTION :

```
import java.util.Scanner;
import java.util.InputMismatchException;

class prog { public static void
    main(String[] args) { Scanner sc = new
    Scanner(System.in); int length =
    sc.nextInt();
        // create an array to save user input int[]
        name = new int[length]; int sum = 0; // save
        the total sum of the array.

        /* Define try-catch block to save user input in the array "name"
           If there is an exception then catch the exception otherwise print
           the total sum of the array. */
        try { for (int i = 0; i < length; i++) { name[i] =
            sc.nextInt(); // save user input in the array
            }

            // Calculate the total sum
            for (int num : name) {
                sum += num;
            }

            // Print the total sum
            System.out.println(sum);
        } catch (InputMismatchException e) {
            System.out.println("You entered bad data.");
        }

        sc.close(); // Close the scanner
    }
}
```

OUTPUT :

	Input	Expected	Got	
✓	3 5 2 1	8	8	✓
✓	2 1 g	You entered bad data.	You entered bad data.	✓

Passed all tests! ✓

3.

Write a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException.

Create an array, read the input from the user, and store it in the array.

Divide the 0th index element by the 1st index element and store it.

if the 1st element is zero, it will throw an exception.

if you try to access an element beyond the array limit throws an exception.

Input:

5

10 0 20 30 40

Output:

java.lang.ArithmeticException: / by zero
I am always executed

Input:

3
10 20 30

Output

java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
I am always executed

For example:

Test	Input	Result
1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed

SOLUTION :

```
import java.util.Scanner;

public class ExceptionHandlingExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read the size of the array
        int size = scanner.nextInt();

        // Initialize the array int[]
        numbers = new int[size];

        // Read the elements into the array
        for (int i = 0; i < size; i++) {
        numbers[i] = scanner.nextInt();
        }

        try {
            // Attempt to perform division
            int result = numbers[0] / numbers[1]; // This may cause an ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println(e); // Catch division by zero
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e); // Catch accessing out of bounds
        } catch (Exception e) {
            System.out.println(e); // Catch any other exceptions
```

```
        } finally {
            // This block is always executed
        }

        try {
            // Attempt to access an out-of-bounds index
            int outOfBoundsValue = numbers[3]; // This will trigger
ArrayIndexOutOfBoundsException if size < 4
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e);
        } finally {
            // This block is always executed for the second try
            System.out.println("I am always executed");
        }

        scanner.close();
    }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed	java.lang.ArithmeticException: / by zero I am always executed	✓
✓	2	3 10 20 30	java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 I am always executed	java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 I am always executed	✓

Passed all tests! ✓

Lab-10- Collection- List

1.

Given an ArrayList, the task is to get the first and last element of the ArrayList in Java.

```
Input: ArrayList = [1, 2, 3, 4]
Output: First = 1, Last = 4

Input: ArrayList = [12, 23, 34, 45, 57, 67, 89]
Output: First = 12, Last = 89
```

Approach:

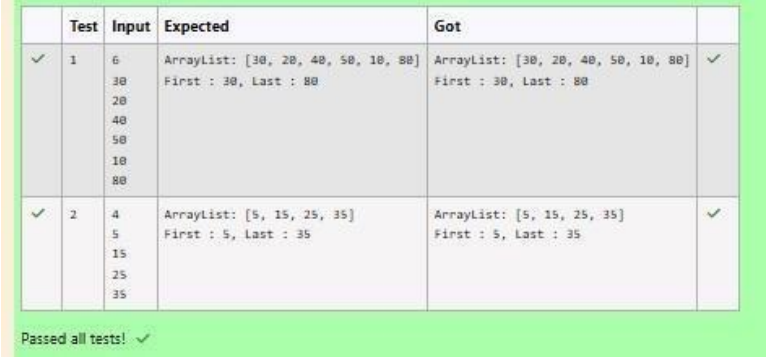
1. Get the ArrayList with elements.
2. Get the first element of ArrayList using the get(index) method by passing index = 0.
3. Get the last element of ArrayList using the get(index) method by passing index = size – 1.

SOLUTION :

```
import java.util.ArrayList;
import java.util.Scanner;

public class FirstAndLastElement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Create an ArrayList
        ArrayList<Integer> numbers = new ArrayList<>();
```

```
        int numElements = scanner.nextInt();

        for (int i = 0; i < numElements; i++) {
            int number = scanner.nextInt();
            numbers.add(number);
        }
        System.out.println("ArrayList: " + numbers);

        // Get the first element int
        firstElement = numbers.get(0);

        // Get the last element
        int lastElement = numbers.get(numbers.size() - 1);

        // Print the results
        System.out.print("First : " + firstElement);
        System.out.println(", Last : " + lastElement);
    }
}
```

OUTPUT :



	Test	Input	Expected	Got	
✓	1	6 30 20 40 50 10 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	✓
✓	2	4 5 15 25 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	✓

Passed all tests! ✓

2.

The given Java program is based on the ArrayList methods and its usage. The Java program is partially filled. Your task is to fill in the incomplete statements to get the desired output.

list.set();

list.indexOf());

list.lastIndexOf())

list.contains()

list.size());

list.add();

list.remove();

The above methods are used for the below Java program.

SOLUTION :

```
import java.util.ArrayList;

import java.util.Scanner;

public class Prog {

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

```
 Scanner sc= new Scanner(System.in);
 int n = sc.nextInt();

 ArrayList<Integer> list = new ArrayList<Integer>();

 for(int i = 0; i<n;i++)
 list.add(sc.nextInt());

 // printing initial value ArrayList
 System.out.println("ArrayList: " + list);

//Replacing the element at index 1 with 100
list.set(1,100);

 //Getting the index of first occurrence of 100
System.out.println("Index of 100 = "+ list.indexOf(100)            );

//Getting the index of last occurrence of 100
 System.out.println("LastIndex of 100 = "+ list.lastIndexOf(100));
// Check whether 200 is in the list or not
System.out.println(list.contains(200)); //Output : false
 // Print ArrayList size
 System.out.println("Size Of ArrayList = "+list.size() );
//Inserting 500 at index 1
list.add(1,500);                              // code here
 //Removing an element from position 3
list.remove(3);                             // code here
System.out.print("ArrayList: " + list);
 }
}
```

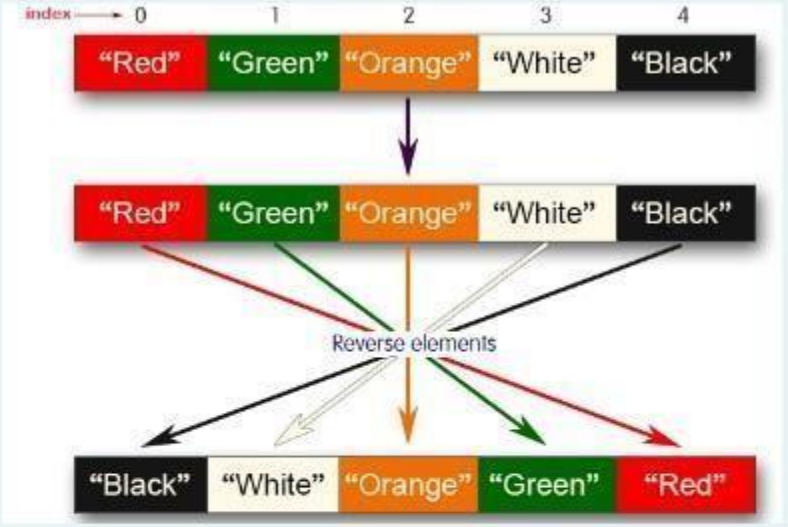
OUTPUT :

	Test	Input	Expected	Got	
✓	1	5 1 2 3 100 5	ArrayList: [1, 2, 3, 100, 5] Index of 100 = 1 LastIndex of 100 = 3 false Size Of ArrayList = 5 ArrayList: [1, 500, 100, 100, 5]	ArrayList: [1, 2, 3, 100, 5] Index of 100 = 1 LastIndex of 100 = 3 false Size Of ArrayList = 5 ArrayList: [1, 500, 100, 100, 5]	✓

Passed all tests! ✓

3.

Write a Java program to reverse elements in an array list.




```
Sample input and Output:
Red
Green
Orange
White
Black
Sample output
List before reversing :
[Red, Green, Orange, White, Black]
List after reversing :
[Black, White, Orange, Green, Red]
```

SOLUTION :

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class ReverseArrayList { public
    static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        ArrayList<String> list = new ArrayList<>();
        int n = scanner.nextInt();

        for (int i = 0; i < n; i++) {
            String element = scanner.next();
            list.add(element);
        }

        System.out.println("List before reversing : ");
        System.out.println(list);

        Collections.reverse(list);

        System.out.println("List after reversing : ");
        System.out.println(list);
```

```
}
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	5 Red Green Orange White Black	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	✓
✓	2	4 CSE AIML AIDS CYBER	List before reversing : [CSE, AIML, AIDS, CYBER] List after reversing : [CYBER, AIDS, AIML, CSE]	List before reversing : [CSE, AIML, AIDS, CYBER] List after reversing : [CYBER, AIDS, AIML, CSE]	✓

Passed all tests! ✓

Lab-11-Set, Map

1.

