Name : Chandrakanth HV
Assignment : Bridge Course Day 7
Date : 02-07-2025

SECTION 1 :
Problem Statement 1 :Write a program that works with an array of integer temperatures.

1. Create an integer array of temperatures.
2. Print all the temperature values.
3. Calculate and display the sum, average, and highest temperature from the array.

Pseudo code :

- START
- Create a Scanner to read user input.
- Prompt user to enter number of temperatures (n).
- Create an integer array of size n.
- Prompt user to enter n temperature values and store them in array.
- Print all temperatures.
- Initialize sum = 0 and highest = first temperature in array.
- Loop through the array:
- Add each temperature to sum.
- If temperature > highest, update highest.
- Calculate average = sum / n.
- Display sum, average, and highest temperature.
- Close Scanner.
- END

Algorithm: steps

Step 1: Start

Step 2: Read integer n from user (number of temperatures)

Step 3: Create an array 'temperatures' of size n

Step 4: For i = 0 to n-1:

Read temperature and store in temperatures[i]

Step 5: Print all temperature values from the array

Step 6: Initialize sum = 0, highest = temperatures[0]

Step 7: For each temperature in the array:

Add to sum

If temperature > highest, then update highest

Step 8: Calculate average = sum / n

Step 9: Print sum, average, and highest temperature

Step 10: End

Code :

```
import java.util.Scanner;

public class TemperatureAnalyzer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of temperatures: ");
        int n = scanner.nextInt();

        int[] temperatures = new int[n];

        System.out.println("Enter the temperatures:");
        for (int i = 0; i < n; i++) {
            temperatures[i] = scanner.nextInt();
        }

        System.out.print("Temperatures: ");
        for (int temp : temperatures) {
            System.out.print(temp + " ");
        }
        System.out.println();

        int sum = 0;
        int highest = temperatures[0];
        for (int temp : temperatures) {
            sum += temp;
            if (temp > highest) {
```

```
                highest = temp;
            }
        }
        double average = (double) sum / n;

        System.out.println("Sum: " + sum);
        System.out.println("Average: " + average);
        System.out.println("Highest Temperature: " + highest);

        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter number of temperatures: 5 Enter the temperatures: 20 20 25 25 20	Sum: 110 Average: 22.0 Highest Temperature: 25	Sum: 110 Average: 22.0 Highest Temperature: 25	Pass
TC2	Enter number of temperatures: 3 Enter the temperatures: -20 20 -20	Temperatures: -20 20 -20 Sum: -20 Average: -6.666666666666667 Highest Temperature: 20	Temperatures: -20 20 -20 Sum: -20 Average: -6.666666666666667 Highest Temperature: 20	Pass
TC3	Enter number of temperatures: 3	Temperatures: 10 10 10	Temperatures: 10 10 10 Sum: 30	Pass

	Enter the temperatures: 10 10 10 3	Sum: 30 Average: 10.0 Highest Temperature: 10	Average: 10.0 Highest Temperature: 10	

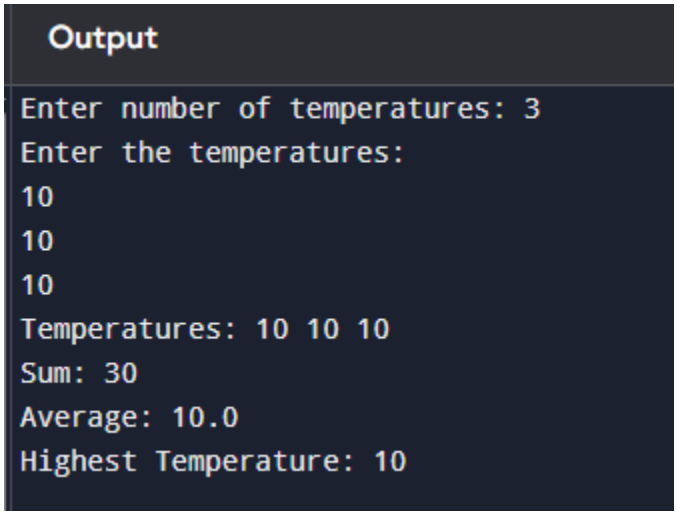
TC1 :

```
Output
Enter number of temperatures: 5
Enter the temperatures:
20
20
25
25
20
Temperatures: 20 20 25 25 20
Sum: 110
Average: 22.0
Highest Temperature: 25
```

TC2 :

