Rajalakshmi Engineeering College

CS23333- OBJECT ORIENTED PROGRAMMING USING JAVA

LAB Record

231201112

WEEK-01

Question **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

Program:

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

	Input	Expected	Got	
	123	2	2	
	456	1	1	

231201112

Question **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

Program:

```
        import java.util.Scanner;

class prog

{

   public static void main(String[] args)

   {

      int a;

      Scanner scanner=new Scanner(System.in);

      int n=scanner.nextInt();

      n=n%10;

      a=Math.abs(n);

      System.out.println(a);

   }
```

231201112

}

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

Question **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

Program:

```
import java.util.Scanner;
class prog
{
   public static void main(String[] args)
   {
```

231201112

```
        Scanner scanner=new Scanner(System.in);
        int n=scanner.nextInt();
        int m=scanner.nextInt();
        int a,b;
        n=n%10;
        m=m%10;
        a=Math.abs(n);
        b=Math.abs(m);
        System.out.println(a+b);
    }
}
```

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

231201112

WEEK-02

Question **1**

Write a Java program to input a number from user and print it into words using for loop. How to display number in words using loop in Java programming.

Logic to print number in words in Java programming.

Example

Input

1234

Output

One Two Three Four

Input:

16

Output:

one six

For example:

Test	Input	Result
1	45	Four Five
2	13	One Three
3	87	Eight Seven

Program:

```
import java.util.Scanner;
public class prog{
   public static void main(String[] args){
      Scanner input=new Scanner(System.in);
      String name="";
      String value=input.nextLine();
      for(int i=0;i<value.length();i++){
         switch((value.charAt(i))){
            case '1':
```

231201112

```
                name="One";
                break;
            case '2':
                name="Two";
                break;
            case '3':
                name= "Three";
                break;
            case '4':
                name="Four";
                break;
            case '5':
                name="Five";
                break;
            case '6':
                name="Six";
                break;
            case '7':
                name="Seven";
                break;
            case '8':
                name="Eight";
                break;
            case '9':
                name="Nine";
                break;
            case '0':
                name="Zero";
                break;
        }
        System.out.print(name+" ");
    }
  }
}
```

Test	Input	Expected	Got	
	1	45	Four Five	Four Five
	2	13	One Three	One Three
	3	87	Eight Seven	Eight Seven

231201112

Question **2**

You have recently seen a motivational sports movie and want to start exercising regularly. Your coach tells you that it is important to get up early in the morning to exercise. She sets up a schedule for you:

On weekdays (Monday - Friday), you have to get up at 5:00. On weekends (Saturday & Sunday), you can wake up at 6:00. However, if you are on vacation, then you can get up at 7:00 on weekdays and 9:00 on weekends.

Write a program to print the time you should get up.

Input Format

Input containing an integer and a boolean value.

The integer tells you the day it is (1-Sunday, 2-Monday, 3-Tuesday, 4-Wednesday, 5-Thursday, 6-Friday, 7-Saturday). The boolean is true if you are on vacation and false if you’re not on vacation.

You have to print the time you should get up.

Example Input:

1 false

Output:

6:00

Example Input:

5 false

Output:

5:00

Example Input:

1 true

Output:

9:00

For example:

Input	Result
`1 false`	`6:00`
`5 false`	`5:00`

231201112

Input	Result
1 true	9:00

Program:

```
import java.util.Scanner;
public class WakeUpTime{
    public static String getwakeUpTime(int day, boolean isVacation){
        if(isVacation){
            if(day==1|| day==7){
                return "9:00";
            }else{
                return "7:00";
            }
        }else{
            if(day==1||day==7){
                return "6:00";
            }else{
                return "5:00";
            }
        }
    }
    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);
        int day=scanner.nextInt();
        boolean isVacation=scanner.nextBoolean();
        System.out.println(getwakeUpTime(day, isVacation));
        scanner.close();
    }
}
```

	Input	Expected	Got	
	1 false	6:00	6:00	
	5 false	5:00	5:00	
	1 true	9:00	9:00	

Question **3**

Write a program that takes as parameter an integer n.

231201112

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

Program:

```
import java.util.Scanner;

public class prog {

    // Function to return trailing
```

231201112

```
    // 0s in factorial of n
    public static int findTrailingZeros(int n)
    {
        int count=0;
        for (int i = 5; n / i >= 1;i*=5){
            count += n / i;
        }
        return count;
    }

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

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

231201112

WEEK-03

Question **1**

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

Program:

```
import java.util.Scanner;


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



        Scanner scan=new Scanner(System.in);
        int n=scan.nextInt();
        int[] arr=new int[n];
        for(int i=0;i<n;i++){
```

```
    arr[i]=scan.nextInt();
}


int maxLen=0,maxsum=0,currLen=0,currsum=0;
boolean haspos=false;
for(int i=0;i<n;i++){
    if(arr[i]>=0){
        haspos=true;
        currLen++;
        currsum+=arr[i];
    }
    else{
        if(currLen>maxLen){
                maxLen=currLen;
                maxsum=currsum;
       }
        else if(currLen==maxLen){
                maxsum+=currsum;
        }

    currLen=0;
    currsum=0;
    }}
    if(currLen>maxLen){
        maxsum=currsum;
    }
```