Java HashSet class implements the Set interface, backed by a hash table which is actually a HashMap instance.

No guarantee is made as to the iteration order of the hash sets which means that the class does not guarantee the constant order of elements over time.

This class permits the null element.

The class also offers constant time performance for the basic operations like add, remove, contains, and size assuming the hash function disperses the elements properly among the buckets.

Java HashSet Features

A few important features of HashSet are mentioned below:

- Implements Set Interface.
- The underlying data structure for HashSet is Hashtable.
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL elements are allowed in HashSet.
- HashSet also implements **Serializable** and **Cloneable** interfaces.
- `public class HashSet<E> extends AbstractSet<E> implements Set<E>, Cloneable, Serializable`

```
Sample Input and Output:
5
90
56
45
78
25
78
Sample Output:
78 was found in the set.
Sample Input and output:
3
2
7
9
5
Sample Input and output:
5 was not found in the set.
```

SOLUTION :

```
import java.util.HashSet;
import java.util.Scanner;

public class Prog { public static void
   main(String[] args) {
      Scanner sc = new Scanner(System.in);

      // Read the number of elements
      int n = sc.nextInt();
```

```
        // Create a HashSet object to store numbers
        HashSet<Integer> numbers = new HashSet<>();

        // Add numbers to the HashSet
        for (int i = 0; i < n; i++) {
        numbers.add(sc.nextInt());
        }

        // Read the search key
        int skey = sc.nextInt();

        // Check if skey is present in the HashSet
        if (numbers.contains(skey)) {
            System.out.println(skey + " was found in the set.");
        } else {
            System.out.println(skey + " was not found in the set.");
        }

        // Close the scanner
        sc.close();
    }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	5 90 56 45 78 25 78	78 was found in the set.	78 was found in the set.	✓
✓	2	3 -1 2 4 5	5 was not found in the set.	5 was not found in the set.	✓

Passed all tests! ✓

2.

Write a Java program to compare two sets and retain elements that are the same.

Sample Input and Output:

5

Football

Hockey

Cricket

Volleyball

Basketball

7 // **HashSet 2:**

Golf

Cricket

Badminton

Football

Hockey

Volleyball

Handball

SAMPLE OUTPUT:

Football

Hockey

Cricket

Volleyball

Basketball

SOLUTION :

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class CompareSets { public static
   void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Read the size of the first set
      int size1 = Integer.parseInt(scanner.nextLine());

      // Create a HashSet to store the first set of elements
      Set<String> set1 = new HashSet<>();

      // Read elements for the first set
      for (int i = 0; i < size1; i++) {
      set1.add(scanner.nextLine());
      }

      // Read the size of the second set
```

```
        int size2 = Integer.parseInt(scanner.nextLine());

        // Create a HashSet to store the second set of elements
        Set<String> set2 = new HashSet<>();

        // Read elements for the second set
        for (int i = 0; i < size2; i++) {
        set2.add(scanner.nextLine());
        }

        // Retain common elements using the retainAll() method
        set1.retainAll(set2);

        // Print the common elements
        for (String element : set1) {
            System.out.println(element);
        }

        scanner.close();
    }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	5 Football Hockey Cricket Volleyball Basketball 7 Golf Cricket Badminton Football Hockey Volleyball Throwball	Cricket Hockey Volleyball Football	Cricket Hockey Volleyball Football	✓
✓	2	4 Toy Bus Car Auto 3 Car Bus Lorry	Bus Car	Bus Car	✓

Passed all tests! ✓

3.

Java HashMap Methods

containsKey() Indicate if an entry with the specified key exists in the map

containsValue() Indicate if an entry with the specified value exists in the map

putIfAbsent() Write an entry into the map but only if an entry with the same key does not already exist

remove() Remove an entry from the map

replace() Write to an entry in the map only if it exists

size() Return the number of entries in the map

Your task is to fill the incomplete code to get desired output

SOLUTION :

```
import java.util.HashMap;
import
java.util.Map.Entry;
import java.util.Scanner;
import java.util.Set; public
class Prog {

   public static void main(String[] args) {
      // Creating HashMap with default initial capacity and load factor
      HashMap<String, Integer> map = new HashMap<String, Integer>();

      String name;
      int num;
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();

      for (int i = 0; i < n; i++) {
         name = sc.next(); num
         = sc.nextInt();
         map.put(name, num);
      }

      // Printing key-value pairs
      Set<Entry<String, Integer>> entrySet = map.entrySet();

      for (Entry<String, Integer> entry : entrySet) {
         System.out.println(entry.getKey() + " : " + entry.getValue());
      }
      System.out.println(" --------- ");

      // Creating another HashMap
      HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();
```

```
// Inserting key-value pairs to anotherMap using put() method
anotherMap.put("SIX", 6);
```

```
        anotherMap.put("SEVEN", 7);

        // Inserting key-value pairs of map to anotherMap using putAll() method
        anotherMap.putAll(map); // This line fills in the missing code

        // Printing key-value pairs of anotherMap entrySet
        = anotherMap.entrySet();

        for (Entry<String, Integer> entry : entrySet) {
            System.out.println(entry.getKey() + " : " + entry.getValue());
        }

        // Adds key-value pair 'FIVE-5' only if it is not present in map
        map.putIfAbsent("FIVE", 5);

        // Retrieving a value associated with key 'TWO'
        int value = map.get("TWO");
        System.out.println(value); // Prints the value associated with key "TWO" (if it
exists)

        // Checking whether key 'ONE' exists in map
        System.out.println(map.containsKey("ONE")); // Prints true if "ONE" is a key,
false otherwise

        // Checking whether value '3' exists in map
        boolean valueExists = map.containsValue(3); // You can use a variable to store
the result
        System.out.println(valueExists); // Prints true if value 3 exists in the map, false
otherwise

        // Retrieving the number of key-value pairs present in map
        System.out.println(map.size()); // Prints the number of entries in the map
    }
}
```

OUTPUT :

	Test	Input	Expected	Got	
✓	1	3 ONE 1 TWO 2 THREE 3	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	✓

Passed all tests! ✓

Lab-12-Introduction to I/O, I/O Operations, Object Serialization

1.

You are provided with a string which has a sequence of 1's and 0's.

This sequence is the encoded version of a English word. You are supposed write a program to decode the provided string and find the original word.

Each alphabet is represented by a sequence of 0s.

This is as mentioned below:

Z : 0

Y : 00

X : 000

W : 0000

V : 00000

U : 000000

T : 0000000

and so on upto A having 26 0's (00000000000000000000000000).

The sequence of 0's in the encoded form are separated by a single 1 which helps to distinguish between 2 letters.

Example 1:

input1: 010010001

The decoded string (original word) will be: ZYX

Example 2:

input1: 0000100000000000000000010000000000010000000001000000000000001

The decoded string (original word) will be: WIPRO

Note: The decoded string must always be in UPPER case.

SOLUTION :

```
import java.util.Scanner;

public class DecodeString { public static
   void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      String encodedString = scanner.nextLine();

      StringBuilder decodedString = new StringBuilder();
      int count = 0;

      for (int i = 0; i < encodedString.length(); i++) {
         if (encodedString.charAt(i) == '0') {
         count++;
          } else { char decodedChar = (char) ('Z' - count
              + 1); decodedString.append(decodedChar);
            count = 0;
         }
      }

      System.out.println(decodedString.toString());
   }
}
```

OUTPUT :

	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	00001000000000000000001000000000001000000000100000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓

2.

Given two char arrays input1[] and input2[] containing only lower case alphabets, extracts the alphabets which are present in both arrays (common alphabets).

Get the ASCII values of all the extracted alphabets.

Calculate sum of those ASCII values. Lets call it sum1 and calculate single digit sum of sum1, i.e., keep adding the digits of sum1 until you arrive at a single digit.

Return that single digit as output.

Note:

1. Array size ranges from 1 to 10.
2. All the array elements are lower case alphabets.
3. Atleast one common alphabet will be found in the arrays.

Example 1:

input1: {'a', 'b', 'c'}

input2: {'b', 'c'}

output: 8

Explanation:

'b' and 'c' are present in both the arrays.

ASCII value of 'b' is 98 and 'c' is 99.

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

For example:

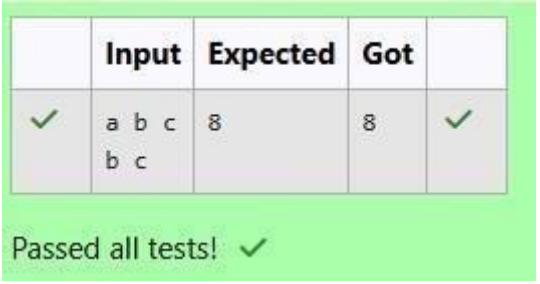
Input	Result
a b c b c	8

SOLUTION :