```
Output
Enter number of temperatures: 3
Enter the temperatures:
-20
20
-20
Temperatures: -20 20 -20
Sum: -20
Average: -6.666666666666667
Highest Temperature: 20
```

TC3 :



Observation:

The program reads n temperature values from the user.It calculates:The sum of all temperatures.The average of temperatures.The highest temperature value.It uses a single loop to compute both sum and highest efficiently.The array allows storage and processing of dynamic user input size.The program ensures good structure and simple logic, useful for beginner-level data processing.

Problem Statement 2 : Write a Java program to perform the following tasks:Create an integer array containing numbers from 1 to 10.Find and calculate the product of all even numbers present in the array.Display the final product.

Pseudo code :

- START
- . Input number of elements (n)
- . Declare an integer array of size n
- . FOR i = 0 to n-1:
- Read number and store in array[i]
- . Initialize product = 1, hasEven = false
- . FOR each number in array:
- IF number is even THEN
- Multiply product with number
- Set hasEven to true
- . IF hasEven is true THEN
- Print product
- ELSE
- Print "No even numbers found"
- END
-

Algorithm: steps

Step 1: Start

Step 2: Ask the user for the number of elements (n)

Step 3: Create an integer array of size n

Step 4: Read n elements from the user into the array

Step 5: Initialize product = 1 and hasEven = false

Step 6: For each number in the array:

If the number is even:

Multiply it with product

Set hasEven = true

Step 7: If hasEven is true:

Print the product of even numbers

Else:

Print "No even numbers found"

Step 8: End

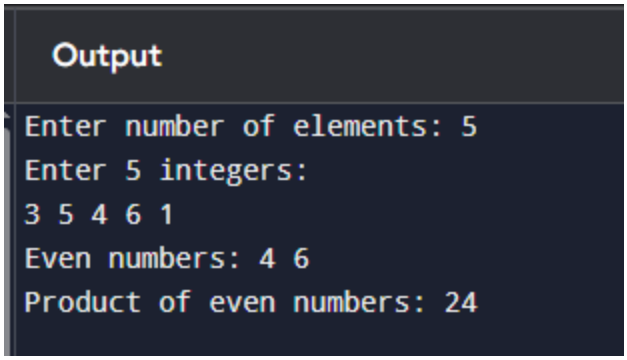
Code :

```
import java.util.Scanner;
class Animal {
    public void makeSound()  {
    }
}


class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Dog says: Bark!");
    }
}


class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Cat says: Meow!");
    }
}


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

        System.out.println("Choose an animal:\n1. Dog\n2. Cat");
        int choice = Integer.parseInt(scanner.nextLine());

        Animal animal;
```

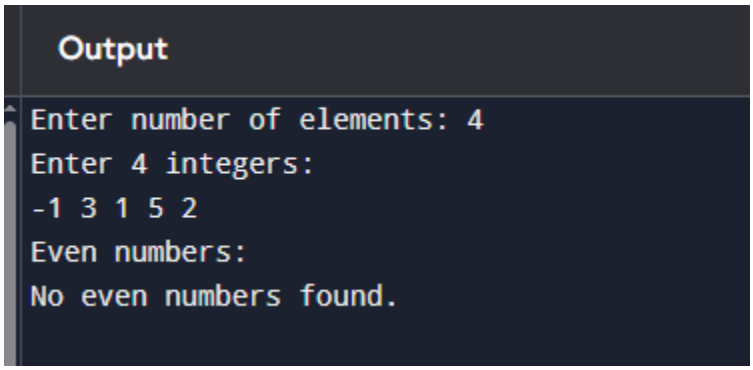
```
        if (choice == 1) {
            animal = new Dog();
        } else if (choice == 2) {
            animal = new Cat();
        } else {
            animal = new Animal();
            System.out.println("Invalid choice.");
        }

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

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter number of elements: 5 Enter 5 integers: 3 5 4 6 1	Even numbers: 4 6 Product of even numbers: 24	Even numbers: 4 6 Product of even numbers: 24	Pass
TC2	Enter number of elements: 4 Enter 4 integers: -1 3 1 5 2	Even numbers: No even numbers found.	Even numbers: No even numbers found.	Pass
TC3	Enter number of elements: 0	Enter 0 integers: Even numbers: No even numbers found.	Enter 0 integers: Even numbers: No even numbers found.	Pass

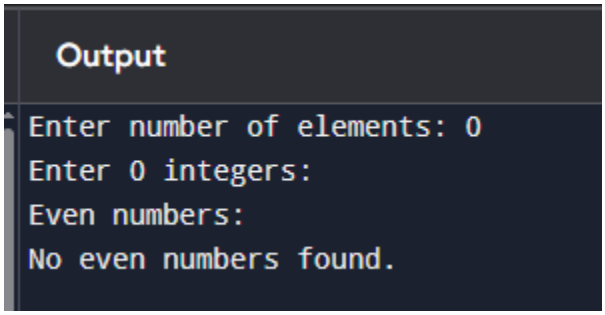
TC1 :



TC2 :



TC3 :



Observation:

The program uses a loop to read user input and find even numbers.It multiplies only the even numbers from the array.A flag (`hasEven`) is used to check if at least one even number was found.If no even number is found, it avoids printing incorrect results by giving a message.

.

Problem Statement 3 : Write a Java program to perform the following tasks:Create an array of strings with a set of items ,Print the elements of the array in reverse order.