```
        else if(currLen==maxLen){
            maxsum+=currsum;
        }
        int finalResult=haspos?maxsum:-1;
        System.out.print(finalResult);
    }
}
```

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

Question **2**

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}

231201112

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

231201112

Input	Result
5 10 87 63 42 2	-6699 0 -2088 -3915 -7395
2 -9 9	-162 0

Program:

```
import java.util.Scanner;

public class ArrayOperations{

    public static int[] performOperations(int[] inputArray){

        int maxNumber=Integer.MIN_VALUE;

        for(int num:inputArray){

            if(num>maxNumber){

                maxNumber=num;

            }

        }

        int[] resultArray=new int[inputArray.length];

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

            resultArray[i]=inputArray[i]-maxNumber;

        }

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

            resultArray[i]*=maxNumber;

        }

        return resultArray;

    }
```

231201112

```
    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);
        int numElements=scanner.nextInt();
        int[] inputArray=new int[numElements];
        for(int i=0;i<numElements;i++){
            inputArray[i]=scanner.nextInt();
        }
        int[] resultArray=performOperations(inputArray);
        for(int i=0;i<resultArray.length;i++){
            System.out.print(resultArray[i]);
            if(i<resultArray.length-1){
                System.out.print(" ");
            }
        }
        System.out.println();
        scanner.close();
    }
}
```

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

231201112

Question **3**

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 0th index of the array pick up digits as per below:

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

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

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

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

4th 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}

231201112

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

Program:

```
import java.util.Scanner;

public class DigitSumCalculator{

    public static int[] extractDigits(int[] arr){

        int[] ex=new int[arr.length];

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

            int num=arr[i];

            int pos=i;

            if(num<Math.pow(10,pos)){

                ex[i]=0;

            }else{

                ex[i]=(int)(num/Math.pow(10,pos))%10;

            }

        }
```

231201112

```
    return ex;
}
public static int[] squareDigits(int[] d){
    int[] s=new int[d.length];
    for(int i=0;i<d.length;i++){
        s[i]=d[i]*d[i];
    }
    return s;
}
public static int sumArray(int[] a){
    int sum=0;
    for(int value:a){
        sum+=value;
    }
    return sum;
}
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[] ex=extractDigits(arr);
        int[] s=squareDigits(ex);
        int finalResult=sumArray(s);
        System.out.println(finalResult);
        scanner.close();
    }
}
```

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

231201112

WEEK-04

Question **1**

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 getManufacturer0{

return manufacturer;}

Display the object details by overriding the toString0 method.

For example:

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

Program:

```
class Mobile{
   //private
   private String manufacturer;
   private String operating_system;
```

```
    private int cost;

    //public
    public String color;

    //parameter
    public Mobile(String manufacturer, String operating_system, String color, int cost){
        this.manufacturer=manufacturer;
        this.operating_system=operating_system;
        this.color=color;
        this.cost=cost;
    }

    //getter

    public void setmanufacturer(String manufacturer){
        this.manufacturer=manufacturer;
    }

    public String getManufacturer(){
        return this.manufacturer;
    }

    public void setoperatingSystem(String operating_system){
        this.operating_system=operating_system;
    }

    public String getOperatingSystem(){
        return this.operating_system;
    }

    public void setCost(int cost){
        this.cost=cost;
    }

    public int getCost(){
        return this.cost;
    }

    //string
    @Override
    public String toString(){
        return "manufacturer = "+manufacturer + "\n" +
```

```
            "operating_system = " + operating_system + "\n" +
            "color = " + color + "\n" +
            "cost = " + cost;
    }
}

public class prog{
    public static void main(String[] args){
        //create
        Mobile mobile = new Mobile("Redmi", "Andriod", "Blue", 34000);

        //display
        System.out.println(mobile);
    }
}
```

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

Question **2**

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 = O

Name =Rajalakshmi , Roll no = O

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

Program:

```
class Student{
    private String name;
    private int rollno;

    public Student(){
        this.name=null;
        this.rollno=0;
        System.out.println("No-arg constructor is invoked");
    }

    public Student(String name){
        this.name=name;
```

```
        this.rollno=0;
        System.out.println("1 arg constructor is invoked");
    }

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

    @Override
    public String toString(){
        return "Name =" + (name==null?"null":name) + " , Roll no = " + rollno;
    }
}

public class TestStudent{
    public static void main(String[] args){
        Student student1=new Student();
        Student student2=new Student("Rajalakshmi");
        Student student3=new Student("Lakshmi",101);

        System.out.println(student1);
        System.out.println(student2);
```

```
        System.out.println(student3);

