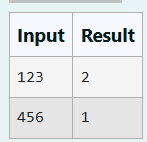
Week1:Q1

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:**



Program:

import java.util.Scanner;

public class ODD{

public static void main(String[] args){

int x;

Scanner s=new Scanner(System.in);

x=s.nextInt();

if(x%2==0)

System.out.println("1");

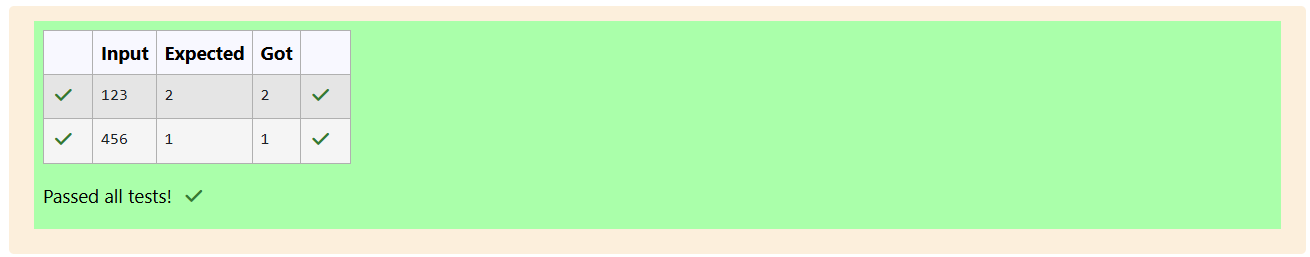
else

System.out.println("2");

}

}

Output:



Week1:Q2

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.

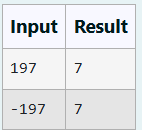
The last digit should be returned as a positive number.

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:**



Program:

import java.util.Scanner;

public class LSD{

public static void main(String[] args){

Scanner s=new Scanner(System.in);

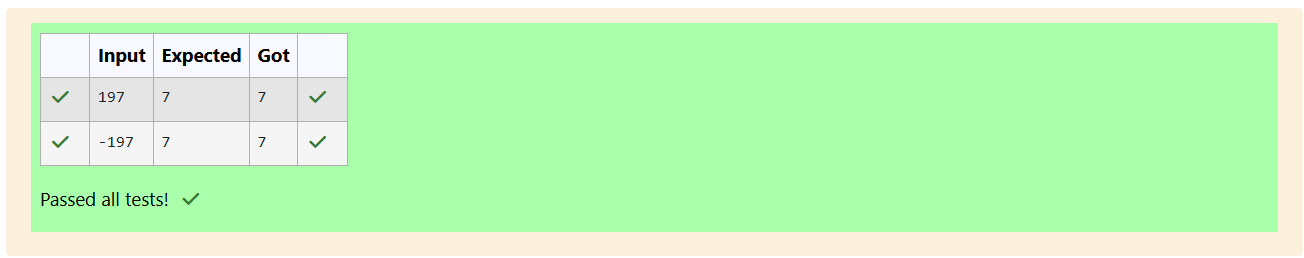
int x=s.nextInt();

System.out.println(Math.abs(x)%10);

}

}

Output:



Week1:Q3

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.

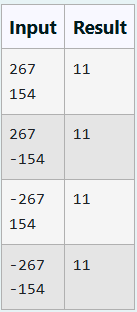
if the input numbers are 267 and 154, the sum of last two digits should be 11

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:**



Program:

import java.util.Scanner;

class prog{

public static void main(String[] args){

Scanner s=new Scanner(System.in);

int n1=s.nextInt();

int n2=s.nextInt();

System.out.println(Math.abs(n1)%10+Math.abs(n2)%10);

}

}

Output:



Week2:Q1

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

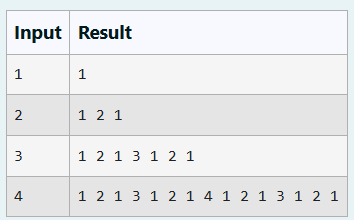
Example Input:

4

Output:

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

**For example:**



Program:

import java.util.Scanner;

public class PrintNOS{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

pat(n);

}

public static void pat(int n){

if(n==1){

System.out.print("1");

}

else{

pat(n-1);

System.out.print(" "+n+" ");

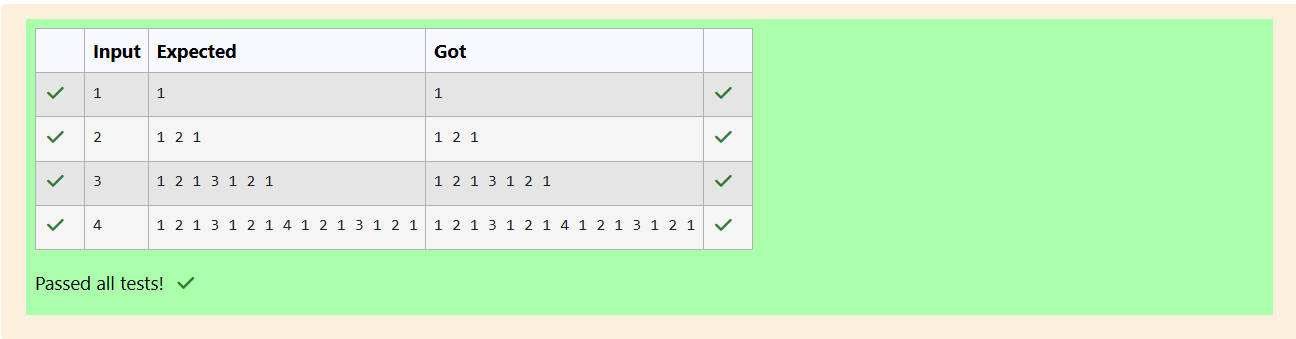
pat(n-1);

}

}

}

Output:



Week2:Q2

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

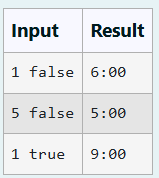
Example Input:

1 true

Output:

9:00

**For example:**



Program:

import java.util.Scanner;

public class Timer{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int i1=sc.nextInt();

//

boolean i2=sc.nextBoolean();

if(i2==false){

if (i1==1||i1==7)

System.out.println("6:00");

else

System.out.println("5:00");

}

else

{

if(i1==1||i1==7)

System.out.println("9:00");

else

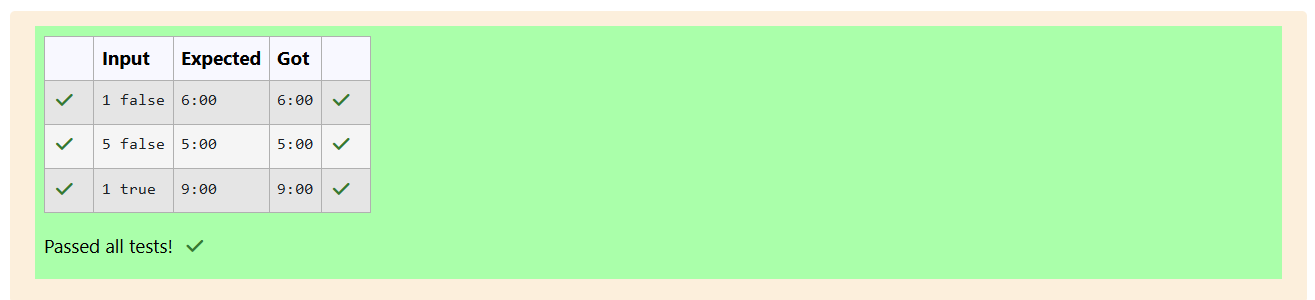
System.out.println("7:00");

}

}

}

Output:



Week2:Q3

You and your friend are movie fans and want to predict if the movie is going to be a hit!

The movie’s success formula depends on 2 parameters:

the acting power of the actor (range 0 to 10)

the critic’s rating of the movie (range 0 to 10)

The movie is a hit if the acting power is excellent (more than 8) or the rating is excellent (more than 8). This holds true except if either the acting power is poor (less than 2) or rating is poor (less than 2), then the movie is a flop. Otherwise the movie is average.

Write a program that takes 2 integers:

the first integer is the acting power

second integer is the critic’s rating.

You have to print Yes if the movie is a hit, Maybe if the movie is average and No if the movie is flop.