Pseudo code :

```
START
1. Input number of items (n)
2. Declare a string array of size n
3. FOR i = 0 to n-1:
    Read item and store in array[i]
4. Print "Original List:"
5. FOR i = 0 to n-1:
    Print array[i]
6. Print "Reversed List:"
7. FOR i = n-1 to 0:
    Print array[i]
END
```

Algorithm: steps

Step 1: Start

Step 2: Ask the user to enter the number of items (n)

Step 3: Create a string array of size n

Step 4: Read n strings from the user and store in the array

Step 5: Print all items in original order

Step 6: Print all items in reverse order using a loop from n-1 to 0

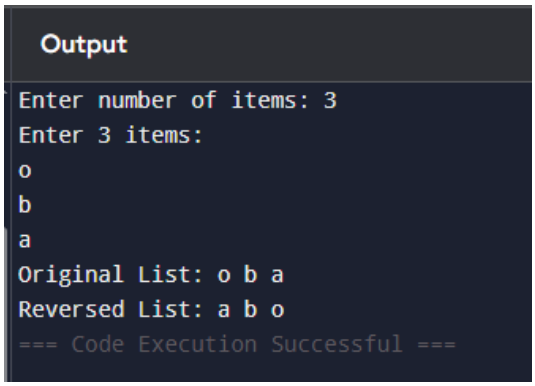
Step 7: End

Code :

```
import java.util.Scanner;

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

        System.out.print("Enter number of items: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        String[] items = new String[n];

        System.out.println("Enter " + n + " items:");
        for (int i = 0; i < n; i++) {
            items[i] = scanner.nextLine();
        }

        System.out.print("Original List: ");
        for (String item : items) {
            System.out.print(item + ", ");
        }

        System.out.print("\nReversed List: ");
        for (int i = n - 1; i >= 0; i--) {
            System.out.print(items[i] + ", ");
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter number of items: 3 Enter 3 items: o b a	Original List: o b a Reversed List: a b o	Original List: o b a Reversed List: a b o	Pass
TC2	Enter number of items: 2 Enter 2 items: or ap	Original List: or, ap, Reversed List: ap ,or	Original List: or, ap, Reversed List: ap ,or	Pass
TC3	Enter number of items: 2 Enter 2 items: 25 36	Original List: 25, 36, Reversed List: 36, 25,	Original List: 25, 36, Reversed List: 36, 25,	Pass

TC1 :



TC2 :

```
Output
Enter number of items: 2
Enter 2 items:
or
ap
Original List: or, ap,
Reversed List: ap ,or ,
```

TC2 :

```
Output
Enter number of items: 2
Enter 2 items:
25
36
Original List: 25, 36,
Reversed List: 36, 25,
=== Code Execution Successful ===
```

Observation:

The program reads a list of strings from the user.
It stores the values in an array and displays them in original and reverse order.
Useful for learning how to:Work with arrays,Take string input from users,Use loops for forward and backward traversal

Problem Statement 4 : Write a Java program to perform the following tasks:Accept a list of elements from the user and store them in an array.Ask the user to enter a search term.Check whether the search term exists in the array.

Pseudo code :

START

1. Input number of items (n)

2. Declare a string array of size n

3. FOR i = 0 to n-1:

    Read item and store in array[i]

4. Input searchItem

5. Initialize found = false

6. FOR i = 0 to n-1:

    IF array[i] equals searchItem (ignore case)

        Print item found and position

        Set found = true

        BREAK

7. IF found is false

    Print "Item not found"

END

Algorithm: steps

1. Step 1: Start
2. Step 2: Ask the user to enter the number of items (n)
3. Step 3: Create a string array of size n
4. Step 4: Read n items from the user into the array
5. Step 5: Ask the user to enter the item to search

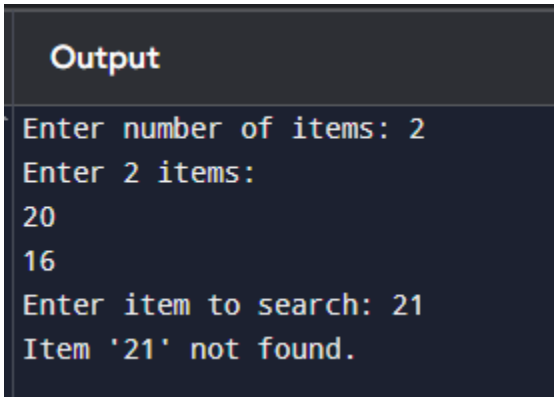
6. Step 6: Search the item in the array using a loop
7. Step 7: If found, print the item and its index
8. Step 8: If not found, print "Item not found"
9. Step 9: End

Code :

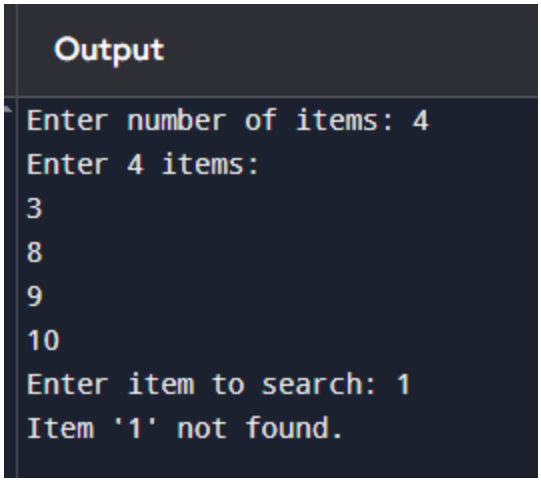
```
import java.util.Scanner;

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

        System.out.print("Enter number of items: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        String[] items = new String[n];

        System.out.println("Enter " + n + " items:");
        for (int i = 0; i < n; i++) {
            items[i] = scanner.nextLine();
        }