    }

}
```

	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	✓

Question **3**

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 = Ttr2

Circumference = 2Ttr

Input:

2

Output:

Area = 12.57

Circumference -

- 12.57

For example:

Test	Input	Result
1	4	Area = 50.27 Circumference = 25.13

Program:

231201112

```
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 this.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()));

    }
}
```

	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	✓

231201112

231201112

WEEK-05

Question **1**

Create a class Mobile with constructor and a method basicMobile0.

Create a subclass CameraMobile which extends Mobile class , with constructor and a method newFeature0.

Create a subclass AndroidMobile which extends CameraMobile, with constructor and a method androidMobile0.

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

Program:

class Mobile{

```
    public Mobile(){
        System.out.println("Basic Mobile is Manufactured");
    }
    void basicMobile(){

    }
}
class CameraMobile extends Mobile{
    public CameraMobile(){
        System.out.println("Camera Mobile is Manufactured");
    }
    void newFeature(){
        System.out.println("Camera Mobile with 5MG px");
    }
}
class AndroidMobile extends CameraMobile{
    public AndroidMobile(){

        System.out.println("Android Mobile is Manufactured");
    }
    void androidMobile(){
        System.out.println("Touch Screen Mobile is Manufactured");
    }
}
```

231201112

```
class prog{
    public static void main(String[] args){
        AndroidMobile mobile=new AndroidMobile();
        mobile.newFeature();
        mobile.androidMobile();
    }
}
```

	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	✓

Question **2**

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

Create a subclass called SavingsAccount that overrides the withdraw0 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
```

Program:

```
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
    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
        this.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 this.balance;
    }
}

 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);
        }
    }
}

class prog {

    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());
    }
}
```

	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	✓

Question **3**

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 toString0

Expected Output:

231201112

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
```

Program:

```
class College

{

protected   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 depart) {

    super(collegeName);

   // initialize the instance variables

   this.studentName=studentName;

   this.department=depart;




}


public String toString(){

    // return the details of the student

    return "CollegeName : "+collegeName+"\nStudentName : "+studentName +
"\nDepartment : "+department;

}

}

class prog {

public static void main (String[] args) {

        Student  s1 = new Student("REC","Venkatesh","CSE");

        s1.admitted();

                                // invoke the admitted() method

        System.out.println(s1.toString());
```

231201112

}

}

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

Program:

```
import java.util.LinkedHashSet;

import java.util.Scanner;

import java.util.Set;


public class StringProcessor {


    public static String processStrings(String input1, String input2) {
        // Concatenate both strings
        String combined = input1 + input2;


        // Remove spaces and create a set to remove duplicates
        Set<Character> charSet = new LinkedHashSet<>();
        for (char c : combined.toCharArray()) {
            if (c != ' ') {
                charSet.add(c);
            }
        }


        // If the set is empty, return "null"
        if (charSet.isEmpty()) {
            return "null";
```

```
        }

        // Convert set to an array and sort it in descending order
        Character[] uniqueChars = charSet.toArray(new Character[0]);
        java.util.Arrays.sort(uniqueChars, java.util.Collections.reverseOrder());

        // Build the result string
        StringBuilder result = new StringBuilder();
        for (char c : uniqueChars) {
            result.append(c);
        }

        return result.toString();
    }

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

        // Prompt for the first input
        String input1 = scanner.nextLine();

        // Prompt for the second input
        String input2 = scanner.nextLine();

        // Process and display the result
        String result = processStrings(input1, input2);
        System.out.println(result);

        scanner.close();
```

231201112

```
    }
}
```

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

231201112

WEEK-06

Question **1**

Given 2 strings inputl & 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

Program:

```
import java.util.LinkedHashSet;

import java.util.Scanner;

import java.util.Set;


public class StringProcessor {


    public static String processStrings(String input1, String input2) {

        // Concatenate both strings

        String combined = input1 + input2;


        // Remove spaces and create a set to remove duplicates

        Set<Character> charSet = new LinkedHashSet<>();

        for (char c : combined.toCharArray()) {

            if (c != ' ') {

                charSet.add(c);

            }

        }
```

231201112

```
    // If the set is empty, return "null"
    if (charSet.isEmpty()) {
        return "null";
    }

    // Convert set to an array and sort it in descending order
    Character[] uniqueChars = charSet.toArray(new Character[0]);
    java.util.Arrays.sort(uniqueChars, java.util.Collections.reverseOrder());

    // Build the result string
    StringBuilder result = new StringBuilder();
    for (char c : uniqueChars) {
        result.append(c);
    }

    return result.toString();
}

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

    // Prompt for the first input
    String input1 = scanner.nextLine();
```

```
        // Prompt for the second input
        String input2 = scanner.nextLine();

        // Process and display the result
        String result = processStrings(input1, input2);
        System.out.println(result);

        scanner.close();
    }
}
```

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

Question 2

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").

231201112

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"

231201112

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

Program:

```
import java.util.Scanner;


public class MiddleExtractor {


    // Method to process the input string and the 2-digit number

    public static String processInput(String input1, int input2) {

        String[] words = input1.split(" "); // Split input string into words

        int firstWordIndex = input2 / 10 - 1; // Get the index of the first word (0-based)

        int secondWordIndex = input2 % 10 - 1; // Get the index of the second word (0-based)


        // Validate indices
```

231201112

```
        if (firstWordIndex < 0 || firstWordIndex >= words.length || secondWordIndex < 0 ||
secondWordIndex >= words.length) {

            return "Invalid word index in input2";

        }



        // Process both words

        String processedFirstWord = processWord(words[firstWordIndex]);

        String processedSecondWord = processWord(words[secondWordIndex]);



        // Return the processed words separated by a space

        return processedFirstWord + " " + processedSecondWord;

    }



    // Method to process a single word

    private static String processWord(String word) {

        int length = word.length();

        int middleIndex = length / 2; // Find the middle index



        // Extract Middle-to-Begin part

        String middleToBegin;

        if (length % 2 == 0) {

            middleToBegin = new StringBuilder(word.substring(0,
middleIndex)).reverse().toString();

        } else {

            middleToBegin = new StringBuilder(word.substring(0, middleIndex +
1)).reverse().toString();
```

231201112

```
        }

        // Extract Middle-to-End part
        String middleToEnd = word.substring(middleIndex); // From middle to end

        // Combine both parts
        return middleToBegin + middleToEnd;
    }

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

        // Get user input
        String input1 = scanner.nextLine();

        int input2 = scanner.nextInt();

        // Process and display the result
        String result = processInput(input1, input2);
        System.out.println(result);

        scanner.close();
    }
```

231201112

}

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**

Question 3

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:

231201112

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

231201112

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

Program:

```
import java.util.Scanner;


public class WordProcessor {


    public static String processInput(String input1) {
        String[] words = input1.split(":"); // Split the input string by colon
        StringBuilder output = new StringBuilder();


        for (String word : words) { // Corrected for loop syntax
            // Check if the word has at least 2 characters
            if (word.length() < 2) {
                continue; // Skip words that are too short
            }


            char firstChar = word.charAt(0);
```

231201112

```
        char secondChar = word.charAt(1);

        // Check if the first and second characters are the same
        if (firstChar == secondChar) {
            output.append(Character.toUpperCase(firstChar));
        } else {
            // Calculate the positions of the characters
            int pos1 = firstChar - 'a' + 1;
            int pos2 = secondChar - 'a' + 1;
            int diff = Math.abs(pos1 - pos2); // Calculate the absolute difference

            // Ensure diff is within bounds
            char resultChar = (char) ('a' + (diff - 1)); // Find resulting character
            output.append(Character.toUpperCase(resultChar));
        }
    }

    return output.toString(); // Return the processed string
}

public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    String a = scan.nextLine(); // Get user input
    System.out.println(processInput(a)); // Process and print result
```

231201112

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

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

WEEK-07

Question 1

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

Program:

```
import java.util.Scanner;


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");
    }
}
```

```
class Volleyball implements Playable {
    String name;

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

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

class Basketball implements Playable {
    String name;

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

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

231201112

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

        String name = scan.nextLine();
        Football foot = new Football(name);

        name = scan.nextLine();
        Volleyball volley = new Volleyball(name);

        name = scan.nextLine();
        Basketball basket = new Basketball(name);

        // Call the play method for each player
        foot.play();
        volley.play();
        basket.play();

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

231201112

	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	✓

Question 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	Rajalakshmi 22 scored

231201112

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

Program:

```
import java.util.Scanner;
 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);


}
class College implements Football {
   String homeTeam;
   String visitingTeam;


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


  }
231201112
```

```
public void setVisitingTeam(String name){
    visitingTeam=name;
}
public void homeTeamScored(int points){
    System.out.println(homeTeam+" "+points+" scored");
}
public void visitingTeamScored(int points){
  System.out.println(visitingTeam+" "+points+" scored");
}
public void winningTeam(int p1, int p2){
    if(p1>p2)
        System.out.println(homeTeam+" is the winner!");
    else if(p1<p2)
        System.out.println(visitingTeam+" is the winner!");
    else
        System.out.println("It's a tie match.");
}
}
class prog{
    public static void main(String[] args){
        String hname;
        Scanner sc= new Scanner(System.in);
         hname=sc.nextLine();
        String  vteam=sc.next();
```

231201112

```
        int htpoints=sc.nextInt();

        int vtpoints=sc.nextInt();

    College s= new College();

     s.setHomeTeam(hname);

     s.setVisitingTeam(vteam);

     s.homeTeamScored(htpoints);

     s.visitingTeamScored(vtpoints);

     s.winningTeam(htpoints,vtpoints);


}

}
```

	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!	✓

Question 3

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.