Example input:

9 5

Output:

Yes

Example input:

1 9

Output:

No

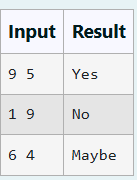
Example input:

6 4

Output:

Maybe

**For example:**



Program:

import java.util.Scanner;

public class moviemeter{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n1=sc.nextInt();

int n2=sc.nextInt();

if(n1>=8){

if(n2>2)

System.out.println("Yes");

else

System.out.println("No");

}

else if(n1<=2){

System.out.println("No");

}

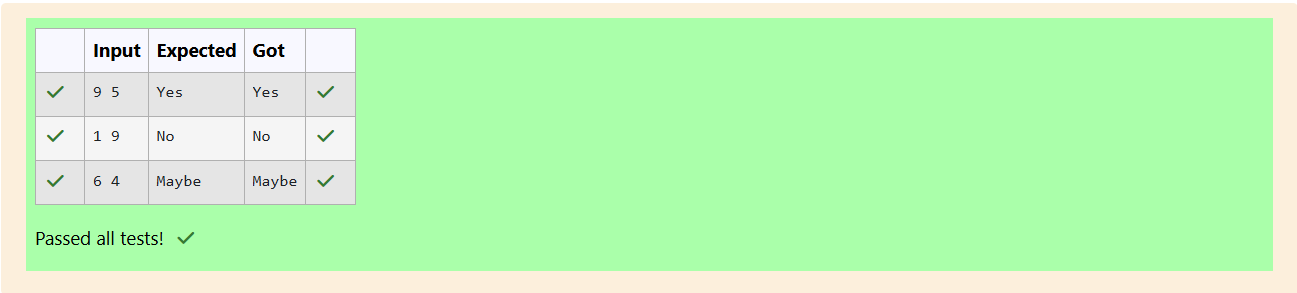
else

System.out.println("Maybe");

}

}

Output:



Week3:Q1

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}

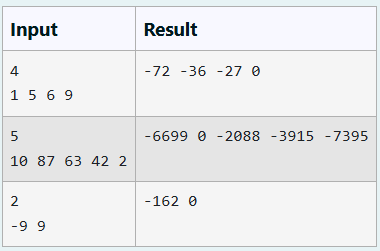
Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-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:**



Program:

import java.util.Arrays;

import java.util.Scanner;

public class ArrayT{

public static int[] Tarray(int[] a){

int maxno=Arrays.stream(a).max().getAsInt();

int[] result=new int[a.length];

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

result[i]=(a[i]-maxno)\*maxno;

}

return result;

}

public static void main(String[] args){

Scanner s=new Scanner(System.in);

int n=s.nextInt();

int[] a=new int[n];

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

a[i]=s.nextInt();

}

int[] result=Tarray(a);

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

System.out.print(result[i]+" ");

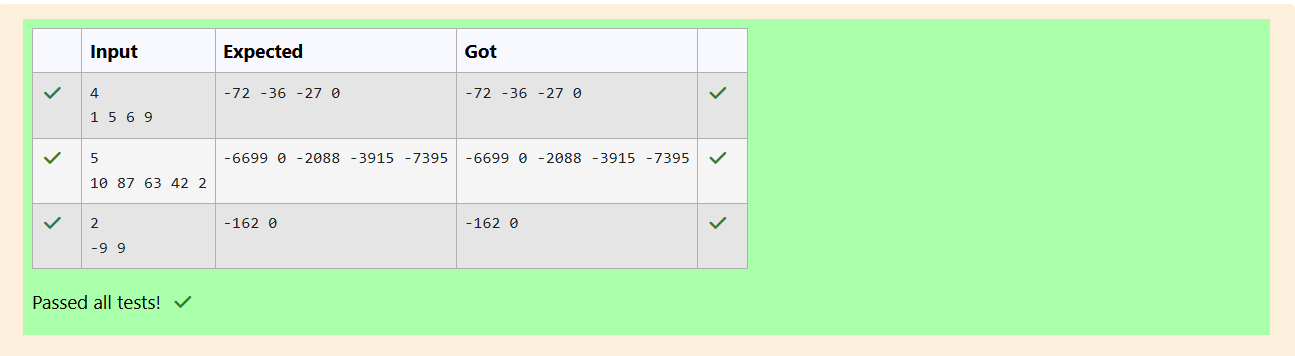
}

s.close();

}

}

Output:



Week3:Q2

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

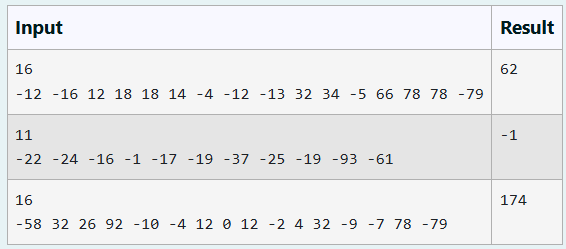
input2 = {-58, 32, 26, 92, -10, -4, 12, 0, 12, -2, 4, 32, -9, -7, 78, -79}

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:**



Program:

import java.util.Scanner;

public class PositiveSeq

{

public static int post(int[] a)

{

int maxL=0,currL=0,currS=0,maxS=0;

for(int num:a)

{

if(num>=0)

{

currL++;

currS+=num;

}

else

{

if(currL>maxL)

{

maxL=currL;

maxS=currS;

}

else if(currL==maxL)

{

maxS+=currS;

}

currL=0;

currS=0;

}

}

if(currL>maxL)

{

maxS=currS;

}

else if(currL==maxL)

{

maxS+=currS;

}

return(maxL>0)?maxS:-1;

}

public static void main(String[] args){

Scanner s=new Scanner(System.in);

int n=s.nextInt();

int [] a=new int[n];

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

a[i]=s.nextInt();

}

int result=post(a);

System.out.println(result);

s.close();

}

}

Output:



Week3:Q3

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}

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.

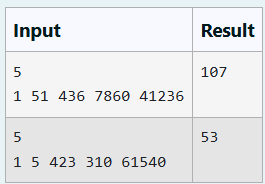
Step 2:

{1, 0, 16, 0, 36}

Step 3:

The final result = 53.

**For example:**



Program:

import java.util.Scanner;

public class CalC{

public static int Sum(int[] a){

int[] d=new int[a.length];

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

int num=a[i];

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

d[i]=num%10;

num/=10;

}

}

int sum=0;

for(int v:d){

sum+=v\*v;

}

return sum;

}

public static void main(String[] args){

Scanner s=new Scanner(System.in);

int n=s.nextInt();

int[] a=new int[n];

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

a[i]=s.nextInt();

}

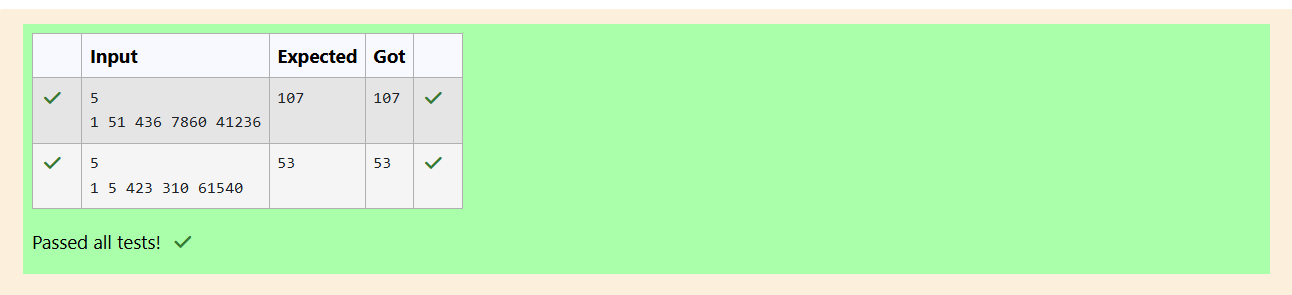
System.out.println(Sum(a));

s.close();

}

}

Output:



Week4:Q1

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)

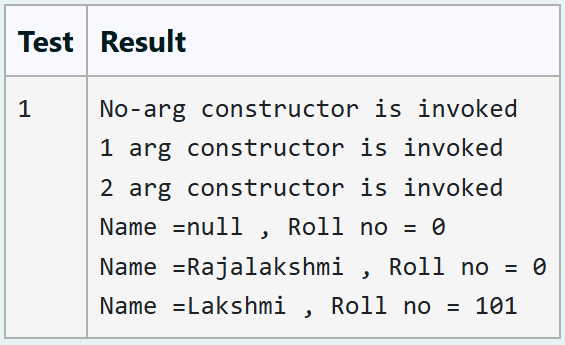
**Input:**

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:**



Program:

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:



Week4:Q2

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;

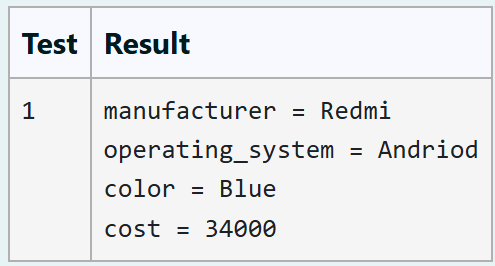
}

String getManufacturer(){

return manufacturer;}

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

**For example:**



Program:

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]+" ");

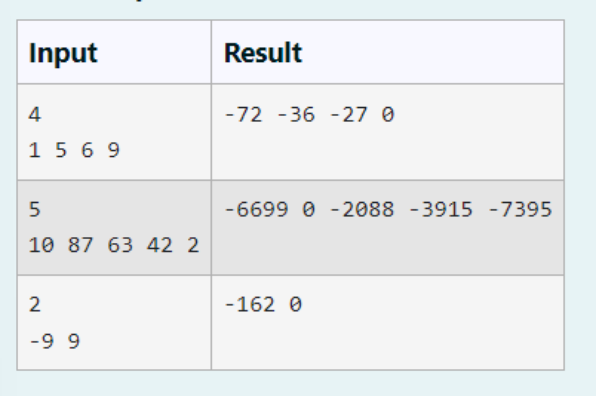
}

scanner.close();

}

}

Output:



Week4:Q3

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 =** **πr2**

**Circumference = 2πr**

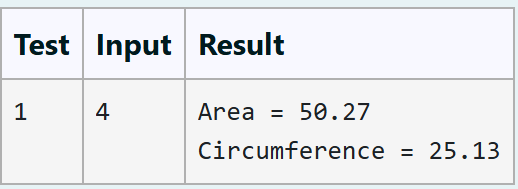
**Input:**

**2**

**Output:**

**Area = 12.57**  
**Circumference = 12.57**

**For example:**



Program:

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

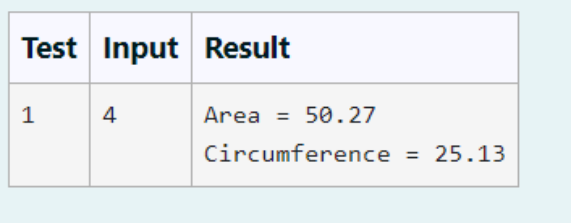
System.out.println("Circumference = "+String.format("%.2f" , c.calculateCircumference()));

sc.close();

}

}

Output:



Week5:Q1

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;

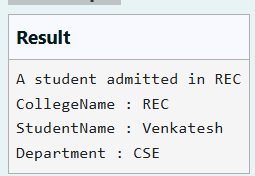
public Student(String collegeName, String studentName,String depart) { }

public toString()

Expected Output:

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

**For example:**



Program:

class College {

public String collegeName;

public College(String collegeName) {

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) {

super(collegeName);

this.studentName = studentName;

this.department = department;

}

public String toString() {

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");

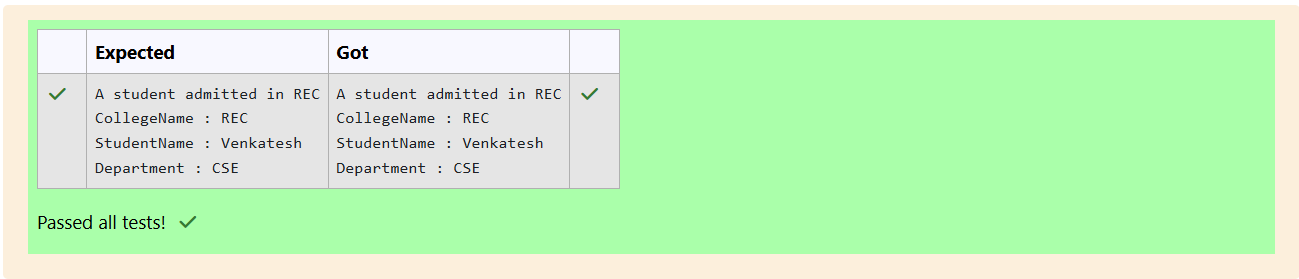
s1.admitted();

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

}

}

Output:



Week5:Q2

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 {

}

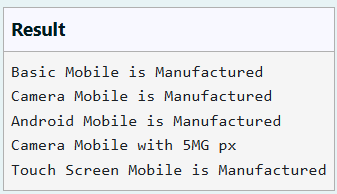
class AndroidMobile extends CameraMobile {

}

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:**



Program:

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();

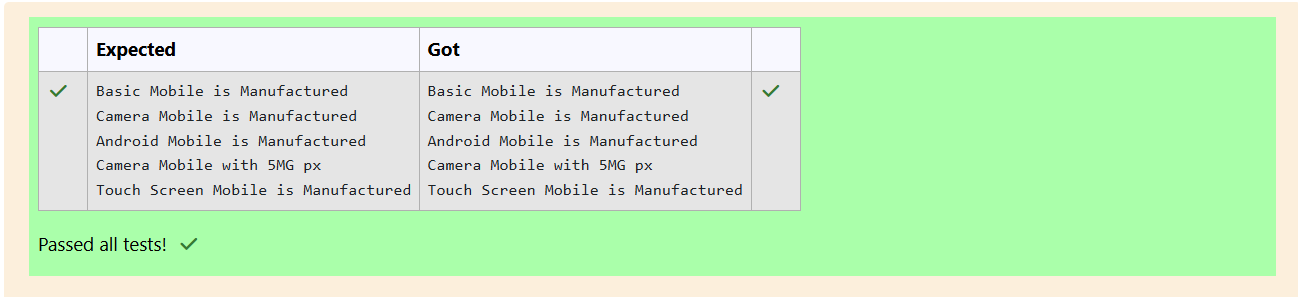
andmob.newm();

andmob.andmob();

}

}

Output:

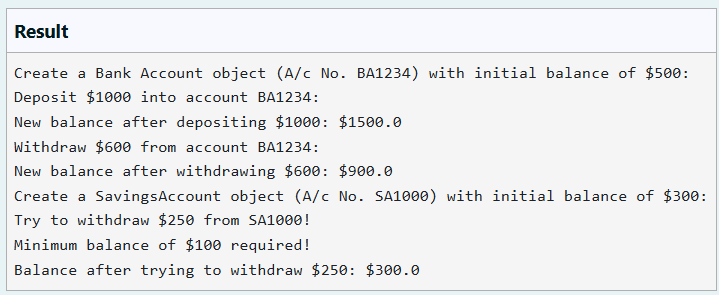


Week5:Q3

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:**



Program:

class BankAccount {

private String accountNumber;

private double balance;

public BankAccount(String accountNumber, double initialBalance) {

this.accountNumber = accountNumber;

this.balance = initialBalance;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

} else {

System.out.println("Insufficient funds!");

}

}

public double getBalance() {

return balance;

}

}

class SavingsAccount extends BankAccount {

public SavingsAccount(String accountNumber, double initialBalance) {

super(accountNumber, initialBalance);

}

@Override

public void withdraw(double amount) {

if (getBalance() - amount >= 100) {

super.withdraw(amount);

} else {

System.out.println("Minimum balance of $100 required!");

}

}

}

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);

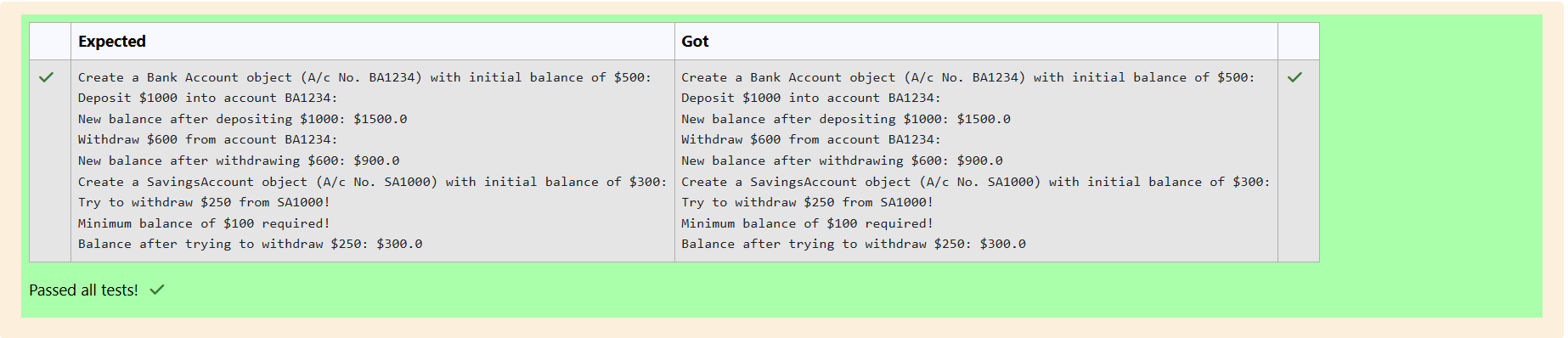
// Print the balance after attempting to withdraw $250

System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance());

}

}

Output:



Week6:Q1

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

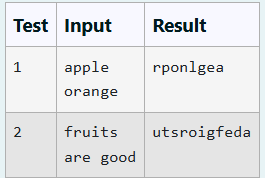
Example 3:

Input 1: ""

Input 2: ""

Output: null

**For example:**



Program:

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(a.charAt(i));

}

}

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

if (b.charAt(i) != ' ') {

ab.append(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]);

}

}

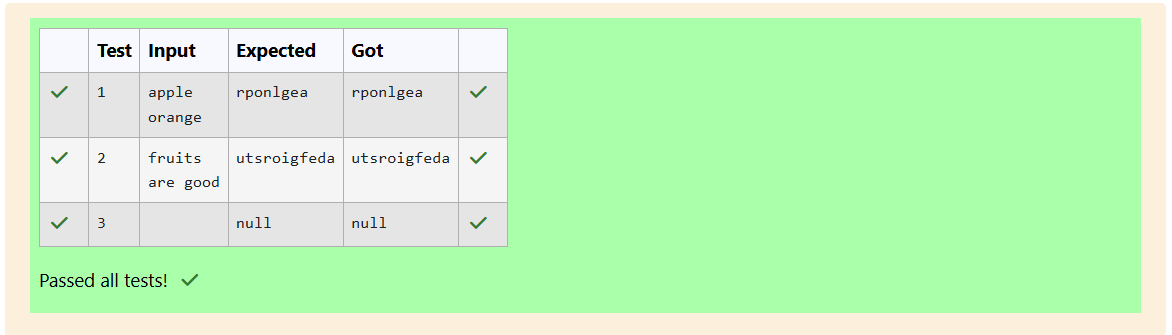
System.out.print(d[0]);

}

}

}

Output:



Week6:Q2

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

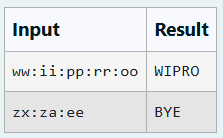
max – min will be 26 – 1 = 25

Alphabet which comes in 25th position is y

word3 is ee, both are same hence take e

Hence the output is BYE

**For example:**



Program:

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))));