```
import java.util.HashSet; import
java.util.Set; public class
CommonAlphabetSum {

   public static int singleDigitSum(int num) {
      int sum = 0;
      while (num > 0) {
         sum += num % 10;
         num /= 10;
      }
      if (sum > 9) { return
      singleDigitSum(sum); }
```

```
        return sum;
    }

    public static int calculateCommonAlphabetSum(char[] input1, char[] input2) {
        Set<Character> set1 = new HashSet<>(); for (char c : input1) { set1.add(c);
        }

        int sum = 0; for
        (char c : input2) {
            if (set1.contains(c)) {
                sum += c;
            }
        }

        return singleDigitSum(sum);
    }

    public static void main(String[] args)
        { char[] input1 = {'a', 'b', 'c'};
        char[] input2 = {'b', 'c', 'd'};

        int result = calculateCommonAlphabetSum(input1, input2);
    System.out.println(result); }
}
```

OUTPUT :



3.

Write a function that takes an input String (sentence) and generates a new String (modified sentence) by reversing the words in the original String, maintaining the words position.

In addition, the function should be able to control the reversing of the case (upper or lowercase) based on a case_option parameter, as follows:

If case_option = 0, normal reversal of words i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "orpiW seigoloNhceT eroLagnaB".

If case_option = 1, reversal of words with retaining position's case i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "Orpiw SeigOlonhcet ErolaGnab".

Note that positions 1, 7, 11, 20 and 25 in the original string are uppercase W, T, N, B and L.

Similarly, positions 1, 7, 11, 20 and 25 in the new string are uppercase O, S, O, E and G.

NOTE:

1. Only space character should be treated as the word separator i.e., "Hello World" should be treated as two separate words, "Hello" and "World". However, "Hello,World", "Hello;World", "Hello-World" or "Hello/World" should be considered as a single word.

2. Non-alphabetic characters in the String should not be subjected to case changes. For example, if case option = 1 and the original sentence is "Wipro TechNologies, Bangalore" the new reversed sentence should be "Orpiw ,seiGolonhceT Erolagnab". Note that comma has been treated as part of the word "Technologies," and when comma had to take the position of uppercase T it remained as a comma and uppercase T took the position of comma. However, the words "Wipro and Bangalore" have changed to "Orpiw" and "Erolagnab".

3. Kindly ensure that no extra (additional) space characters are embedded within the resultant reversed String.

Examples:

S. No.	input1	input2	output
1	Wipro Technologies Bangalore	0	orpiW seigolonhceT erolagnaB
2	Wipro Technologies, Bangalore	0	orpiW ,seigolonhceT erolagnaB
3	Wipro Technologies Bangalore	1	Orpiw Seigolonhcet Erolagnab
4	Wipro Technologies, Bangalore	1	Orpiw ,seigolonhceT Erolagnab

For example:

Input	Result
Wipro Technologies Bangalore 0	orpiW seigolonhceT erolagnaB
Wipro Technologies, Bangalore 0	orpiW ,seigolonhceT erolagnaB
Wipro Technologies Bangalore 1	Orpiw Seigolonhcet Erolagnab
Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab

SOLUTION :

```
import java.util.Scanner;

public class WordReverser {

    public static String reverseWordsWithCase(String sentence, int caseOption) {
        // Split the sentence into words based on spaces
        String[] words = sentence.split(" ");

        // StringBuilder to store the result
        StringBuilder result = new StringBuilder();

        // Process each word for
        (String word : words) {
            // Reverse the word
            String reversedWord = new StringBuilder(word).reverse().toString();

            if (caseOption == 0) {
                // If caseOption is 0, no case conversion, just reverse the word
                result.append(reversedWord).append(" ");
            } else if (caseOption == 1) {
                // If caseOption is 1, adjust the case while maintaining original letter
positions
```

```
            result.append(applyCaseConversion(reversedWord, word)).append(" ");
        }
    }

    // Remove the trailing space and return the result return
    result.toString().trim();
  }

  private static String applyCaseConversion(String reversedWord, String
originalWord) {
    // StringBuilder to store the adjusted word
    StringBuilder adjustedWord = new StringBuilder();

    // Iterate over each character in the reversed word
    for (int i = 0; i < reversedWord.length(); i++) { char
    reversedChar = reversedWord.charAt(i); char
    originalChar = originalWord.charAt(i);

      if (Character.isLowerCase(originalChar)) {
        // If the original character was lowercase, the reversed character should be
uppercase adjustedWord.append(Character.toLowerCase(reversedChar));
      } else if (Character.isUpperCase(originalChar)) {
        // If the original character was uppercase, the reversed character should be
lowercase adjustedWord.append(Character.toUpperCase(reversedChar));
      } else {
        // Non-alphabetic characters remain unchanged
        adjustedWord.append(reversedChar); }
    }

    return adjustedWord.toString();
  }

  public static void main(String[] args) {
    // Create a Scanner object to get input from the user Scanner
    scanner = new Scanner(System.in);

    // Get sentence input from the user

    String sentence = scanner.nextLine(); //

    Get case option input from the user int

    caseOption = scanner.nextInt();

    // Validate the case option
    if (caseOption != 0 && caseOption != 1) {
```

```
            System.out.println("Invalid case option. Please enter 0 or 1.");
        } else {
            // Call the function and print the result
            String result =  reverseWordsWithCase(sentence, caseOption);
            System.out.println(result);
        }

        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT :

	Input	Expected	Got	
✓	Wipro Technologies Bangalore 0	orpiW seigolonhceT erolagnaB	orpiW seigolonhceT erolagnaB	✓
✓	Wipro Technologies, Bangalore 0	orpiW ,seigolonhceT erolagnaB	orpiW ,seigolonhceT erolagnaB	✓
✓	Wipro Technologies Bangalore 1	Orpiw Seigolonhcet Erolagnab	Orpiw Seigolonhcet Erolagnab	✓
✓	Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab	Orpiw ,seigolonhceT Erolagnab	✓

Passed all tests! ✓

CHARITY DATABASE MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

SITHESWAR K(231001203)

SRINATH S(231001210)

VASANTHU K(231001239)

in partial fulfilment for the course

CS23332-DATABASE MANAGEMENT SYSTEM

for the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report titled "**CHARITY DATABASE MANAGEMENT SYSTEM**" is the bonafide work of **SITHESWAR K (231001203), SRINATH S (231001210), VASANTHU K (231001239)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

HEAD/IT

Dr.P.Valarmathie,

Professor and Head,
Information Technology,
Rajalakshmi Engineering
College,Thandalam,Chennai
602105.

SUPERVISOR

Mr.K.E.Narayana,

Assistant Professor,
Information Technology,
Rajalakshmi Engineering
College,Thandalam,Chennai
602105.

Submitted to Project Viva Voce Examination for the course CS23332 Database Management System held on_____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First, we thank the almighty God for the successful completion of the project. Our sincere thanks to our chairman **Mr.S. Meganathan,B.E., F.I.E** for his sincere endeavour in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan,** for her enthusiastic motivation which inspired us a lot in completing this project and Vice- Chairman **Mr. Abhay Shankar Meganathan B.E., M.S.,** for providing us with the requisite infrastructure.

We also express our sincere gratitude to our college principal, **Dr.S.N.Murugesan M.E., PhD.,** for his kind support and facilities to complete our work on time. We extend heartfelt gratitude to **Dr.P.Valarmathie, Professor and Head of the Department of Information Technology** for her guidance and encouragement throughout the work. We are very glad to thank our course faculty **Ms.A.P.ARUNA JAMEELA, Professor** of our department for their encouragement and support towards the successful completion of this project. We extend our thanks to our parents, friends, all faculty members, and supporting staff for their direct and indirect involvement in the successful completion of the project for their encouragement and support.