Program:

```
interface RBI {

    String parentBank = "RBI"; // This is a constant



    double rateOfInterest(); // Method to get the rate of interest



    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 in 2024.");
    }
}

class SBI implements RBI {
    public double rateOfInterest() {
        return 7.6; // SBI rate of interest
    }
}

class Karur implements RBI {
    public double rateOfInterest() {
        return 7.4; // Karur rate of interest
    }
}

public class Main {
    public static void main(String[] args) {
        SBI sbi = new SBI(); // Corrected instantiation of SBI
        Karur karur = new Karur(); // Corrected instantiation of Karur

        sbi.policyNote(); // Calling default method
```

```
        RBI.regulations(); // Calling static method



        // Print the rates of interest

        System.out.println("SBI rate of interest: " + sbi.rateOfInterest() + " per annum.");

        System.out.println("Karur rate of interest: " + karur.rateOfInterest() + " per annum."); //
Fixed concatenation

    }

}
```

	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.	✓

WEEK-08

Question 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:

231201112

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

Program:

```
import java.util.Scanner;


public class VowelStringExtractor {


    public static String extractVowelStrings(int n, String[] strings) {
        StringBuilder concatenatedResult = new StringBuilder(); // To hold concatenated strings


        // Define vowels
        String vowels = "aeiouAEIOU";
```

231201112

```
        // Scan through the array of strings
        for (String str : strings) {
            if (str.length() > 0) { // Check if the string is not empty
                char firstChar = str.charAt(0);
                char lastChar = str.charAt(str.length() - 1);

                // Check if both first and last characters are vowels
                if (vowels.indexOf(firstChar) != -1 && vowels.indexOf(lastChar) != -1) {
                    concatenatedResult.append(str); // Append valid string
                }
            }
        }

        // Check if any valid strings were found
        if (concatenatedResult.length() == 0) {
            return "no matches found"; // Return if no matches found
        } else {
            return concatenatedResult.toString().toLowerCase(); // Return concatenated result in
lowercase
        }
    }

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

        // Read number of elements in the array
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the newline after nextInt
```

```
        // Read the single line of strings
        String inputLine = scanner.nextLine();

        // Split the input line into an array of strings
        String[] strings = inputLine.split(" ");

        // Ensure we only take n strings if more were provided
        if (strings.length > n) {
            String[] temp = new String[n];
            System.arraycopy(strings, 0, temp, 0, n);
            strings = temp; // Trim the array to n elements
        }

        // Extract the vowel strings and print the result
        String result = extractVowelStrings(n, strings);
        System.out.println(result);

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

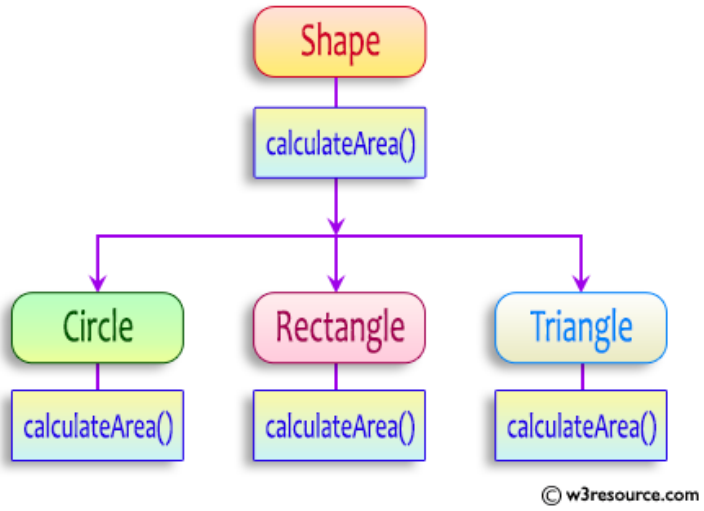
	Input	Expected	Got	
✓	3 oreo sirish apple	oreoapple	oreoapple	✓
✓	2 Mango banana	no matches found	no matches found	✓
✓	3 Ate Ace Girl	ateace	ateace	✓

Question **2**

231201112

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:

231201112

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

Program:

```
import java.util.Scanner;

// Abstract base class
abstract class Shape {
    public abstract double calculateArea(); // Abstract method for area calculation
}

// Circle subclass
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 a circle: π * r * r
    }
}
```

231201112

```
// Rectangle subclass
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 a rectangle: length * breadth
    }
}

// Triangle subclass
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 a triangle: 0.5 * base * height
231201112
```

```
    }
}

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

        // Using a try-catch to handle potential input issues
        try {
            for (int i = 0; i < 2; i++) { // Loop for two sets of inputs
                // Read inputs line by line
                double radius = Double.parseDouble(scanner.nextLine());

                double length = Double.parseDouble(scanner.nextLine());

                double breadth = Double.parseDouble(scanner.nextLine());

                double base = Double.parseDouble(scanner.nextLine());

                double height = Double.parseDouble(scanner.nextLine());

                // Creating objects for each shape
                Circle circle = new Circle(radius);
                Rectangle rectangle = new Rectangle(length, breadth);
                Triangle triangle = new Triangle(base, height);

                // Calculating and displaying areas
                System.out.printf("Area of a circle: %.2f%n", circle.calculateArea());
                System.out.printf("Area of a Rectangle: %.2f%n", rectangle.calculateArea());
```

```
            System.out.printf("Area of a Triangle: %.2f%n", triangle.calculateArea());
        }
    } catch (NumberFormatException e) {
        //System.out.println("Invalid number format: " + e.getMessage());
    } catch (Exception e) {
        //System.out.println("Error reading input: " + e.getMessage());
    } finally {
        scanner.close(); // Close the scanner
    }
  }
}
```

	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	✓

Question 3

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.");
}
```

231201112

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.