        System.out.print("Enter item to search: ");
        String searchItem = scanner.nextLine();

        boolean found = false;
        for (int i = 0; i < n; i++) {
            if (items[i].equalsIgnoreCase(searchItem)) {
                System.out.println("Item '" + searchItem + "' found at position " + i);
                found = true;
                break;
            }
        }

        if (!found) {
            System.out.println("Item '" + searchItem + "' not found.");
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter number of items: 2 Enter 2 items: 20 16	Enter item to search: 21 Item '21' not found.	Enter item to search: 21 Item '21' not found.	Pass
TC2	Enter number of items: 4 Enter 4 items: 3 8 9 10	Enter item to search: 1 Item '1' not found.	Enter item to search: 1 Item '1' not found.	Pass
TC3	Enter number of items: 4 Enter 4 items: 2 3 4 5	Enter item to search: 3 Item '3' found at position 1	Enter item to search: 3 Item '3' found at position 1	Pass

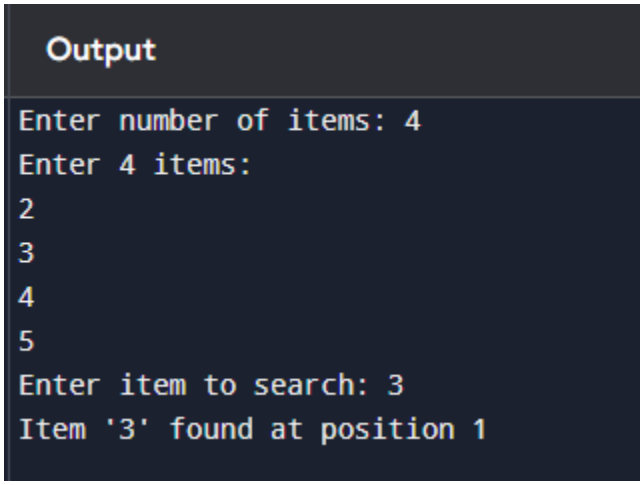
TC1 :



TC2 :



TC3 :



Observation:

The program reads a list of strings from the user and stores them in an array.It then searches for a given string in the array.The comparison is case-insensitive using `.equalsIgnoreCase()`.If the item is found, it prints the position .If not found, it displays a suitable message.Useful for learning:Arrays,Looping through arrays,String comparison in Java,Conditional logic

Section 2 :

Problem Statement 1 : Write a Java program to implement a function that calculates the Greatest Common Divisor (GCD) of two integers using the Euclidean Algorithm.:The function should take two integers as input and return their GCD.Test the function with multiple pairs of integers by taking user input.

Pseudo code :

- Start
- Show instrument options to user
- Read user choice
- If choice is 1 → create guitar
- else if choice is 2 → create piano
- else → print "invalid choice" and exit
- Call play() on selected instrument
- End

Algorithm: steps

1. Start
2. Display menu (1. Guitar, 2. Piano)
3. Take user input → choice
4. If choice = 1 → create Guitar object
5. If choice = 2 → create Piano object
6. Else → print invalid and exit
7. Call play() method
8. End

Code :

```
import java.util.Scanner;

public class GCDCalculator {

    public static int computeGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

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

```
        System.out.print("Enter number of pairs to test: ");
        int pairs = scanner.nextInt();

        for (int i = 1; i <= pairs; i++) {
            System.out.print("\nEnter two numbers for pair " + i + ": ");
            int num1 = scanner.nextInt();
            int num2 = scanner.nextInt();

            int gcd = computeGCD(num1, num2);
            System.out.println("GCD of " + num1 + " and " + num2 + " is: " + gcd);
        }

        scanner.close();
    }
}
```

Test cases	**Input**	**Expected Output**	**Actual Output**	**Status**
TC1	Select an Instrument to Play: 1. Guitar 2. Piano 2	Playing the piano sound: Plink Plonk!	Playing the piano sound: Plink Plonk!	Pass
TC2	Select an Instrument to Play: 1. Guitar 2. Piano 1	Strumming the guitar sound: Twing Twing!	Strumming the guitar sound: Twing Twing!	Pass
TC3	Select an Instrument to Play: 1. Guitar 2. Piano piano	ERROR!	ERROR!	Pass

TC1 :

```
Select an Instrument to Play:
1. Guitar
2. Piano
2
Playing the piano  sound: Plink Plonk!
```

TC2 :

```
Select an Instrument to Play:
1. Guitar
2. Piano
1
Strumming the guitar  sound: Twing Twing!
```

TC3 :