}

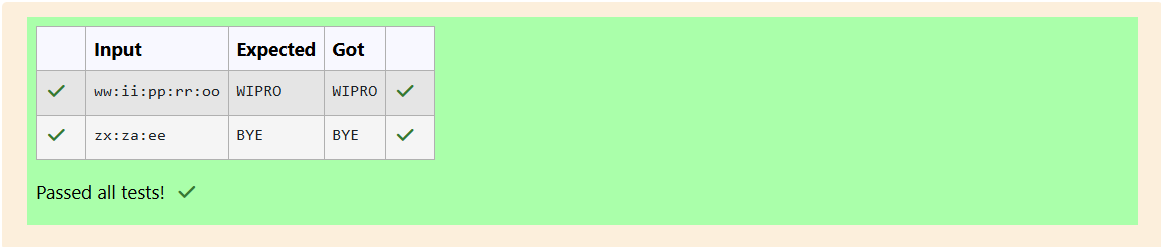
}

System.out.print(ans.toString().toUpperCase());

}

}

Output:



Week6:Q3

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

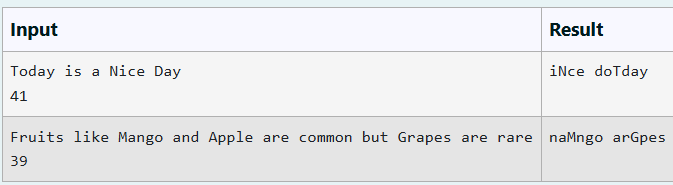
output = "naMngo arGpes"

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:**



Program:

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++;

} 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 = le / 2;

q = le / 2;

} else {

n = (le / 2) - 1;

q = 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)));

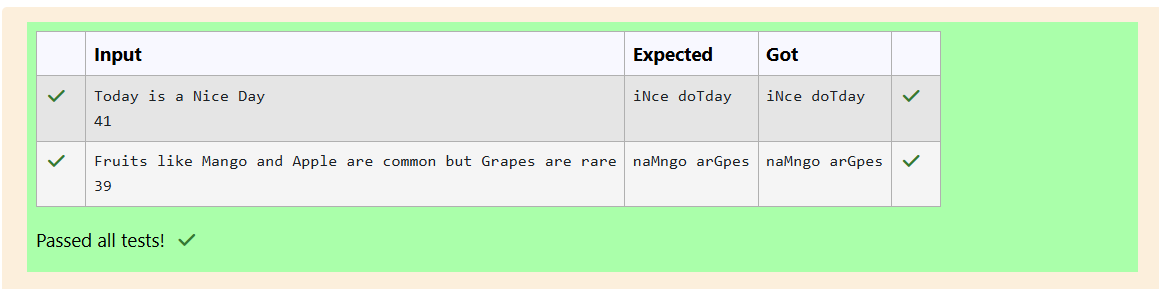
}

return temp3.toString();

}

}

Output:



Week7:Q1

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.");

}

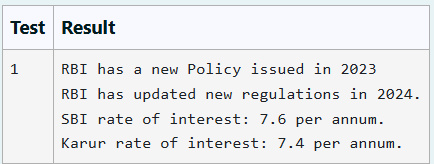
Create two subclasses SBI and Karur which implements the RBI interface.

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:**



Program:

interface RBI {

String parentBank = "RBI";

double rateOfInterest();

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;

}

}

class Karur implements RBI {

public double rateOfInterest() {

return 7.4;

}

}

public class Main {

public static void main(String[] args) {

RBI rbi = new SBI();

rbi.policyNote();

RBI.regulations();

SBI sbi = new SBI();

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

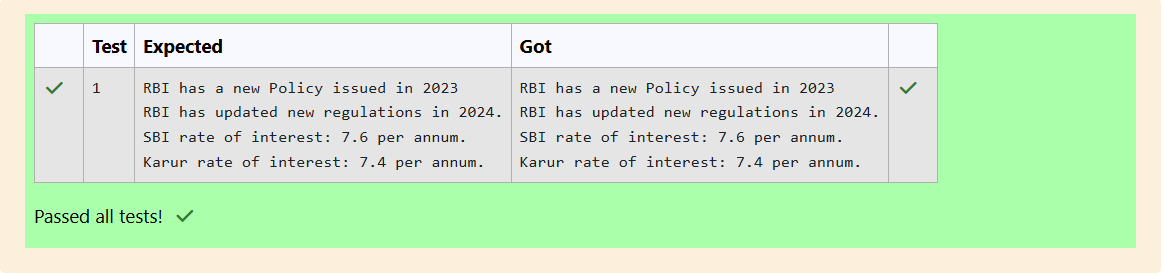
Karur karur = new Karur();

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

}

}

Output:



Week7:Q2

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.

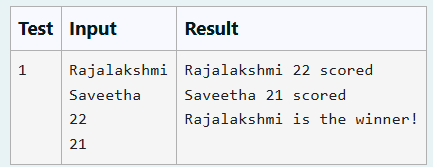
sample Input:

Rajalakshmi  
Saveetha  
22  
21

Output:

Rajalakshmi 22 scored  
Saveetha 21 scored  
Rajalakshmi is the Winner!

**For example:**



Program:

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);

String hname = sc.nextLine();

String vteam = sc.nextLine();

College match = new College();

match.setHomeTeam(hname);

match.setVisitingTeam(vteam);

int htpoints = sc.nextInt();

match.homeTeamScored(htpoints);

int vtpoints = sc.nextInt();

match.visitingTeamScored(vtpoints);

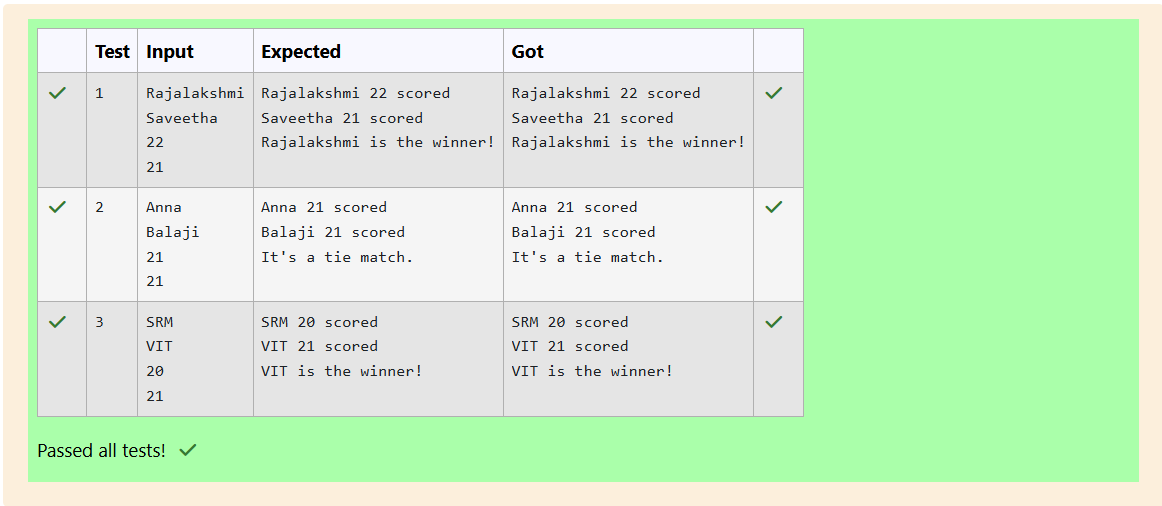
match.winningTeam();

sc.close();

}

}

Output:



Week7:Q3

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();  
}

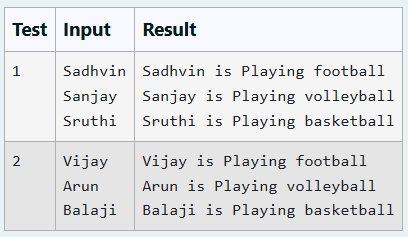
class Football implements Playable {  
 String name;  
 public Football(String name){  
 this.name=name;  
 }  
 public void play() {  
 System.out.println(name+" is Playing football");  
 }  
}

Similarly, create Volleyball and Basketball classes.

**Sample output:**

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

**For example:**



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");

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String footballPlayerName = scanner.nextLine();

Football footballPlayer = new Football(footballPlayerName);

String volleyballPlayerName = scanner.nextLine();

Volleyball volleyballPlayer = new Volleyball(volleyballPlayerName);

String basketballPlayerName = scanner.nextLine();

Basketball basketballPlayer = new Basketball(basketballPlayerName);

footballPlayer.play();

volleyballPlayer.play();

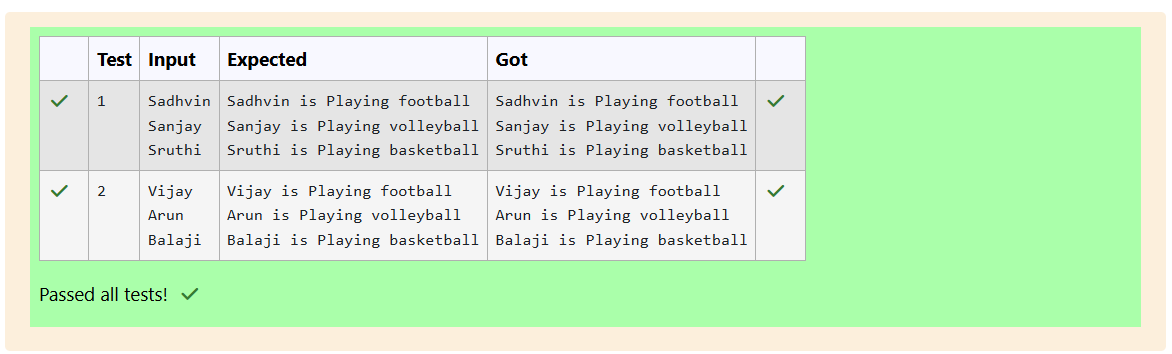
basketballPlayer.play();

scanner.close();

}

}

Output:



Week8:Q1

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.