Program:

```
class FinalExample {
   // Final variable
   final int maxSpeed = 120; // Changed to final to prevent modification

   // Final method
   public final void displayMaxSpeed() { // Marked as final to prevent overriding
      System.out.println("The maximum speed is: " + maxSpeed + " km/h"); // Corrected string concatenation
   }
}

class Subclass extends FinalExample {
   // Cannot override the final method displayMaxSpeed()
   // public void displayMaxSpeed() {
```

```
    //     System.out.println("Cannot override a final method"); // This is commented out since we can't override a final method
    // }

    // You can create new methods here
    public void showDetails() {
        System.out.println("This is a subclass of FinalExample.");
    }
}

public class Prog { // Ensure the class name is capitalized
    public static void main(String[] args) {
        FinalExample obj = new FinalExample();
        obj.displayMaxSpeed(); // Correctly calls the method

        Subclass subObj = new Subclass();
        subObj.showDetails(); // Correctly calls the new method
    }
}
```

	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.	✓

WEEK-09

Question 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.

Program:

```
public class OddEvenCheck {

    // Custom exception for odd numbers
    static class OddNumberException extends Exception {
        public OddNumberException(String message) {
            super(message);
        }
    }

    public static void checkEven(int number) throws OddNumberException {
        if (number % 2 != 0) {
            throw new OddNumberException(number + " is odd.");
        } else {
            System.out.println(number + " is even.");
```

```
        }
    }

    public static void main(String[] args) {
        // Sample numbers to check
        int[] numbers = {82, 37};

        for (int number : numbers) {
            try {
                checkEven(number);
            } catch (OddNumberException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }
}
```

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

Question 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:

231201112

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.

Program:

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


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



        int size = scanner.nextInt();
        int[] numbers = new int[size];
```

231201112

```
        try {


            for (int i = 0; i < size; i++) {
                numbers[i] = scanner.nextInt(); // This line can throw
InputMismatchException
            }


            // Calculate the total sum
            int sum = 0;
            for (int number : numbers) {
                sum += number;
            }


            // Print the total sum
            System.out.println(sum);
        } catch (InputMismatchException e) {
            System.out.println("You entered bad data.");
        } finally {
            scanner.close(); // Close the scanner resource
        }
    }
}
```

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

Question 3

Write a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException.
231201112

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

Program:

```
import java.util.Scanner;


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


      // Read the size of the array
```

231201112

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

        // Declare the array
        int[] array = new int[size];

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

        // Attempt to perform division and access an out-of-bounds index
        try {
            // This will throw an ArithmeticException if array[1] is zero
            int result = array[0] / array[1];
        } catch (ArithmeticException e) {
            System.out.println(e);
        }

        try {
            // This will throw an ArrayIndexOutOfBoundsException if size <= 3
            int outOfBoundsValue = array[3];
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e);
        }

        // This will always execute
        System.out.println("I am always executed");

        scanner.close();
```

231201112

```
    }
}
```

	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	✓

231201112

WEEK-10

Question 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.**

Program:

```
import java.util.ArrayList;

import java.util.Scanner;


public class Main {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      int count = scanner.nextInt();


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

      for (int i = 0; i < count; i++) {

         list.add(scanner.nextInt());

      }


      printArrayListDetails(list);

   }
```

231201112

```
    public static void printArrayListDetails(ArrayList<Integer> list) {
        if (list.isEmpty()) {
            System.out.println("The ArrayList is empty.");
            return;
        }

        System.out.println("ArrayList: " + list);

        int first = list.get(0);
        int last = list.get(list.size() - 1);

        System.out.println("First : " + first + ", Last : " + last);
    }
}
```

	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	✓

Question 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());

231201112

list.lastIndexOf())

list.contains()

list.size());

list.add();

list.remove();

The above methods are used for the below Java program.