```
Select an Instrument to Play:
1. Guitar
2. Piano
piano
ERROR!
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
    at Main.main(Main.java:28)
```

Observation: The program uses abstraction to handle different automated tasks with a common extence() method. It simplifies task execution and improves code structure by treating all tasks uniformly.

Problem Statement 3 : Write a Java program to compute the Least Common Multiple (LCM) of two integers.
Reuse the GCD function implemented in Problem 2.1.
Use the formula:
LCM(a,b)=|a×b|
GCD(a,b)
LCM(a,b)= GCD(a,b)
|a×b|
Allow the user to input multiple pairs of numbers and display the LCM for each pair.

Pseudo code :

Start

Read number of pairs

For each pair from 1 to number of pairs do

read num1, num2

while num2 ≠ 0

temp = num2

num2 = num1 % num2

num1 = temp

gcd = num1

lcm = (original num1 × original num2) / gcd

display lcm

End for

End

Algorithm: steps

1. Start
2. Input the number of pairs to test
3. Repeat for each pair:
   a. Input two numbers
   b. Compute GCD using Euclidean algorithm
   c. Use GCD to compute LCM = (num1 × num2) / GCD
   d. Display the LCM

4. End

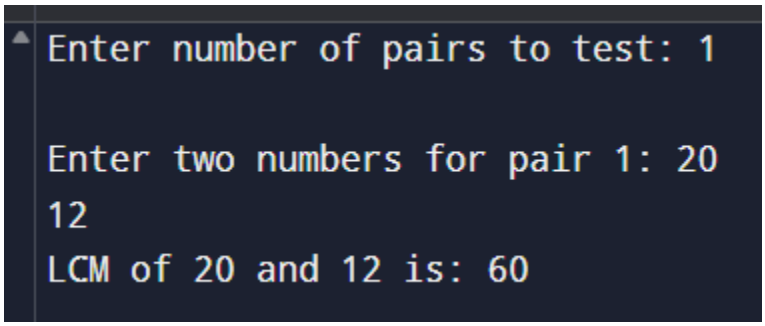
Code :

```
import java.util.Scanner;

public class LCMCalculator {

    public static int computeGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

    public static int computeLCM(int a, int b) {
        return Math.abs(a * b) / computeGCD(a, b);
    }

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

        System.out.print("Enter number of pairs to test: ");
        int pairs = scanner.nextInt();

        for (int i = 1; i <= pairs; i++) {
            System.out.print("\nEnter two numbers for pair " + i + ": ");
            int num1 = scanner.nextInt();
            int num2 = scanner.nextInt();

            int lcm = computeLCM(num1, num2);
            System.out.println("LCM of " + num1 + " and " + num2 + " is: " + lcm);
        }

        scanner.close();
    }
}
```

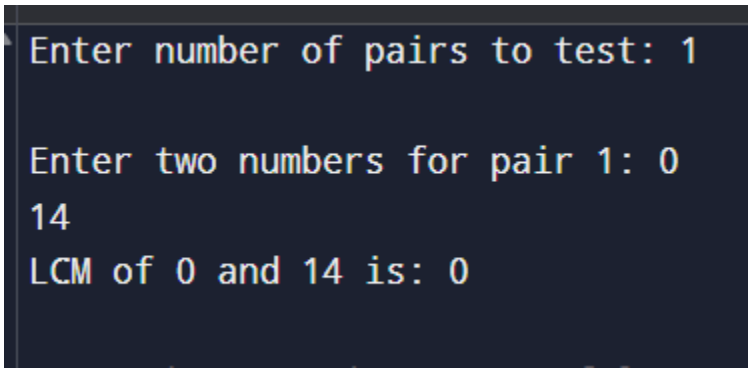
Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter number of pairs to test: 1	LCM of 20 and 12 is: 60	LCM of 20 and 12 is: 60	Pass

	Enter two numbers for pair 1: 20 12			
TC2	Enter number of pairs to test: 1 Enter two numbers for pair 1: 0 14	LCM of 0 and 14 is: 0	LCM of 0 and 14 is: 0	Pass
TC3	Enter number of pairs to test: 1 Enter two numbers for pair 1: -5 -7	LCM of -5 and -7 is: -35	LCM of -5 and -7 is: -35	Pass

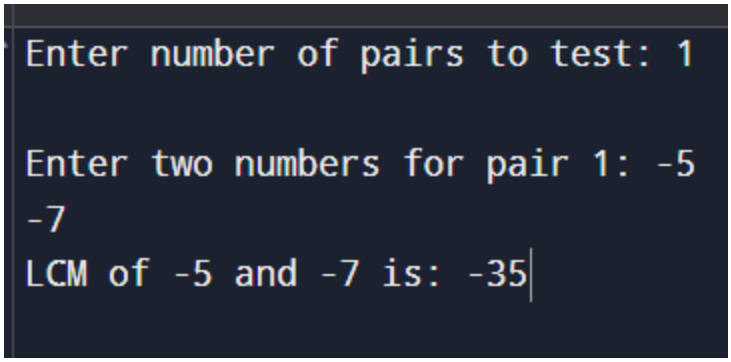
TC1:



TC2:



TC3:



Observation: The program efficiently calculates LCM for multiple pairs using the GCD function. It uses a loop for repeated input, applies the Euclidean algorithm for GCD, and ensures accurate results using the LCM formula. Input handling is user-friendly and scalable.

Problem Statement 3 : write a java program for calculating the LCM/GCD.

Pseudo code :

Start

Read num1, num2

Set a = num1, b = num2

While b ≠ 0

temp = b

b = a % b

a = temp

Gcd = a

lcm formula

Lcm = (num1 × num2) / gcd

Display gcd and lcm

End

Algorithm: steps

1. Start

2. Input two numbers: num1, num2
3. Assign a = num1, b = num2
4. While b is not zero:
    - Set temp = b
    - Set b = a % b
    - Set a = temp
5. GCD = a
6. LCM = (num1 × num2) / GCD
7. Display GCD and LCM
8. End

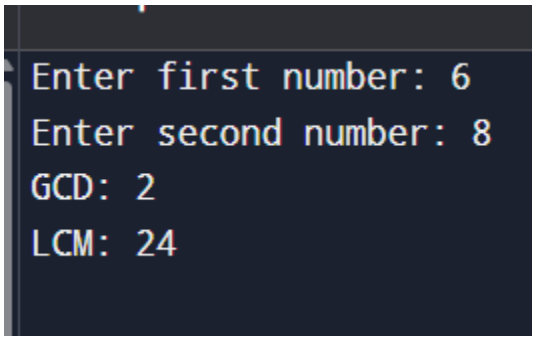
Code :