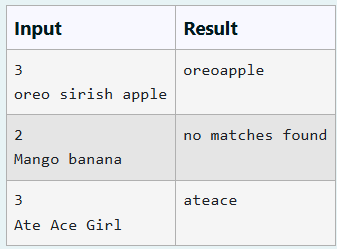
Example 3:

input1: 3

input2: {“Ate”, “Ace”, “Girl”}

output: ateace

**For example:**



Program:

import java.util.Scanner;

public class VowelStringExtractor {

public static String extractVowelStrings(String[] stringArray) {

StringBuilder result = new StringBuilder();

String vowels = "aeiouAEIOU";

for (String s : stringArray) {

if (s.length() > 0 && vowels.indexOf(s.charAt(0)) != -1 && vowels.indexOf(s.charAt(s.length() - 1)) != -1) {

result.append(s);

}

}

return result.length() > 0 ? result.toString().toLowerCase() : "no matches found";

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

scanner.nextLine();

String input = scanner.nextLine();

String[] strings = input.split(" ");

String result = extractVowelStrings(strings);

System.out.println(result);

scanner.close();

}

}

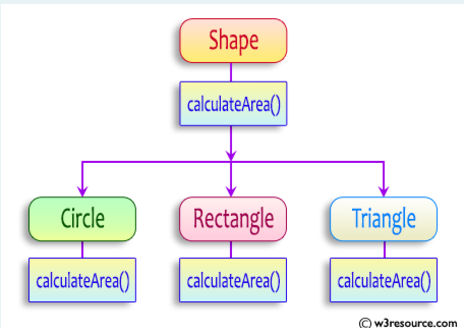
Output:



Week8:Q2

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

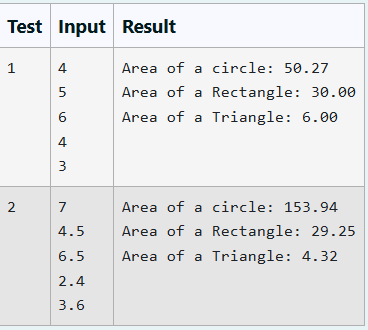
4 // base of the triangle

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:**



Program:

import java.util.Scanner;

abstract class Shape {

public abstract double calculateArea();

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

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;

}

}

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;

}

}

public class ShapeTest {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

double radius = scanner.nextDouble();

Circle circle = new Circle(radius);

System.out.printf("Area of a circle: %.2f%n", circle.calculateArea());

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());

double base = scanner.nextDouble();

double height = scanner.nextDouble();

Triangle triangle = new Triangle(base, height);

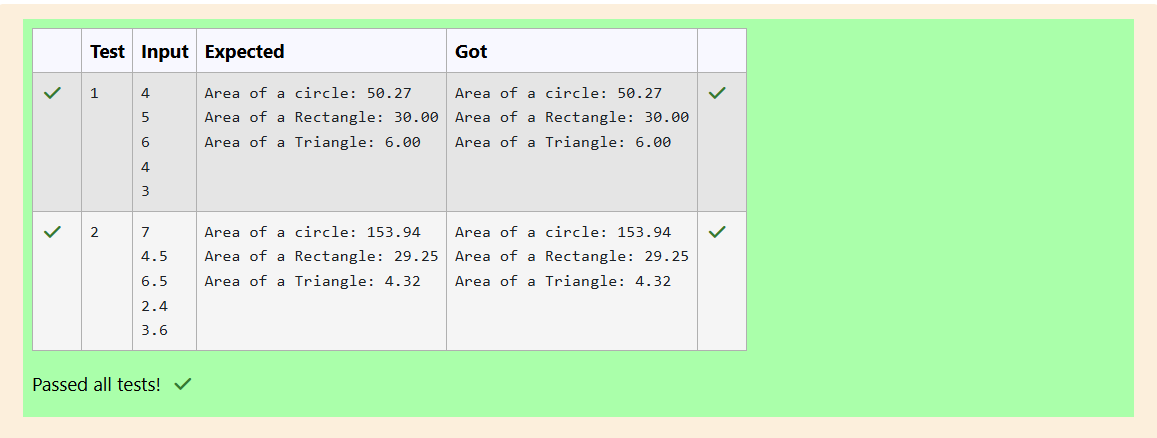
System.out.printf("Area of a Triangle: %.2f%n", triangle.calculateArea());

scanner.close();

}

}

Output:



Week8:Q3

### 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.");  
}

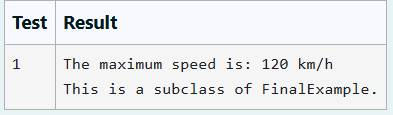
### 3. **Final Class:**

* 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:**



Program:

// 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.

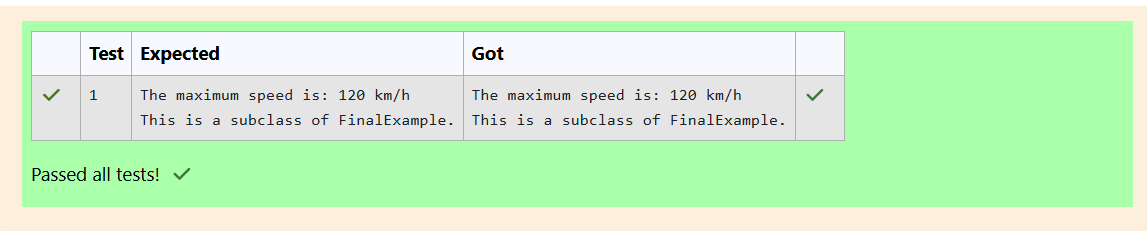
// class SubclassExample extends FinalExample { }

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

}

}

Output:



Week9:Q1

Write a Java program to create a method that takes an integer as a parameter

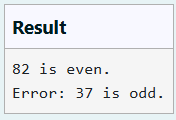
and throws an exception if the number is odd.

**Sample input and Output:**

82 is even.  
Error: 37 is odd.

Fill the preloaded answer to get the expected output.

**For example:**



Program:

class prog {

public static void main(String[] args) {

int n = 82;

trynumber(n);

n = 37;

trynumber(n);

}

public static void trynumber(int n) {

try {

checkEvenNumber(n);

System.out.println(n + " is even.");

} catch (Exception e) {

System.out.println("Error: "+ n + " is odd.");

}

}

public static void checkEvenNumber(int number) throws Exception {

if (number % 2 != 0) {

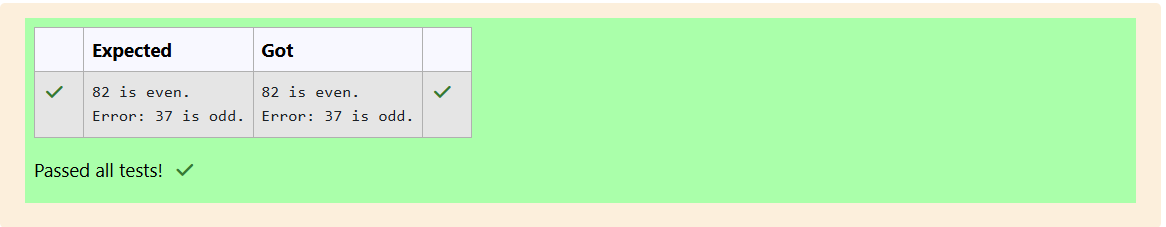
throw new Exception(number + " is odd.");

}

}

}

Output:



Week9:Q2

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:**

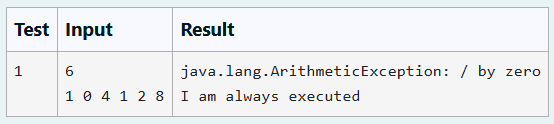
**java.lang.ArithmeticException: / by zero**  
**I am always executed**

Input:

3  
10 20 30

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

**For example:**



Program:

import java.util.Scanner;

public class ExceptionHandling {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

//System.out.print("Enter the size of the array: ");

int n = sc.nextInt();

// Create an array of size n

int[] arr = new int[n];

// Read elements into the array

//System.out.println("Enter " + n + " elements:");

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

arr[i] = sc.nextInt();

}

try {

// Divide the 0th element by the 1st element and store the result

int result = arr[0] / arr[1];

System.out.printf("java.lang.ArrayIndexOutOfBoundsException: Index %d out of bounds for length %d\n",n,n);

} catch (ArithmeticException e) {

System.out.println("java.lang.ArithmeticException: " + e.getMessage());

} catch (ArrayIndexOutOfBoundsException e) {

System.out.printf("java.lang.ArrayIndexOutOfBoundsException: Index %d out of bounds for length %d",n,n);