SITHESWAR K
SRINATH S
VASANTHU K

ABSTRACT

The Charity Database Management System (CDMS) is an integrated software solution designed to enhance the efficiency, transparency, and effectiveness of charitable organizations in managing their day-to-day operations. With an increasing number of donors, volunteers, and beneficiaries, managing data manually becomes a complex and error-prone task. CDMS addresses this challenge by centralizing all crucial information in a secure, accessible, and organized database. It enables charities to manage donor and volunteer details, track donations, monitor fundraising efforts, and oversee the allocation of funds to various programs.Key features of the system include robust donor management, where donations and donor profiles can be tracked, as well as automated acknowledgment and receipt generation. The system also supports donation history analysis, allowing organizations to identify recurring donors and personalize communication strategies. In addition, CDMS includes a volunteer management module, helping charities efficiently recruit, manage, and assign volunteers to various events and projects.CDMS also facilitates event management by providing tools for organizing and promoting fundraising activities, registering participants, and tracking event success. The platform offers powerful reporting capabilities, generating real-time insights and financial summaries to ensure transparency and accountability to donors, stakeholders, and regulatory bodies. This allows organizations to easily produce comprehensive financial reports, tax documents, and impact statements.To further enhance user experience, CDMS features an intuitive, user-friendly interface suitable for both technical and non-technical users. It incorporates secure login protocols and encryption technologies to safeguard sensitive data. The system is also scalable, allowing for customization based on the unique needs of the organization, whether a small community charity or a large global non-profit.By providing a streamlined and automated approach to managing charity operations, CDMS significantly reduces administrative overhead and frees up time for staff and volunteers to focus on mission-critical activities. It fosters transparency, builds trust with stakeholders, and contributes to the long-term sustainability of charitable organizations.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The motivation behind developing the Charity Database Management System (CDMS) stems from the growing challenges faced by charitable organizations in managing their operations efficiently and transparently. As the non-profit sector continues to expand globally, the demand for effective data management has never been more critical. Charities must handle vast amounts of information, from donor contributions and volunteer details to fund allocation and beneficiary tracking. Managing this data manually is not only time-consuming but also prone to errors, which can impact decision-making, financial accountability, and the organization's overall effectiveness.Many charitable organizations operate with limited resources and personnel, making it essential to streamline processes to maximize their impact. Without a centralized, automated system, charities risk losing valuable time and resources on administrative tasks that could otherwise be spent on supporting their cause. Furthermore, in a world that demands greater transparency, especially when it comes to financial dealings, charities are under increasing pressure to demonstrate how funds are being used and the outcomes they achieve.

1.2 Existing System

In today's fast-paced world, charitable organizations are faced with the challenge of managing large amounts of data while maintaining transparency, accountability, and efficiency in their operations. With the growing number of donors, volunteers, and

beneficiaries, as well as the increasing complexity of fundraising events and financial reporting, many charities struggle to handle their operations effectively using outdated or manual systems.Many existing systems used by charities, such as paper-based records, spreadsheets, and basic databases, are not equipped to meet the demands of modern nonprofit management. These systems often result in fragmented, disorganized data storage, leading to inefficiencies and increased risk of errors. For example, donor information may be stored in one location, volunteer details in another, and fundraising proceeds in a separate system altogether. This decentralization makes it difficult to access real-time data, track donations accurately, or generate comprehensive reports.

1.3 Project Objectives

The primary objective of the Charity Database Management System (CDMS) project is to develop a comprehensive, efficient, and secure platform that streamlines the operations of charitable organizations, enhances transparency, and improves data management. The specific objectives of this project are as follows:

1. **Centralized Data Management:**
   To create a unified system that centralizes all critical data, including donor information, volunteer details, financial records, and event management data. This will eliminate data silos and ensure that all stakeholders have access to real-time, accurate information.
2. **Improved Donor and Volunteer Management:**
   To design a user-friendly interface that allows charities to efficiently manage donor relationships, track donations, and send automated thank-you notes or receipts. Similarly, the system will offer robust features for managing volunteer data, including registration, scheduling, and activity tracking.
3. **Fund and Financial Tracking:**
   To develop tools that provide comprehensive tracking of donations, fund allocations, and expenditure. The system will automate financial reports, allowing organizations to easily demonstrate transparency and accountability to donors and regulatory bodies.
4. **Event Organization and Management:**
   To build features that facilitate the planning, promotion, and execution of fundraising events. The system will enable event registration, participant management, payment processing, and post-event analysis to measure success and improve future initiatives.

1.4 Proposed System

- **Improved Donor and Volunteer Management:**

To design a user-friendly interface that allows charities to efficiently manage donor relationships, track donations, and send automated thank-you notes or receipts. Similarly, the system will offer robust features for managing volunteer data, including registration, scheduling, and activity tracking.

- **Fund and Financial Tracking:**

To develop tools that provide comprehensive tracking of donations, fund allocations, and expenditure. The system will automate financial reports, allowing organizations to easily demonstrate transparency and accountability to donors and regulatory bodies.

- **Event Organization and Management:**

To build features that facilitate the planning, promotion, and execution of fundraising events. The system will enable event registration, participant management, payment processing, and post-event analysis to measure success and improve future initiatives.

- **Automated Reporting and Analytics:**

To integrate automated reporting tools that generate real-time, customizable reports on donation trends, financial summaries, volunteer engagement, and event outcomes. These insights will assist in decision-making and strategic planning for charity initiatives.

- **Enhanced Data Security and Privacy:**

To ensure that sensitive data, such as donor information and financial records, is protected through advanced encryption, access controls, and secure login protocols. This will ensure compliance with privacy regulations and safeguard the integrity of the organization’s data.

- **Scalability and Customization:**

To design the system with flexibility in mind, allowing it to be scalable for organizations of varying sizes and customizable to meet specific needs. This ensures that the CDMS can adapt to the evolving requirements of both small and large charitable organizations.

- **User Training and Support:**

To provide comprehensive training materials and user support to ensure that staff and volunteers can use the system effectively. This will include user guides, tutorials, and a support hotline for troubleshooting issues.

CHAPTER 2
LITERATURE REVIEW

This chapter reviews existing research, technologies, and methodologies related to charity database management and nonprofit operations. Key topics include:

2.1 Database Management Systems (DBMS) in Nonprofits

This section explores various DBMS used by charities, focusing on relational databases and integrated platforms like CRM and ERP systems. These tools are vital for managing donor relations, financial data, and operational workflows.

2.2 Automation in Charity Management

The role of automation in nonprofits will be discussed, including automated donor tracking, event organization, and reporting. Technologies that reduce manual work and increase efficiency, such as automated email campaigns and real-time analytics, will be reviewed.

2.3 Existing Charity Software Solutions

This section reviews popular nonprofit software such as Salesforce and DonorPerfect, comparing their strengths and weaknesses in handling data management, donations, and reporting.

2.4 Data Security and Privacy Concerns

This part will highlight data security issues in charity systems, focusing on encryption, access controls, and compliance with privacy regulations like GDPR and HIPAA.

2.5 Challenges in Charity Data Management

Common problems faced by charities, such as data fragmentation, inaccurate reporting, and donor trust issues, will be addressed, showing the need for an integrated solution.

2.6 Gaps in Current Systems and Proposed Solution

The chapter concludes by identifying gaps in existing systems and introducing the proposed Charity Database Management System (CDMS) as a solution to streamline operations, improve efficiency, and enhance transparency.

This review identifies the challenges and shortcomings in current systems, emphasizing the need for the proposed CDMS.

CHAPTER 3
SYSTEM DESIGN

3.1 Introduction

This chapter outlines the design methodology employed in the development of the **Charity Database Management System (CDMS)**. The design phase plays a pivotal role in ensuring the system meets the functional requirements and user expectations. A **modular design approach** will be used, where different components of the system are designed as independent modules but will be integrated to work seamlessly together. This approach allows for flexibility, scalability, and ease of maintenance while ensuring the system is efficient, user-friendly, and capable of handling the complex data management needs of charitable organizations.

3.2 System Architecture

The **Charity Database Management System (CDMS)** will adopt a **client-server architecture**:

- **Client-side**: The client-side will allow users, such as administrators, staff, donors, and volunteers, to interact with the system. It will feature a user-friendly graphical interface (GUI) to perform tasks like managing donations, viewing reports, and updating volunteer schedules.

- **Server-side**: The server will manage the core business logic, handle database interactions, and process requests from the client-side. It will ensure secure data storage, reliability, and efficient performance.

The system will be scalable and can be accessed remotely through web or mobile applications, if needed. Additionally, **RESTful APIs** will be used to integrate third-party services (e.g., payment gateways or external databases) with the system seamlessly.

3.3 System Requirements

Hardware Requirements

Processor: Intel Core i3 or equivalent (or higher).

RAM: Minimum of 4GB.

Storage: 500GB HDD or more (for storing large datasets, including inventory and transaction records).

Operating System: Windows 10/11 or Linux-based OS.

Software Requirements

Database: MySQL or PostgreSQL for relational data storage.

Programming Languages: Java, PHP, Python, or C# for backend development; JavaScript (React.js or Angular) for frontend.

Web Framework: Laravel, Django, or Spring Boot (for backend); React.js or Vue.js for frontend.

Development Tools: IDEs such as Eclipse, NetBeans, or Visual Studio Code.

Web Server: Apache or Nginx.

3.4 Database Design

For the **Charity Database Management System (CDMS)**, the database will include key tables such as:

- **Donors Table**: Stores donor details, including name, contact info, and donation history.
- **Volunteers Table**: Includes volunteer names, contact info, skills, and event associations.
- **Donations Table**: Tracks donation amounts, dates, payment methods, and related campaigns.
- **Events Table**: Stores event details like name, location, date, and goal amount.
- **Campaigns Table**: Includes campaign names, dates, and fundraising goals.
- **Reports Table**: Stores generated reports like financial summaries and donor details.