Program:

```
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());
      System.out.println("ArrayList: " + list);
      list.set(1, 100);
      System.out.println("Index of 100 = " + list.indexOf(100));
      System.out.println("LastIndex of 100 = " + list.lastIndexOf(100));
      System.out.println(list.contains(200));
      System.out.println("Size Of ArrayList = " + list.size());
      list.add(1, 500);
      list.remove(3);
      System.out.println("ArrayList: " + list);
```

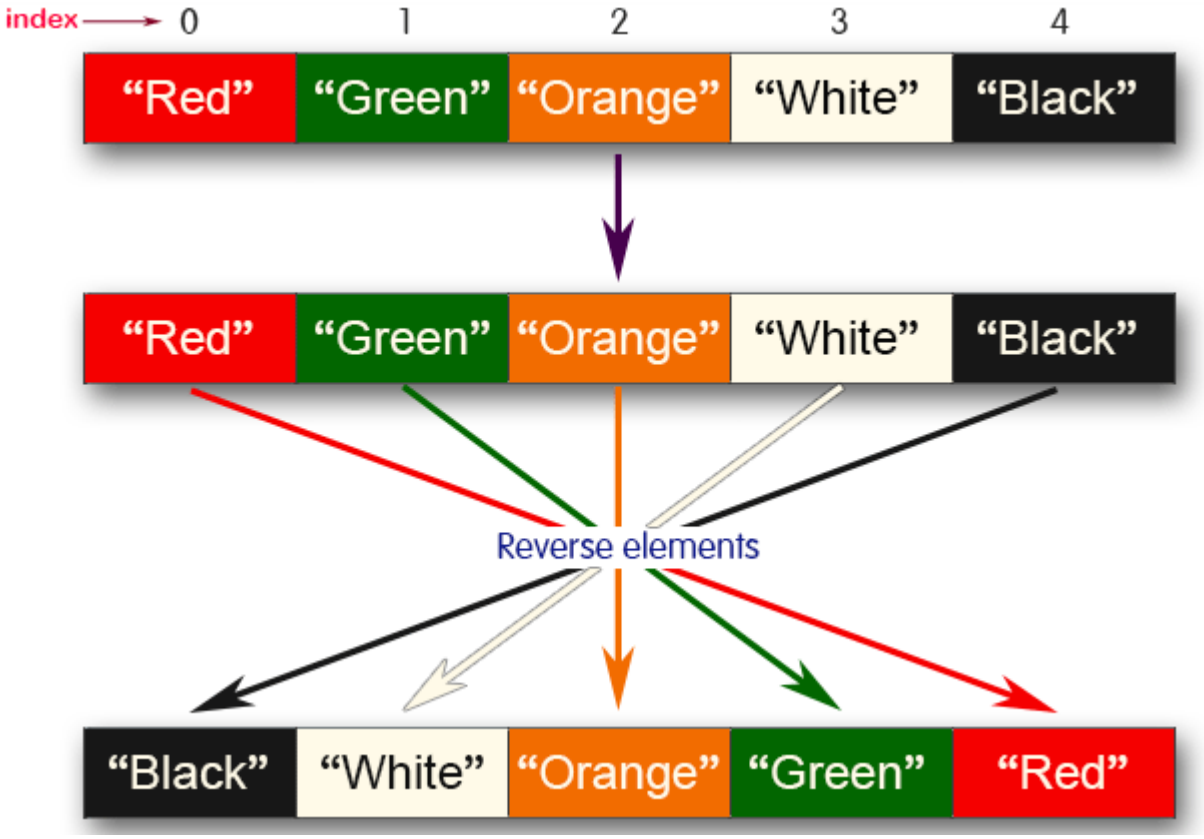
231201112

```
    }
}
```

	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]	✓

Question 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 :

231201112

[Red, Green, Orange, White, Black]

List after reversing :

[Black, White, Orange, Green, Red]

Program:

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

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<String> list = new ArrayList<>();
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            list.add(sc.next());
        }
        System.out.println("List before reversing :");
        System.out.println(list);
        Collections.reverse(list);
        System.out.println("List after reversing :");
        System.out.println(list);
    }
}
```

	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]	✓

WEEK-11

Question 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

231201112

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.

Program:

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

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

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

        // Initialize HashSet
        HashSet<Integer> set = new HashSet<>();

        // Read elements into the HashSet
        for (int i = 0; i < n; i++) {
            set.add(scanner.nextInt());
        }

        // Read the target element to search
```

231201112

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

        // Check if the element is in the set
        if (set.contains(target)) {
            System.out.println(target + " was found in the set.");
        } else {
            System.out.println(target + " was not found in the set.");
        }

        scanner.close();
    }
}
```

	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.	✓

Question 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:

231201112

Golf

Cricket

Badminton

Football

Hockey

Volleyball

Handball

SAMPLE OUTPUT:

Football

Hockey

Cricket

Volleyball

Basketball

Program:

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

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

        // Read size and elements of the first HashSet
        int n1 = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        HashSet<String> set1 = new HashSet<>();
        for (int i = 0; i < n1; i++) {
            set1.add(scanner.nextLine());
        }
```

231201112

```
        // Read size and elements of the second HashSet
        int n2 = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        HashSet<String> set2 = new HashSet<>();
        for (int i = 0; i < n2; i++) {
            set2.add(scanner.nextLine());
        }

        // Retain only common elements in set1
        set1.retainAll(set2);

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

        scanner.close();
    }
}
```