```
import java.util.Scanner;

public class GcdLcmCalculator {

    static int findGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

    static int findLCM(int a, int b) {
        return (a * b) / findGCD(a, b);
    }

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

        System.out.print("Enter first number: ");
        int num1 = scanner.nextInt();

        System.out.print("Enter second number: ");
        int num2 = scanner.nextInt();

        int gcd = findGCD(num1, num2);
        int lcm = findLCM(num1, num2);

        System.out.println("GCD: " + gcd);
        System.out.println("LCM: " + lcm);
    }
```

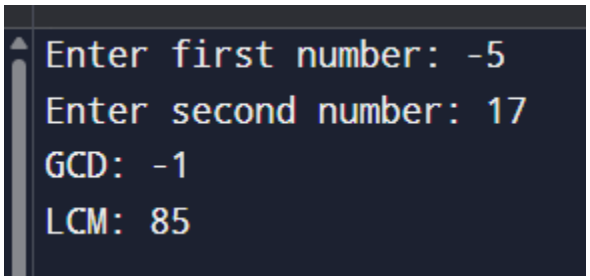
```
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter first number: 6 Enter second number: 8	GCD: 2 LCM: 24	GCD: 2 LCM: 24	Pass
TC2	Enter first number: -5 Enter second number: 17	GCD: -1 LCM: 85	GCD: -1 LCM: 85	Pass
TC3	Enter first number: -20 Enter second number: -15	GCD: -5 LCM: -60	GCD: -5 LCM: -60	Pass

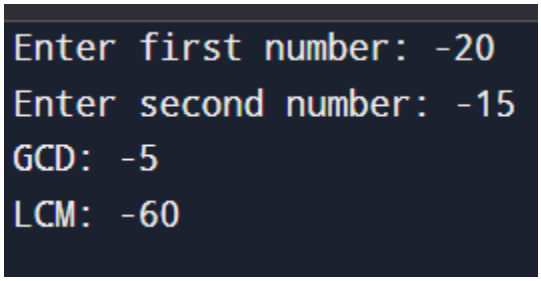
TC1 :



TC2 :



TC3 :



Observation: The program calculates GCD using the Euclidean algorithm and uses it to find LCM. It takes two numbers as input, works efficiently, and applies real-life math logic. The approach is simple, fast, and accurate.

SECTION 3 :(Frontend Basics)

Problem Statement 1 : (Simple Sum Calculator Web Page) create an HTML page for sum of two numbers and use javascript to compute and display the sum.

Code :

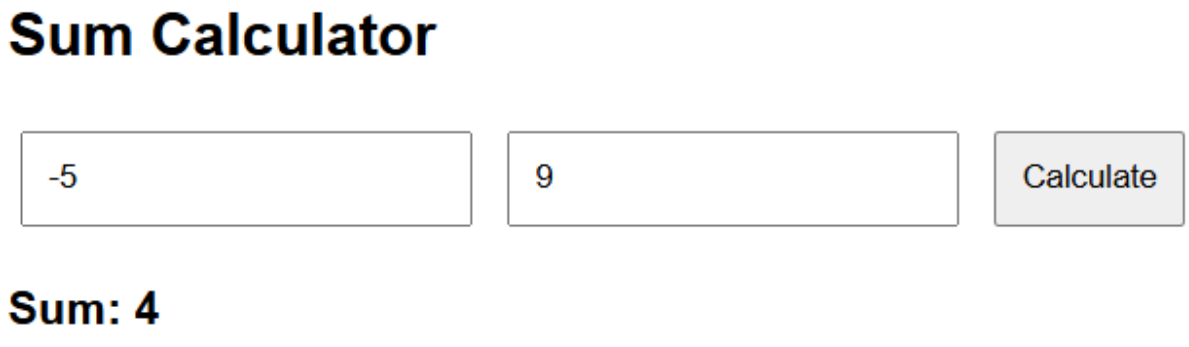
```
<!DOCTYPE html>
<html>
<head>
   <title>Simple Sum Calculator</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         padding: 30px;
      }
      input, button {
         margin: 5px;
         padding: 10px;
      }
   </style>
</head>
<body>

   <h2>Sum Calculator</h2>

   <form id="sumForm">
      <input type="number" id="num1" placeholder="Enter first number" required>
      <input type="number" id="num2" placeholder="Enter second number" required>
      <button type="button" onclick="calculateSum()">Calculate</button>
   </form>

   <h3 id="result"></h3>

   <script>
      function calculateSum() {

         const number1 = parseFloat(document.getElementById('num1').value);
         const number2 = parseFloat(document.getElementById('num2').value);

         if (isNaN(number1) || isNaN(number2)) {
            document.getElementById('result').innerText = "Please enter valid numbers.";
            return;
         }
```

```
        const sum = number1 + number2;

        document.getElementById('result').innerText = "Sum: " + sum;
    }
  </script>

</body>
</html>
```