} finally {

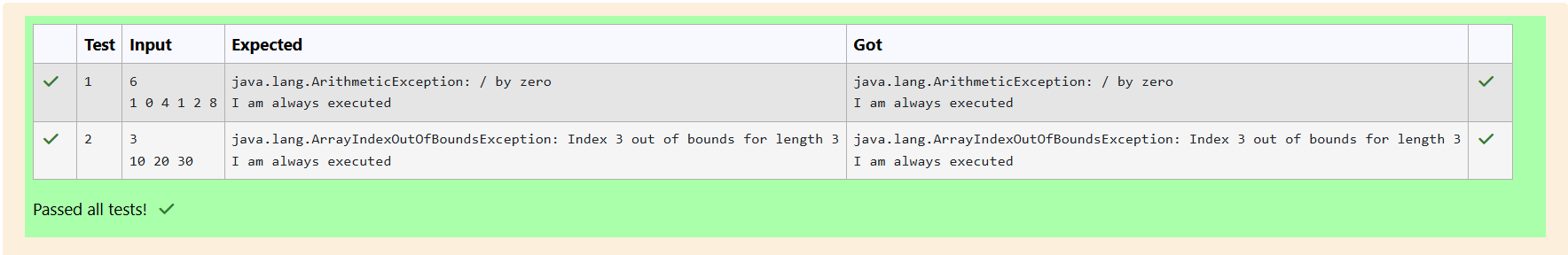
System.out.println("I am always executed");

}

}

}

Output:



Week9:Q3

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:**

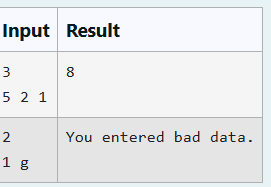
2

1 g

**Sample Output:**

You entered bad data.

**For example:**



Program:

import java.util.Scanner;

import java.util.InputMismatchException;

public class ArrayInitialization {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

try {

//System.out.print("Enter the size of the array: ");

int

length = sc.nextInt();

int[] name = new int[length];

//System.out.println("Enter " + length + " integers:");

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

name[i] = sc.nextInt();

}

int sum = 0;

for (int num : name) {

sum += num;

}

System.out.println(sum);

} catch (InputMismatchException e) {

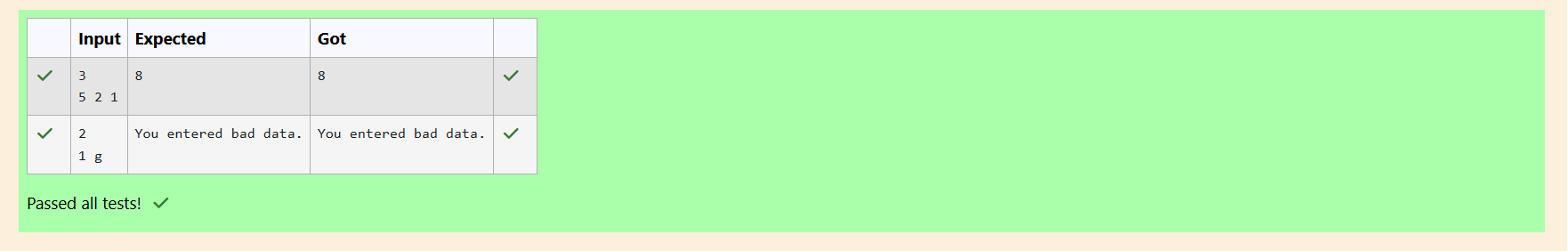
System.out.println("You entered bad data.");

}

}

}

Output:



Week10:Q1

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 FirstLastElement{

public static void main(String[] args){

Scanner scanner = new Scanner(System.in);

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

int size = scanner.nextInt();

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

int element = scanner.nextInt();

numbers.add(element);

}

int firstElement = numbers.get(0);

int lastElement = numbers.get(numbers.size() - 1);

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

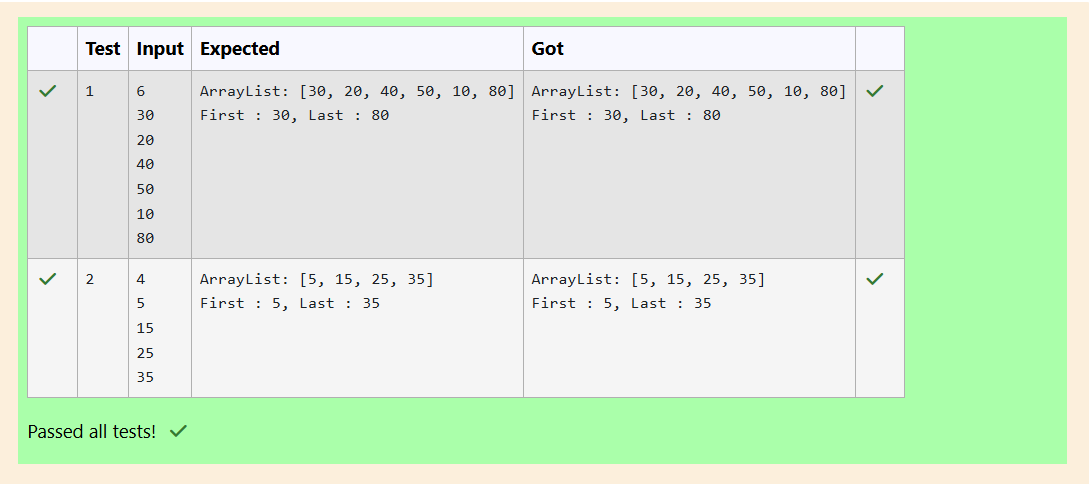
System.out.println("First : " + firstElement + ", Last : " + lastElement);

scanner.close();

}

}

Output:



Week10:Q2

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.

Program:

import java.util.ArrayList;

import java.util.Scanner;

public class Prog { // Changed class name to 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);

// Removing an element from position 3

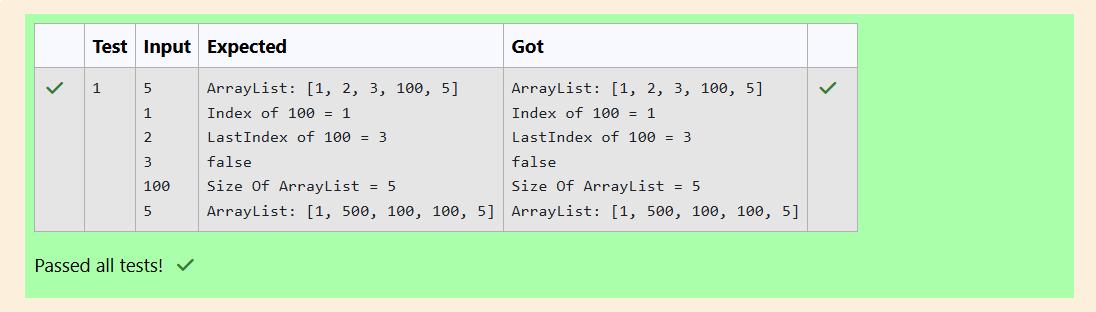
list.remove(3);

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

}

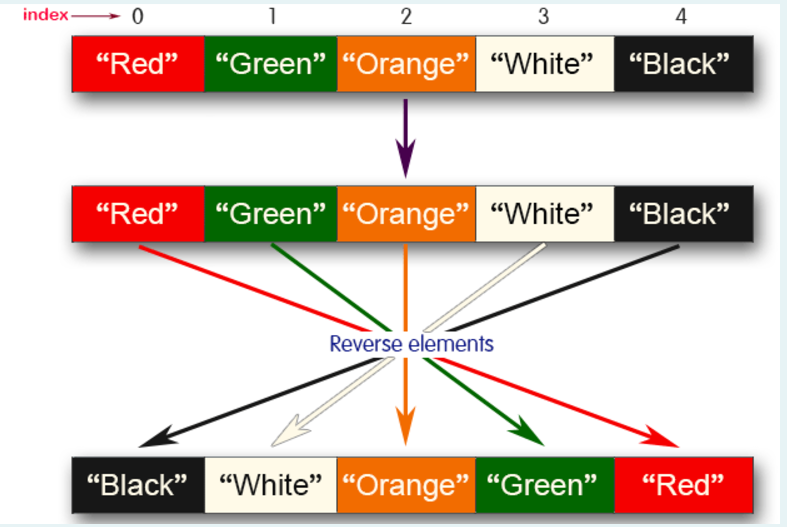
}

Output:



Week10:Q3

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]

Program:

import java.util.ArrayList;

import java.util.Scanner;

public class ReverseArrayList {

public static void main(String[] args) {

Scanner scanner= new Scanner(System.in);

int n = scanner.nextInt();

scanner.nextLine();

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

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

String element = scanner.nextLine();

list.add(element);

}

System.out.println("List before reversing :"+"\n"+"[" + String.join(", ", list) + "]");

for (int i = 0, j = list.size() - 1; i < j; i++, j--) {

String temp = list.get(i);

list.set(i, list.get(j));

list.set(j, temp);