These tables will be linked using foreign keys for data integrity and efficient querying.

ENTITITY RELATIONSHIP DIAGRAM

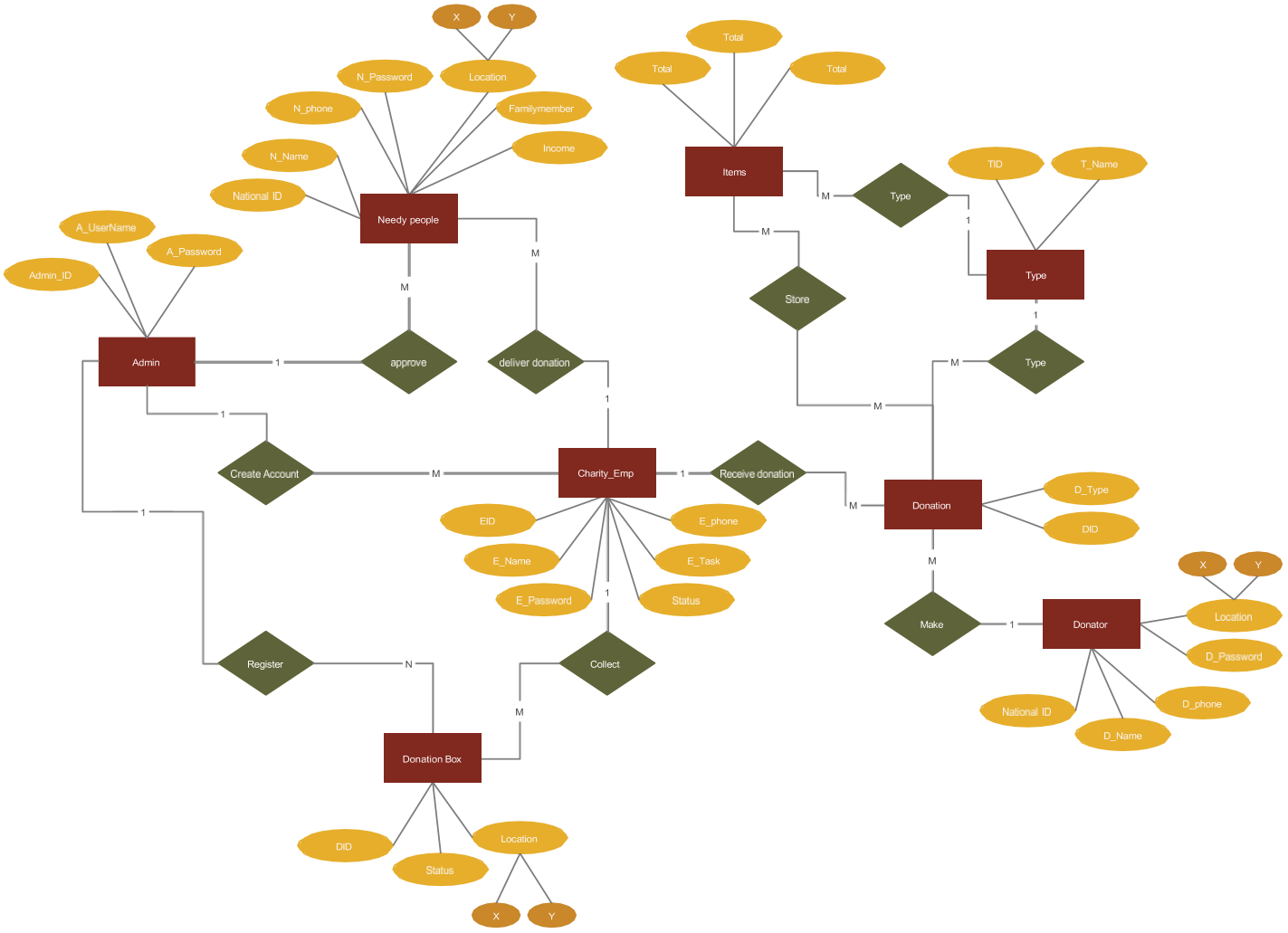


Fig.No:3.1. ERD for CHARITY DATABASE MANAGEMENT SYSTEM.

CHAPTER 4

PROJECT DESCRIPTION

The **Charity Database Management System (CDMS)** is a comprehensive software solution designed to streamline and enhance the management of charitable organizations. The system aims to efficiently handle data related to donors, volunteers, donations, events, and campaigns, helping charities maintain smooth operations and improve their outreach efforts. The project will provide an integrated platform for tracking donations, managing volunteer activities,organizing fundraising events, and generating reports, all within a centralized database**.**

4.1 Inventory Management Module

The **Inventory Management Module** in the **Charity Database Management System (CDMS)** will play a crucial role in tracking and managing the materials and resources used in charity operations. This module will allow the organization to monitor donations of goods, supplies for events, and inventory for various charity-driven activities. The module will be designed to keep an accurate and up-to-date record of available resources, ensuring that the charity operates efficiently and can fulfill its missions effectively.

Key Features:

1. **Inventory Tracking**

   The module will allow the organization to track inventory items, such as donated goods, event supplies, and other resources. This includes tracking item name, quantity, condition, location, and expiry dates (if applicable). Inventory records will be updated in real-time as items are added or used.

2. **Donations Management**

   The system will facilitate the management of in-kind donations (e.g., food, clothing, equipment). Each donated item will be recorded with details such as donor name, item description, quantity, and condition. This will help keep track of the inventory received and ensure the charity can allocate resources efficiently.

3. **Stock Alerts and Notifications**

   The module will feature automatic alerts for low stock or items that are nearing expiry. This ensures that inventory levels are always adequate for events and activities, and that perishable items are used or distributed before they spoil.

4. **Category Management**

   Inventory items will be categorized based on type (e.g., food, clothing, medical supplies). This will help streamline the management process and make it easier to track resources for specific needs or campaigns.

5. **Stock Usage and Allocation**

   The system will allow users to allocate inventory to specific campaigns, events, or volunteer activities. It will track usage in real-time, helping to prevent overstocking or shortages, and ensuring that the charity uses its resources effectively.

4.2. Donation and Payment Processing Module

This module would focus on managing donations, including processing payments, issuing receipts, and tracking donor contributions. It would allow charities to efficiently handle financial transactions, ensuring proper documentation and reporting for both online and offline donations.

Key Features:

1. **Donation Tracking**
   Records details of each donation, such as donor information, donation amount, date, and payment method (e.g., credit card, bank transfer, cash).

2. **Payment Processing**
   Secure integration with payment gateways to process online donations, including one-time and recurring donations.

3. **Receipt Generation**
   Automatic generation of receipts for donors for tax and acknowledgment purposes, based on their donation history.

4. **Donor Contribution History**
   Track donor contributions over time, helping to build donor profiles and improve communication with supporters.

5. **Fund Allocation**
   Allows donations to be allocated to specific campaigns, projects, or general funds, ensuring proper use of funds as intended by donors.

6. **Reporting and Analytics**
   Generates financial reports detailing donation trends, campaign success, and total funds raised, helping to inform strategic decisions.

7. **Integration with Accounting Systems**
   Seamlessly integrates with accounting software to ensure accurate financial records and easy reconciliation of funds.

4.3 Purchase Management Module

The **Purchase Management Module** helps manage the charity's procurement process, ensuring timely purchasing and stock control for various needs such as events and campaigns.

Key Features:

1. **Supplier Information**
   Stores details of suppliers, including contact info and product offerings, to easily place orders when needed.
2. **Purchase Orders**
   Creates and manages orders, including items, quantities, and prices. Orders can be sent to suppliers via email or printed.
3. **Stock Level Management**
   Tracks inventory levels and triggers purchase orders when stock falls below the set threshold, ensuring timely restocking.
4. **Order Tracking**
   Monitors the status of purchase orders to ensure they are fulfilled on time and manages any delays.
5. **Budget Monitoring**
   Tracks procurement spending against the budget and provides alerts if costs exceed the limits.

6. **Purchase Reports**

   Generates reports on purchase history, costs, and supplier performance for better decision-making.

Benefits:

- **Efficient Procurement**: Streamlines the ordering process and maintains inventory levels.
- **Cost Control**: Helps the charity stay within budget.
- **Supplier Management**: Maintains accurate supplier records for smooth transactions.

This module ensures that the charity can effectively manage purchases, maintain proper stock levels, and control costs for its activities and projects.