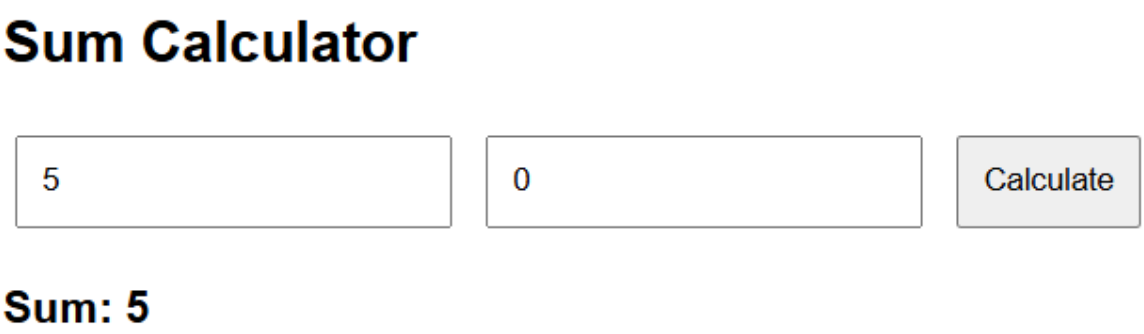
Test cases	Input	Expected Output	Actual Output	Status
TC1	12+4	16	16	Pass
TC2	-5+9	4	4	Pass
TC3	5+0	5	5	Pass

TC1 :

TC2 :



TC3 :



Observation: The Simple Sum Calculator uses HTML to create a basic web form with two number input fields and a calculate button. It provides a simple layout where users can enter two values. While the actual calculation is done using JavaScript, HTML is responsible for structuring the input fields and buttons that trigger the process.

Problem Statement 2 : (Web-based GCD/LCM Calculator) create the gcd/lcm calculator using HTML code and use javascript code for perform the calculations.

Code :

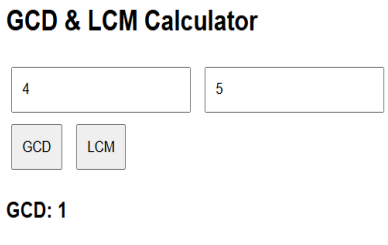
```
<!DOCTYPE html>
<html>
<head>
   <title>GCD & LCM Calculator</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         padding: 30px;
      }
      input, button {
         margin: 5px;
         padding: 10px;
      }
   </style>
</head>
<body>

   <h2>GCD & LCM Calculator</h2>

   <form id="calcForm">
      <input type="number" id="num1" placeholder="Enter first number" required>
```

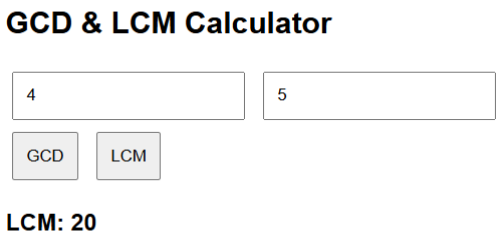
```
    <input type="number" id="num2" placeholder="Enter second number" required><br>
    <button type="button" onclick="calculateGCD()">GCD</button>
    <button type="button" onclick="calculateLCM()">LCM</button>
</form>

<h3 id="result"></h3>

<script>
    function getValues() {
        const a = parseInt(document.getElementById('num1').value);
        const b = parseInt(document.getElementById('num2').value);
        return [a, b];
    }

    function calculateGCD() {
        let [a, b] = getValues();
        if (isNaN(a) || isNaN(b)) {
            document.getElementById('result').innerText = "Please enter valid numbers.";
            return;
        }
        while (b !== 0) {
            let temp = b;
            b = a % b;
            a = temp;
        }
        document.getElementById('result').innerText = "GCD: " + a;
    }

    function calculateLCM() {
        let [a, b] = getValues();
        if (isNaN(a) || isNaN(b)) {
            document.getElementById('result').innerText = "Please enter valid numbers.";
            return;
        }
        let gcd = a;
        let temp = b;
        while (temp !== 0) {
            let t = temp;
            temp = gcd % temp;
            gcd = t;
        }
        let lcm = Math.abs(a * b) / gcd;
        document.getElementById('result').innerText = "LCM: " + lcm;
    }
```

```
</script>

</body>
</html>
```