	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	✓

Question 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

Program:

```
import java.util.HashMap;
```

```
import java.util.Map.Entry;
import java.util.Set;
import java.util.Scanner;

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>();
```

231201112

```
// 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); // code here

// 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.getOrDefault("TWO", -1); // Using getOrDefault for safety
System.out.println(value);

// Checking whether key 'ONE' exist in map
System.out.println(map.containsKey("ONE")); // Filled code here

// Checking whether value '3' exist in map
System.out.println(map.containsValue(3)); // Filled code here

// Retrieving the number of key-value pairs present in map
```

```
        System.out.println(map.size()); // Filled code here


        sc.close();
    }
}
```

	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	✓

WEEK-12

Question 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

231201112

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: 00001000000000000000001000000000001000000000100000000000001

The decoded string (original word) will be: WIPRO

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

For example:

Input	Result
010010001	ZYX
00001000000000000000001000000000001000000000100000000000001	WIPRO

Program:

```
import java.util.Scanner;


public class Decoder {


   // Method to decode the sequence
   public static String decode(String input) {
      // Split the input by '1' to separate the sequences of 0's
      String[] sequences = input.split("1");
```

231201112

```
    // StringBuilder to build the decoded word
    StringBuilder decodedWord = new StringBuilder();

    // Iterate over each sequence
    for (String sequence : sequences) {
      // If the sequence is not empty (it could be empty due to split)
      if (!sequence.isEmpty()) {
        int length = sequence.length();

        // The letter corresponding to the sequence length
        // 'Z' corresponds to length 1, 'Y' to length 2, ..., 'A' to length 26
        char letter = (char) ('Z' - (length - 1));

        // Append the letter to the decoded word
        decodedWord.append(letter);
      }
    }

    return decodedWord.toString();
  }

  public static void main(String[] args) {
    // Example input 1
    Scanner scanner = new Scanner(System.in);
String input1= scanner.nextLine();

    System.out.println(decode(input1)); // Output: ZYX
```

231201112

```
    }
}
```

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

Question 2

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

231201112

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

Program:

```
import java.util.Scanner;


public class SentenceReverser {
```

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

        for (int i = 0; i < words.length; i++) {
            String reversedWord = reverseWord(words[i], caseOption);
            result.append(reversedWord);

            // Add space after each word except the last one
            if (i < words.length - 1) {
                result.append(" ");
            }
        }

        return result.toString();
    }

    private static String reverseWord(String word, int caseOption) {
        StringBuilder reversed = new StringBuilder();

        // Reverse the word
        for (int i = word.length() - 1; i >= 0; i--) {
            reversed.append(word.charAt(i));
        }

        // If caseOption is 1, adjust the case based on the original positions
        if (caseOption == 1) {
            char[] resultChars = reversed.toString().toCharArray();
```

```
            for (int i = 0; i < word.length(); i++) {
                if (Character.isLetter(word.charAt(i))) {
                    if (Character.isUpperCase(word.charAt(i))) {
                        resultChars[i] = Character.toUpperCase(resultChars[i]);
                    } else {
                        resultChars[i] = Character.toLowerCase(resultChars[i]);
                    }
                }
            }
            return new String(resultChars);
        }

        // If caseOption is 0, return the reversed word as is
        return reversed.toString();
    }

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

        // Prompt user for sentence input
        String sentence = scanner.nextLine();

        // Prompt user for case option input
        int caseOption = scanner.nextInt();

        // Output the result
        String result = reverseWords(sentence, caseOption);
        System.out.println(result);
```

231201112

```
        scanner.close();
    }
}
```

	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	✓

Question 3

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.

231201112

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

For example:

Input	Result
a b c b c	8

Program:

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

public class CommonAlphabetSum {

    public static int calculateSingleDigitSum(int sum) {
        // Keep adding digits until the sum is a single digit
        while (sum > 9) {
            int temp = 0;
            while (sum != 0) {
                temp += sum % 10;
                sum /= 10;
            }
            sum = temp;
        }
        return sum;
```

231201112

```
}

public static int findCommonAlphabetSum(char[] input1, char[] input2) {
    // Convert the first input array to a set to get unique characters
    HashSet<Character> set1 = new HashSet<>();
    for (char c : input1) {
        set1.add(c);
    }

    // Sum the ASCII values of characters present in both arrays
    int sum = 0;
    for (char c : input2) {
        if (set1.contains(c)) {
            sum += (int) c;
        }
    }

    // Calculate the single-digit sum
    return calculateSingleDigitSum(sum);
}

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

    // Input for the first array
    String input1Str = scanner.nextLine();
    char[] input1 = input1Str.replace(" ", "").toCharArray();

    // Input for the second array
```

231201112

```
        String input2Str = scanner.nextLine();
        char[] input2 = input2Str.replace(" ", "").toCharArray();

        // Calculate and print the result
        int result = findCommonAlphabetSum(input1, input2);
        System.out.println(result);

        scanner.close();
    }
}
```

	Input	Expected	Got	
✓	a b c b c	8	8	✓

231201112