}

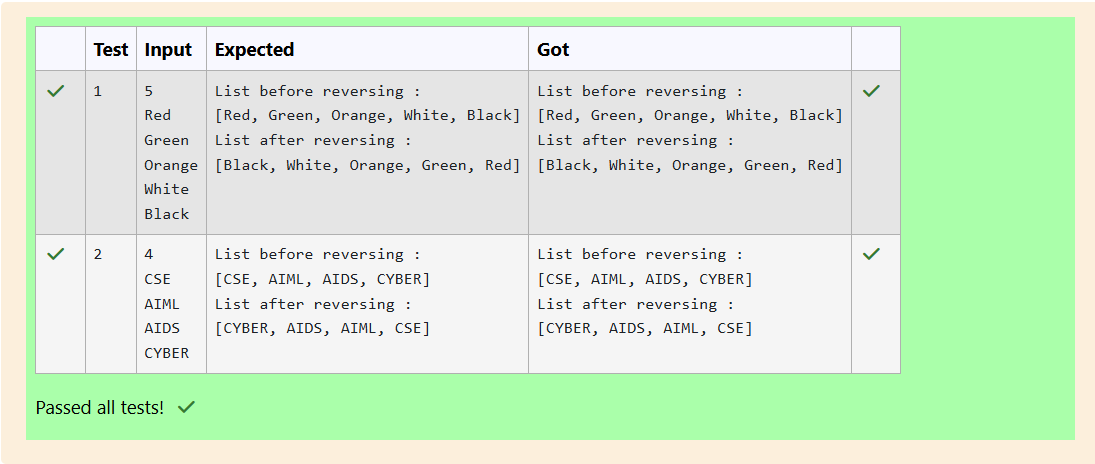
System.out.println("List after reversing :"+"\n"+"[" + String.join(", ", list) + "]");

scanner.close();

}

}

Output:



Week11:Q1

**Java HashSet** class implements the Set interface, backed by a hash table which is actually a [HashMap](https://www.geeksforgeeks.org/java-util-hashmap-in-java/) 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](https://www.geeksforgeeks.org/set-in-java/).
* The underlying data structure for HashSet is [Hashtable](https://www.geeksforgeeks.org/hashtable-in-java/).
* 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.

Program:

import java.util.HashSet;

import java.util.Scanner;

class prog {

public static void main(String[] args) {

Scanner sc= new Scanner(System.in);

int n = sc.nextInt();

HashSet<Integer> numbers=new HashSet<>();

// Add values to the set

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

numbers.add(sc.nextInt());

int skey=sc.nextInt();

if(numbers.contains(skey))

{

System.out.println(skey + " was found in the set.");

} else {

System.out.println(skey + " was not found in the set.");

}

}

}

Output:



Week11:Q2

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

Program:

import java.util.Scanner;

import java.util.HashSet;

public class Main{

public static void main(String[] args) {

HashSet<String> set1 = new HashSet<>();

HashSet<String> set2 = new HashSet<>();

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

sc.nextLine();

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

String y = sc.nextLine();

set1.add(y);

}

int t = sc.nextInt();

sc.nextLine();

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

String y = sc.nextLine();

set2.add(y);

}

set1.retainAll(set2);

for (String element : set1) {

System.out.println(element);

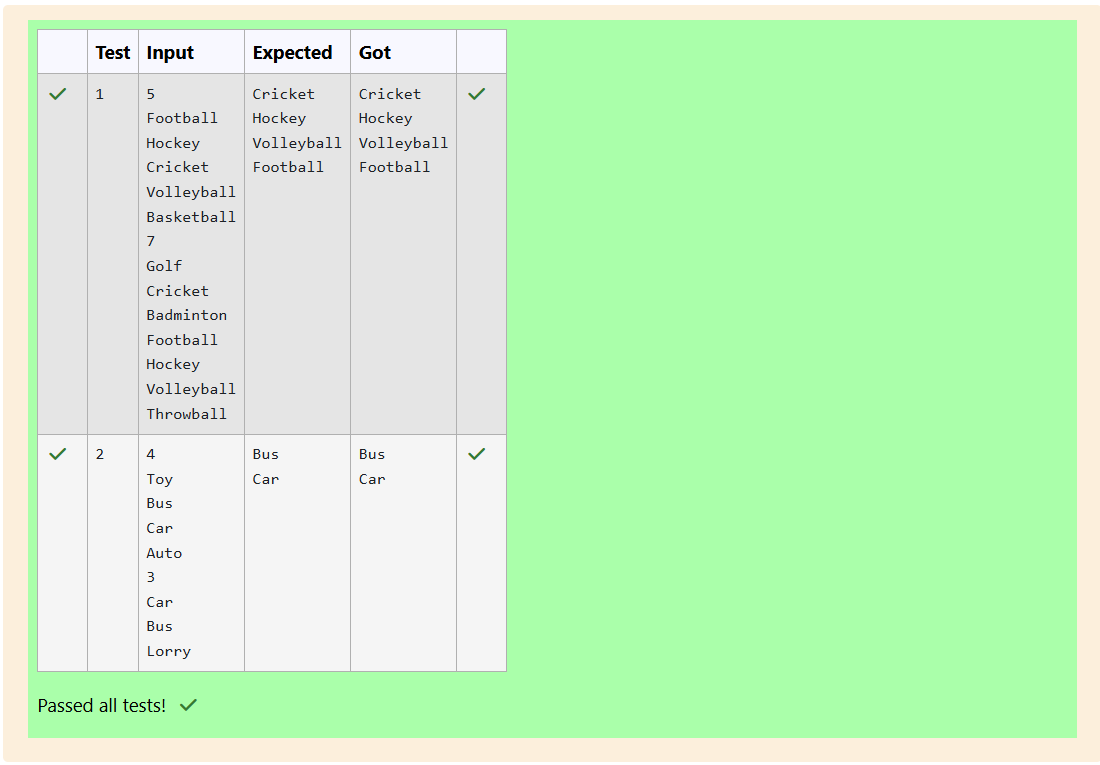
}

sc.close();

}

}

Output:



Week11:Q3

Java HashMap Methods

[containsKey()](https://www.w3schools.com/java/ref_hashmap_containskey.asp) Indicate if an entry with the specified key exists in the map

[containsValue()](https://www.w3schools.com/java/ref_hashmap_containsvalue.asp) Indicate if an entry with the specified value exists in the map

[putIfAbsent()](https://www.w3schools.com/java/ref_hashmap_putifabsent.asp) Write an entry into the map but only if an entry with the same key does not already exist

[remove()](https://www.w3schools.com/java/ref_hashmap_remove.asp) Remove an entry from the map

[replace() Write to an entry in the map only if it exists](https://www.w3schools.com/java/ref_hashmap_replace.asp)

[size()](https://www.w3schools.com/java/ref_hashmap_size.asp) 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();

// Reading and putting entries in the map

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); // Filling 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'

Integer value = map.get("TWO");

System.out.println(value != null ? value : "Key 'TWO' not found");

// Checking whether key 'ONE' exists in map

System.out.println(map.containsKey("ONE"));

// Checking whether value '3' exists in map

System.out.println(map.containsValue(3));

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

System.out.println(map.size());

// Closing the scanner

sc.close();

}

}

Output:



Week12:Q1

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.

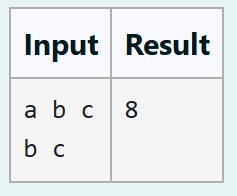
ASCII value of ‘b’ is 98 and ‘c’ is 99.

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

**For example:**



Program:

public class CommonAlphabetsSum {

public static int calculateSingleDigitSum(char[] input1, char[] input2) {

StringBuilder commonChars = new StringBuilder();

for (char c1 : input1) {

for (char c2 : input2) {

if (c1 == c2 && commonChars.indexOf(String.valueOf(c1)) == -1) {

commonChars.append(c1);

}

}

}

int sum1 = 0;

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

sum1 += (int) commonChars.charAt(i);

}

while (sum1 > 9) {

int tempSum = 0;

while (sum1 > 0) {

tempSum += sum1 % 10;

sum1 /= 10;

}

sum1 = tempSum;

}

return sum1;

}

public static void main(String[] args) {

char[] input1 = {'a', 'b', 'c'};

char[] input2 = {'b', 'c'};

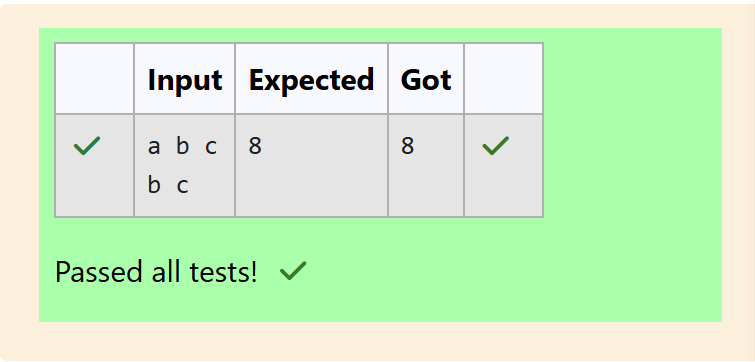
int result = calculateSingleDigitSum(input1, input2);

System.out.println(result);

}

}

Output:



Week12:Q2

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

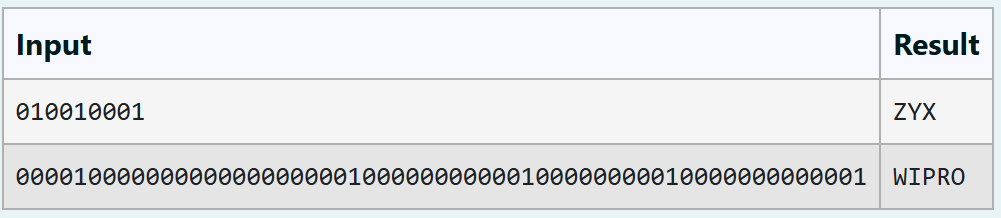
Example 2:

input1: 00001000000000000000000100000000000100000000010000000000001

The decoded string (original word) will be: WIPRO

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

**For example:**



Program:

import java.util.Scanner;

public class DecodeBinaryString {

public static String decode(String input) {

String[] groups = input.split("1");