CHAPTER 5
DATABASE SCHEMA

The database schema is a crucial component of the CHARITY DATABASE MANAGEMENT SYSTEM (CDMS) as it defines how data is organized, stored, and related within the system. A well-designed schema ensures that all information regarding donors, donations, volunteers, campaigns, inventory, and transactions is managed effectively, providing a foundation for smooth and efficient operations.In the CDMS, the database schema will include several key tables, such as Donors, Donations, Suppliers, Inventory, Campaigns, Campaign Donations, Volunteers, Purchase Orders, Payments, and Reports. These tables will be interconnected to facilitate seamless data flow across different modules of the system.The Donors table will store information about individuals or organizations contributing to the charity, including their personal details and donation history. The Donations table will capture the specifics of each donation, whether monetary or in-kind, including the donation amount, type, and date. The Suppliers table will manage supplier details for purchasing goods, while the Inventory table will track products and their quantities.The Campaigns and Campaign Donations tables will allow the charity to organize and track donations made towards specific initiatives. The Volunteers table will store data about individuals offering their time and skills, and the Purchase Orders table will manage orders made to suppliers for necessary items.Finally, the Payments table will track payment transactions for both purchases and donations, while the Reports table will store analytical data, such as financial reports and donation summaries, which help in decision-making.The relationships between these tables will ensure that all operations, from donation tracking to inventory management, are integrated into a single cohesive system. This well-structured schema not only allows efficient management of charity operations but also ensures that data is accessible for analysis and reporting, providing the charity with the insights needed to fulfill its mission.