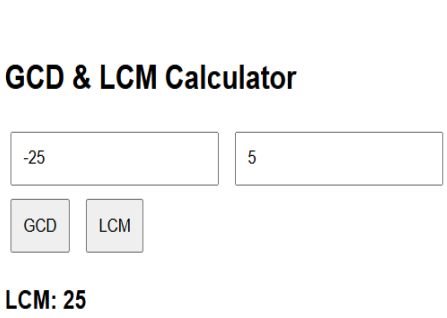
Test cases	Input	Expected Output	Actual Output	Status
TC1	4,5	LCM=20 GCD=1	LCM=20 GCD=1	Pass
TC2	-25,5	LCM=25 GCD=5	LCM= GCD=	Pass
TC3	-20,-3	LCM=-60 GCD=-1	LCM=-60 GCD=-1	Pass

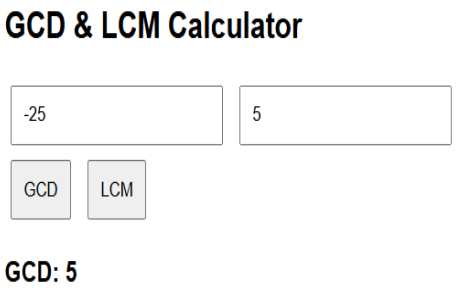
TC1 :



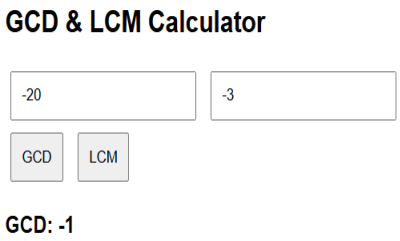


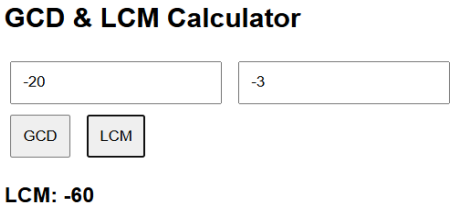
TC2:





TC3:





Observation: The GCD/LCM Calculator uses HTML to create input fields and buttons, allowing users to enter two numbers and choose either GCD or LCM. JavaScript handles the calculations and displays results instantly on the page. It's simple, interactive, and works fully in the browser.

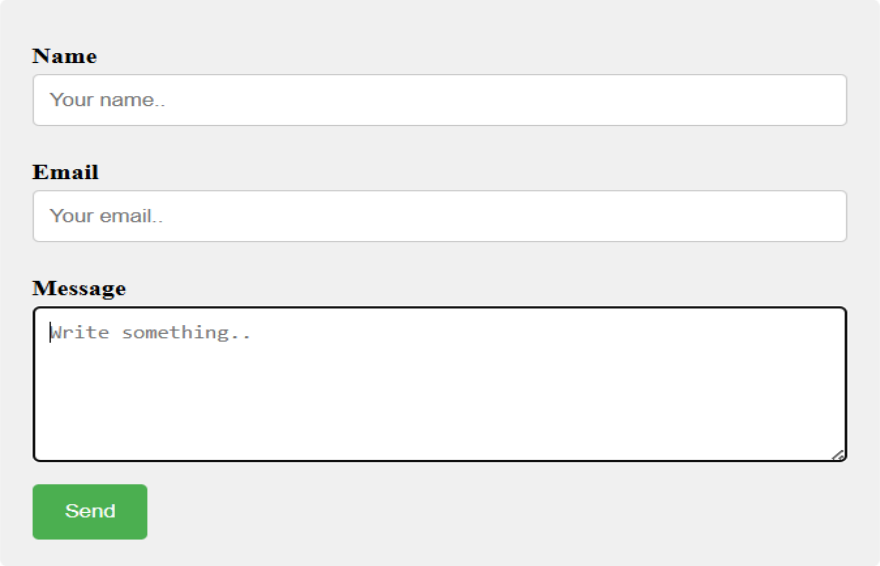
Problem Statement 3 : (Inspect & Replicate) create a login page using HTML and inspect the HTML/CSS.

Code :

```
<!DOCTYPE html>
<html>
<head>
  <title>Clone: Contact Form</title>
  <style>
    .form-container {
      max-width: 500px;
      margin: 40px auto;
      background: #f2f2f2;
      padding: 20px;
      border-radius: 6px;
    }
    label {
      display: block;
      margin-top: 12px;
      font-weight: bold;
    }
    input, textarea {
      width: 100%;
      padding: 10px;
      margin-top: 4px;
      margin-bottom: 12px;
      border: 1px solid #ccc;
      border-radius: 4px;
      box-sizing: border-box;
    }
    button {
      padding: 12px 20px;
      background: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background: #45a049;
```

```
  }
 </style>
</head>
<body>
 <div class="form-container">
  <form action="#" method="post">
   <label for="name">Name</label>
   <input type="text" id="name" name="name" placeholder="Your name.." required>

   <label for="email">Email</label>
   <input type="email" id="email" name="email" placeholder="Your email.." required>

   <label for="message">Message</label>
   <textarea id="message" name="message" placeholder="Write something.." rows="6"
required></textarea>

   <button type="submit">Send</button>
  </form>
 </div>
</body>
</html>
```

Output:



Observation: The form uses clean HTML and CSS to create a simple, user-friendly contact layout. Inputs are clearly labeled, responsive, and styled with soft borders and padding. The design is centered, minimal, and uses a green submit button with hover effects for better UX.