StringBuilder decodedWord = new StringBuilder();

for (String group : groups) {

int zeroCount = group.length();

char decodedChar = (char) ('Z' - zeroCount + 1);

decodedWord.append(decodedChar);

}

return decodedWord.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String input = scanner.nextLine();

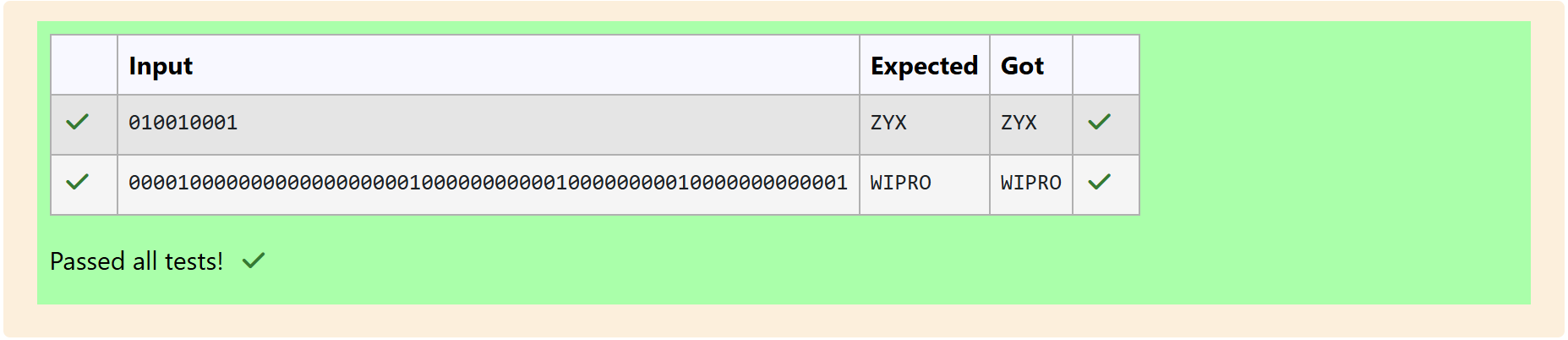
String decodedString = decode(input);

System.out.println(decodedString);

}

}

Output:



Week12:Q3

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.

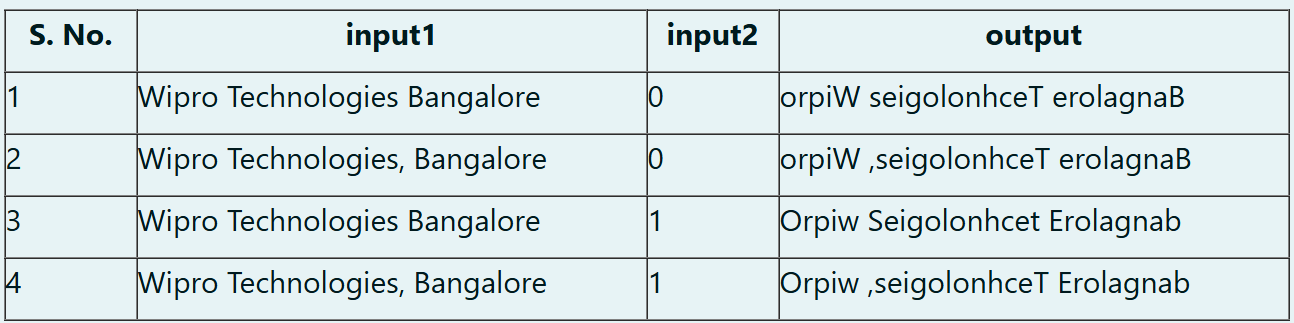
NOTE:

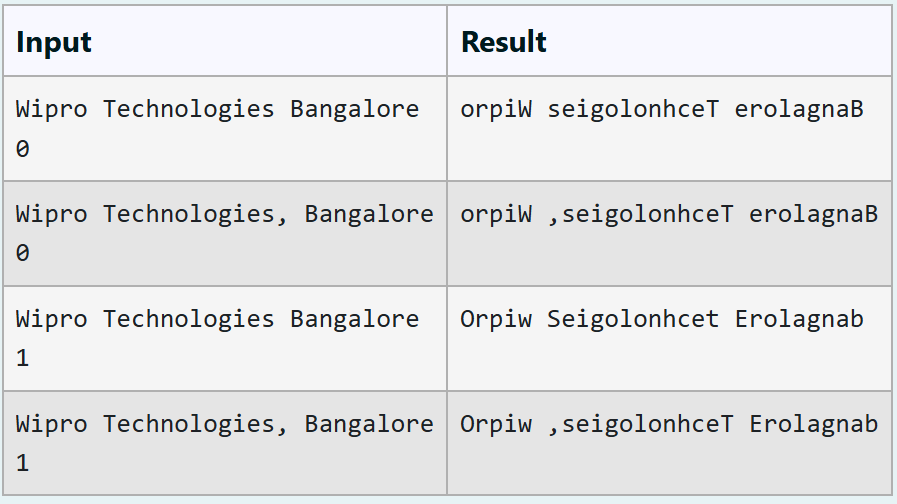
1. Only space character should be treated as the word separator i.e., “Hello World” should be treated as two separate words, “Hello” and “World”. However, “Hello,World”, “Hello;World”, “Hello-World” or “Hello/World” should be considered as a single word.

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:

**For example:**



Program:

import java.util.Scanner;

public class StringReversal {

public static String reverseWords(String sentence, int caseOption) {

String[] words = sentence.split("[ ,]");

StringBuilder reversedSentence = new StringBuilder();

for (String word : words) {

String reversedWord = reverseWord(word);

if (caseOption == 0) {

reversedSentence.append(reversedWord).append(" ");

} else if (caseOption == 1) {

reversedSentence.append(reversedWordWithCase(word, reversedWord)).append(" ");

}

}

return reversedSentence.toString().trim();

}

private static String reverseWord(String word) {

StringBuilder reversed = new StringBuilder(word);

return reversed.reverse().toString();

}

private static String reversedWordWithCase(String original, String reversed) {

char[] originalChars = original.toCharArray();

char[] reversedChars = reversed.toCharArray();

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

if (Character.isUpperCase(originalChars[i])) {

reversedChars[i] = Character.toUpperCase(reversedChars[i]);

} else if (Character.isLowerCase(originalChars[i])) {

reversedChars[i] = Character.toLowerCase(reversedChars[i]);

}

}

return new String(reversedChars);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String inputSentence = scanner.nextLine();

int caseOption = 1;

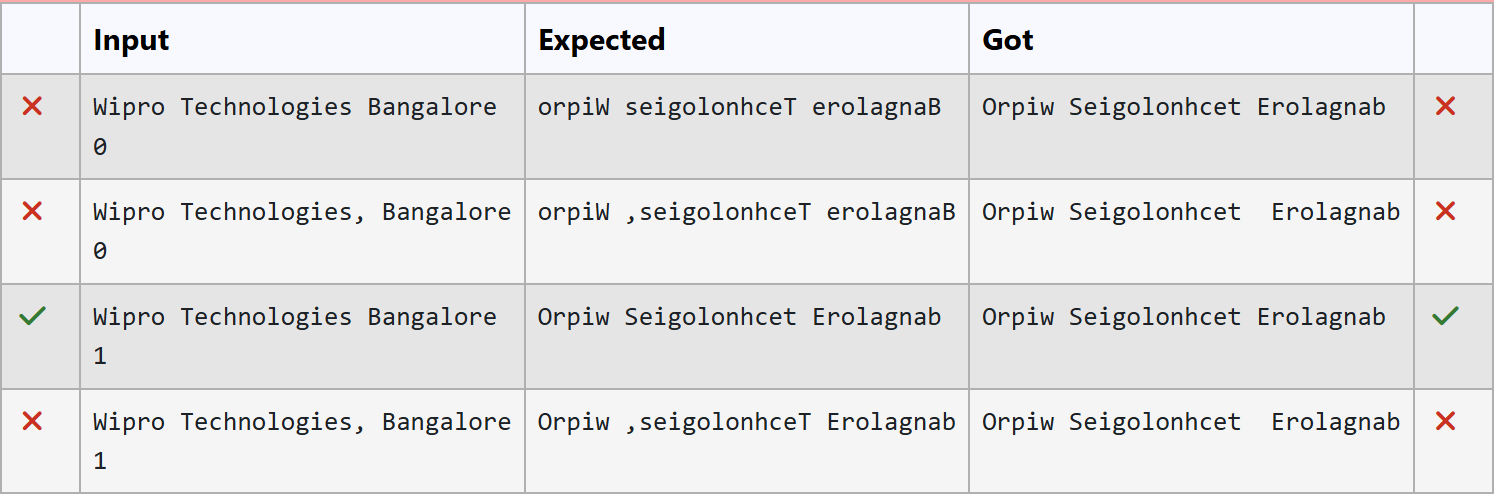
String reversedSentence = reverseWords(inputSentence,caseOption);

System.out.println(reversedSentence);

}

}

Output:



**JAVA MINI PROJECT-TIMETABLE MANAGEMENT SYSTEM**