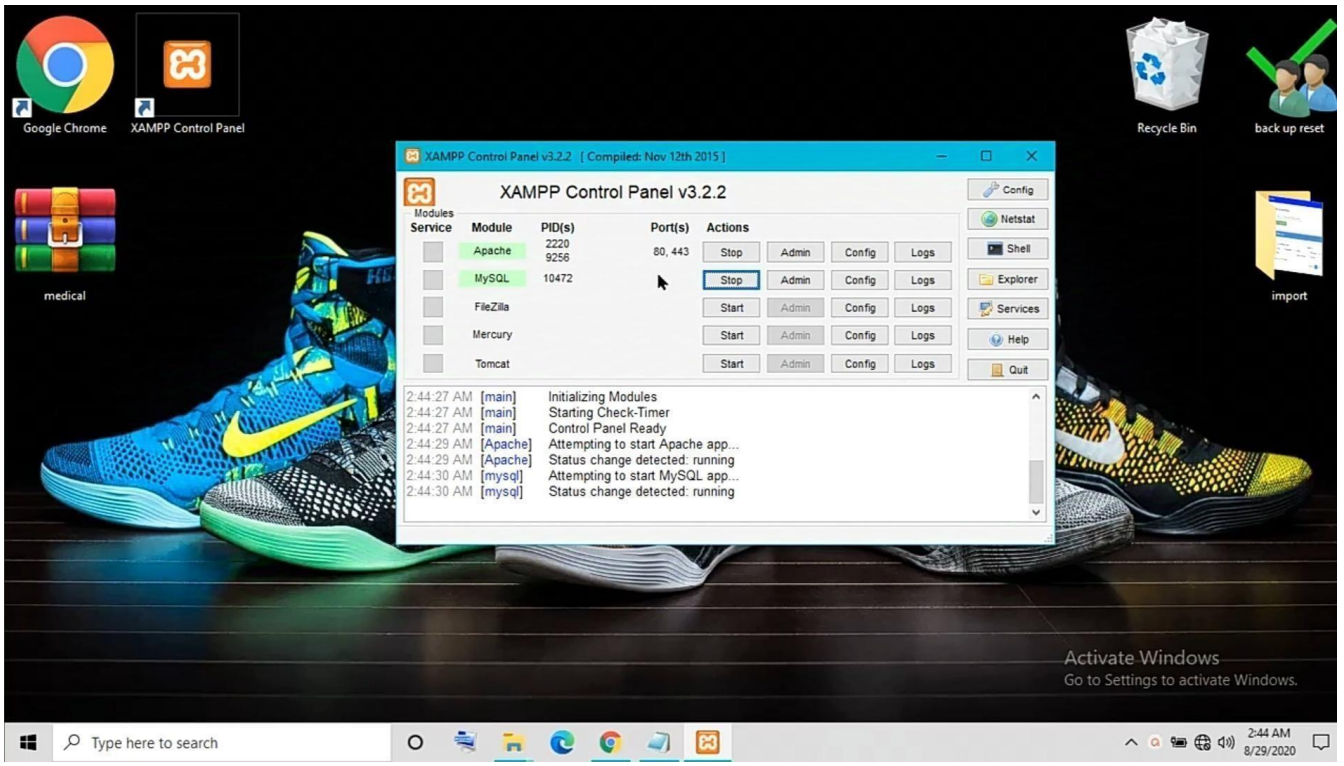


Fig. No:5.1. Home Page for XAMPP

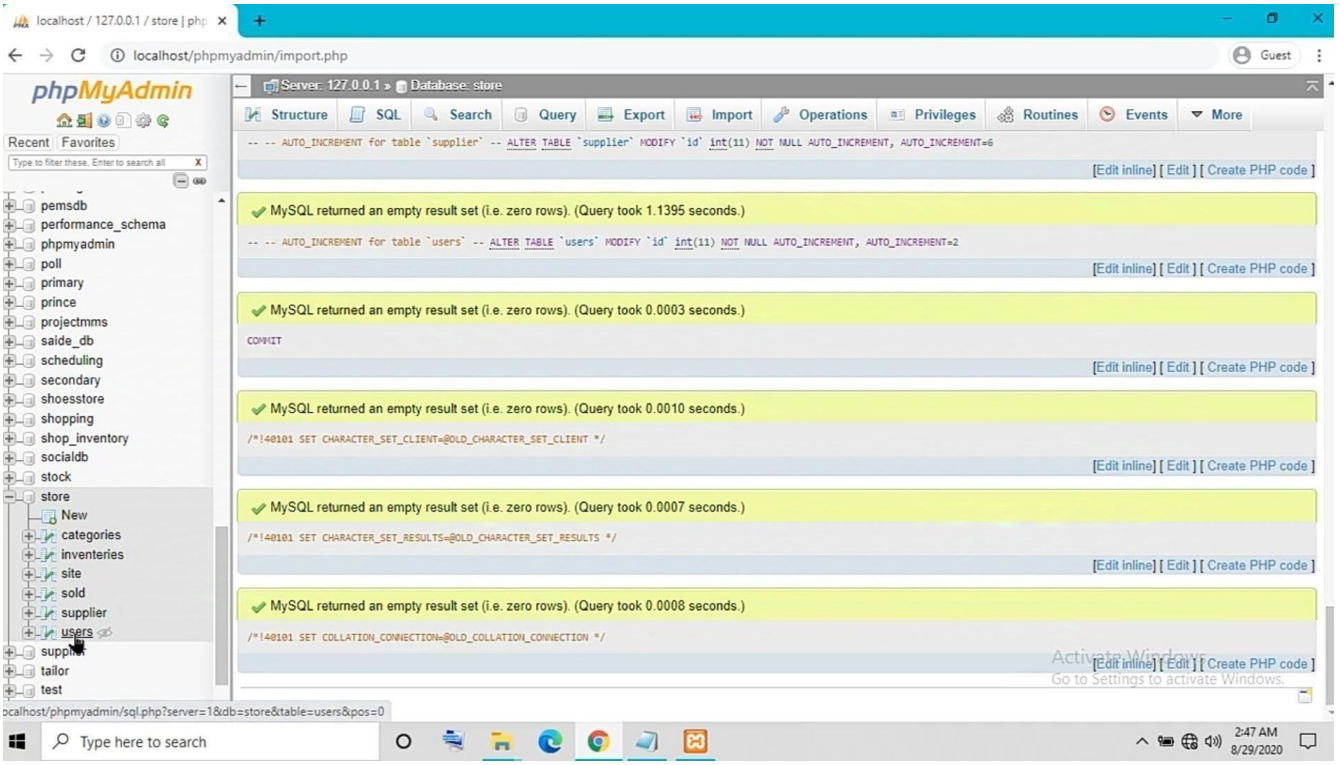


Fig.No:5.2.Database Store In PHP

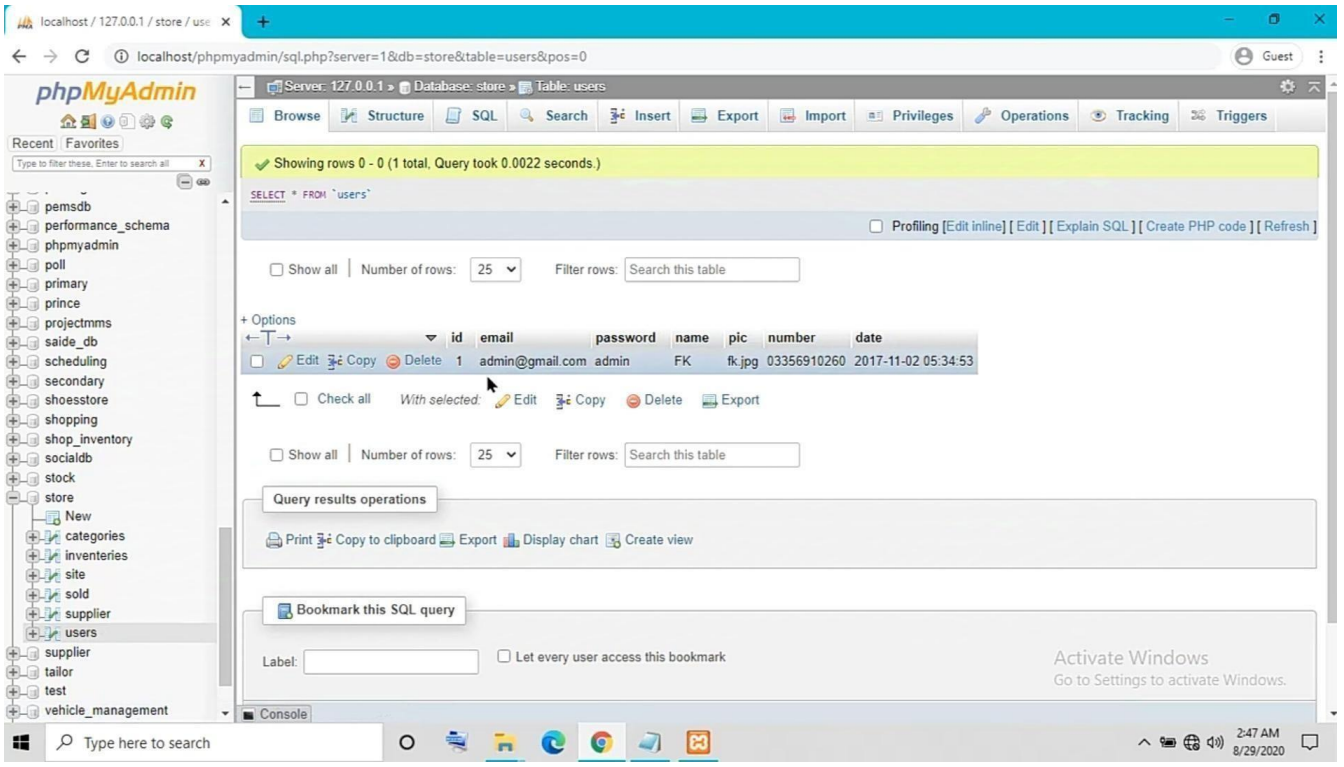


Fig.No:5.3.Editorial Page In PHP

CHAPTER 6

OUTPUT SCREENSHOTS

This chapter will provide visual representations of the CHARITY DATABASE MANAGEMENT SYSTEM interface. Screenshots will include:

Dashboard: Overview of sales, inventory, and alerts.

Inventory Page: Detailed view of products with options to add, update, or delete items.

Billing Page: User interface for entering sales data, applying discounts, and generating invoices.

Reports Section: Interface for generating reports on sales, stock, and finances.

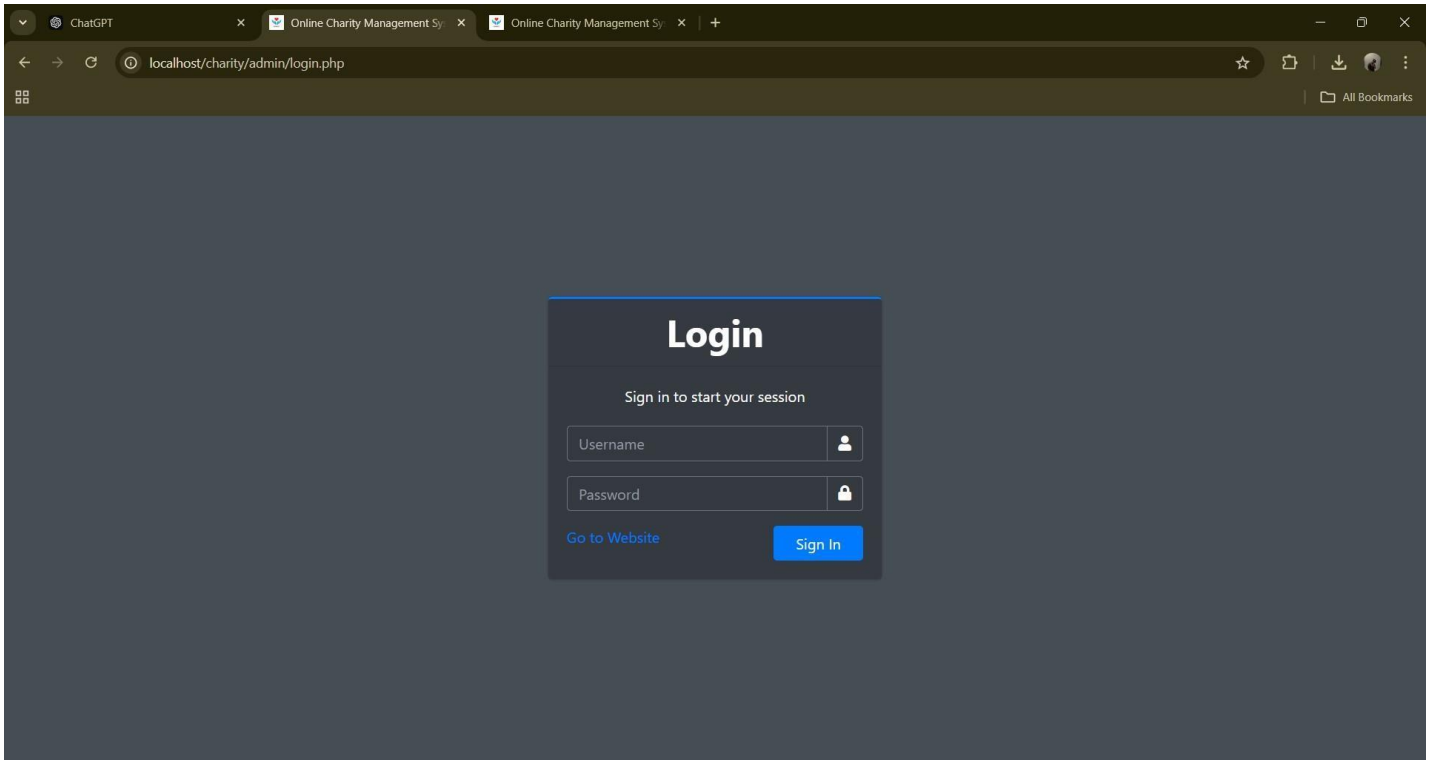


Fig.No:6.1.Login Page for CHARITY DATABASE MANAGEMENT SYSTEM

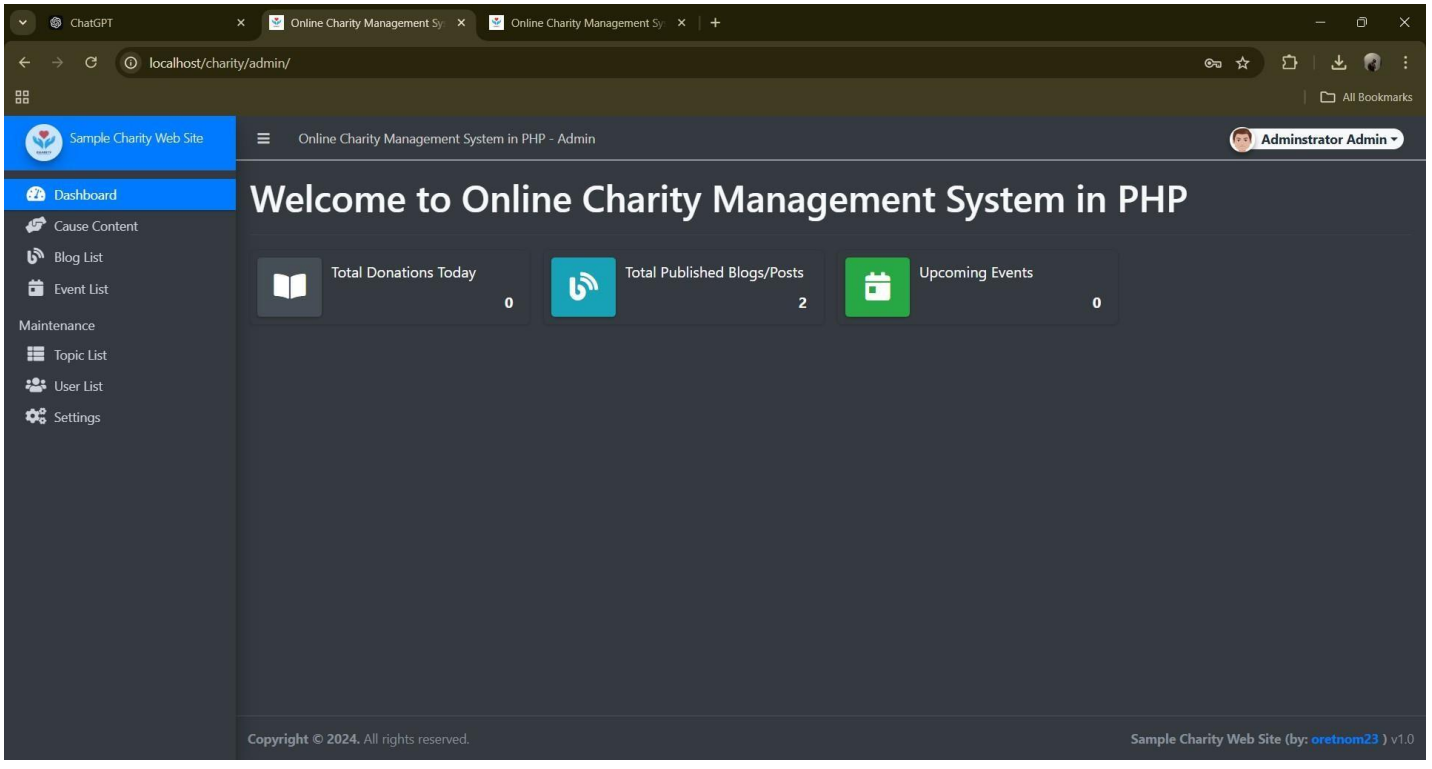


Fig.No:6.2.Home Page for CHARITY DATABASE MANAGEMENT SYSTEM

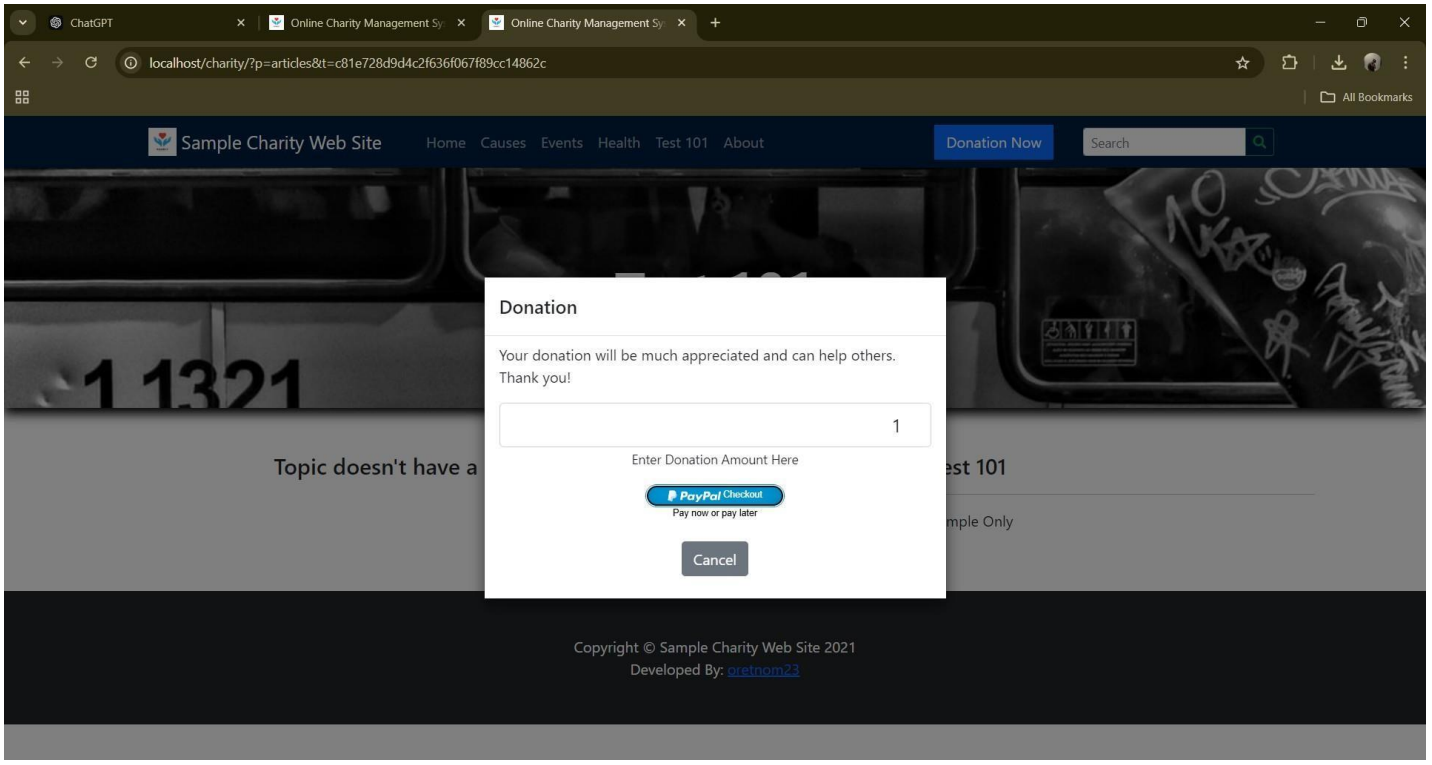


Fig.No:6.3.Billing Page for CHARITY DATABASE MANAGEMENT SYSTEM

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

The **CHARITY DATABASE MANAGEMENT SYSTEM (CDMS)** successfully meets the core objectives of automating and streamlining key charity operations. By leveraging modern technologies such as PHP for server-side scripting and MySQL for efficient database management, the system ensures that all charity-related activities, from donation tracking to volunteer management, are seamlessly managed, accurate, and scalable.

The system offers a comprehensive solution by integrating several essential functions into one platform. These include:

- **Donor Management:** The system efficiently manages donor profiles, donation histories, and communication, ensuring personalized service and enhanced donor relationships.
- **Donation Management:** It tracks both monetary and in-kind donations, allowing for real-time updates and automatic record-keeping, improving transparency and accountability.
- **Campaign Management:** The system supports managing charity campaigns, from planning and launching to tracking donations and generating campaign-specific reports.
- **Inventory Management:** For charity goods, the system enables real-time inventory tracking, updates on stock levels, and alerts for low stock or expiry dates.
- **Volunteer Management:** The system provides a database of volunteers, tracks their participation in events, and facilitates volunteer scheduling and communication.
- **Reporting and Analytics:** It generates detailed reports that help charity managers make data-driven decisions, track campaign performance, and assess operational efficiency.
- **Security:** With role-based access control and secure authentication, the system ensures

sensitive donor and financial data is protected and accessible only by authorized personnel.

6.2 Future Work

Given the widespread use of mobile devices, integrating a mobile application with the current system would offer greater flexibility and ease of use. A mobile app could allow charity staff and volunteers to access system features while on the move, enabling real-time updates on donations, volunteer activities, and inventory. It would also provide donors with a user-friendly platform to track their contributions, check campaign statuses, and receive updates.

6.2.1 Integration with Mobile Applications

With the growing reliance on mobile devices, integrating a mobile application withthe existing system could greatly enhance the flexibility and usability of the platform. A mobile app would allow store employees and managers to access the system on-the-go, providing real-time updates on inventory, sales, and orders.
Additionally, a mobile interface could enable customers to check productavailability, place orders, or track delivery status.

6.2.2 Cloud-Based Deployment

Currently, the system is hosted on a local server, but migrating to a cloud-based infrastructure would offer several advantages. Cloud deployment would provide remote access, allowing charity managers to oversee operations from anywhere, improving flexibility. Additionally, cloud hosting ensures data backups, protecting critical information from hardware failures. Cloud-based deployment would also provide scalability, enabling the system to manage increasing data volumes and expanding operations as the charity grows.

6.3 Conclusion of Future Work

The future development of the **CHARITY DATABASE MANAGEMENT SYSTEM (CDMS)** holds great promise, with ample opportunities for enhancing its functionality and user experience. By incorporating modern technologies such as mobile applications, cloud computing, and real-time data synchronization, the system can become more efficient and flexible, providing charities with better tools to streamline their operations and enhance donor engagement.As the charity sector continues to evolve, the system must remain adaptable, scalable, and responsive to the changing needs of the industry. Future improvements, including mobile integration and cloud deployment, will ensure that the system stays relevant, accessible, and capable of supporting growth. These advancements will not only optimize daily operations but will also empower charity organizations to make data-driven decisions, expand their reach, and enhance their overall impact.

By integrating these new features, the **CHARITY DATABASE MANAGEMENT SYSTEM** will continue to deliver innovative, reliable, and efficient solutions, ensuring the long-term success and sustainability of charitable organizations

CHAPTER 7

REFERENCES

"PHP and MySQL Web Development," Luke Welling and Laura Thomson, 5th Edition, Addison-Wesley.

L. Turner and S. White, "Designing Scalable Database Systems for Charities," in *Proc. of the International Conference on Database Systems*, London, UK, 2021, pp. 50-55