

## Bridge course Assignment-7

**Name:**Balam Indira Priyadarsini

**Date:**2/07/25

### Problem Solving Activity 1.1 : List Explorer

**1. Program Statement:** To Create a program that manages a list of temperatures. The program should consist of the following points:

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

### 2. Algorithm:

Step 1:start program

Step 2: Create an array to store integer temperature values

Step 3: Traverse the array to print each temperature.

Step 4: Use a loop to calculate the sum of all values.

Step 5 : Calculate the average by dividing the sum by the number of elements

Step 6: Use another loop or built-in function to find the highest temperature.

Step 7: Print the sum, average, and highest temperature.

Step 8:End the program

### 3. Pseudocode:

Start

BEGIN

DECLARE temperatures as array of integers

INITIALIZE sum = 0

INITIALIZE max = temperatures[0]

FOR each temperature in temperatures DO

PRINT temperature

sum = sum + temperature

IF temperature > max THEN

max = temperature

ENDIF


ENDFOR

average = sum / number of elements in temperatures

PRINT sum, average, and max

End

#### 4. Program Code:

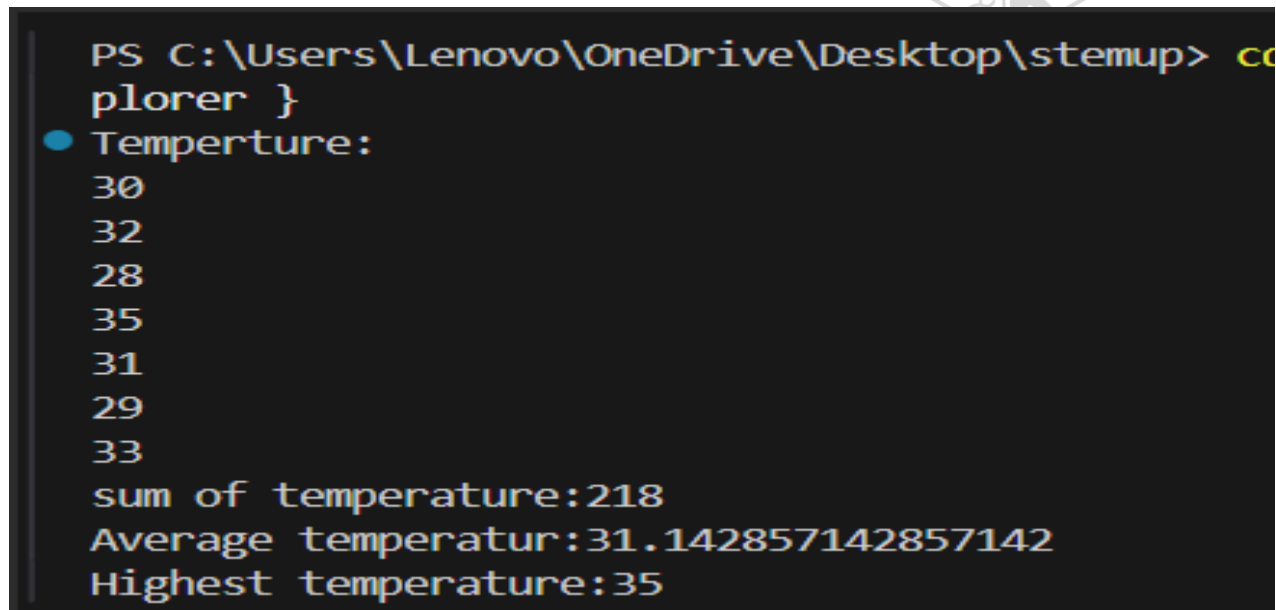


```
ListExplorer.java X
Assignment 7 > J ListExplorer.java > ListExplorer > main(String[])
1  public class ListExplorer {
    Run | Debug
2      public static void main(String[] args) {
3          int[] temperature={30,32,28,35,31,29,33};
4          int sum=0;
5          int max=temperature[0];
6          System.out.println("Temperture:");
7          for(int i = 0; i < temperature.length; i++) {
8              System.out.println(temperature[i]);
9              sum = sum + temperature[i];
10             if (temperature[i] > max) {
11                 max = temperature[i];
12             }
13         }
14         double average=(double) sum/temperature.length;
15         System.out.println("sum of temperature:"+sum);
16         System.out.println("Average temperatur:"+average);
17         System.out.println("Highest temperature:"+max);
18     }
19 }
20
```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	{30, 32, 28, 35, 31, 29, 33}	Sum: 218, Avg: 31.14, Max: 35	Sum: 218, Avg: 31.14, Max: 35	Pass
2	{25, 25, 25, 25}	Sum: 100, Avg: 25.0, Max: 25	Sum: 100, Avg: 25.0, Max: 25	Pass
3	{10, 50, 20, 40, 30}	Sum: 150, Avg: 30.0, Max: 50	Sum: 150, Avg: 30.0, Max: 50	Pass

## 6. Output



```

PS C:\Users\Lenovo\OneDrive\Desktop\stemup> cd explorer }
● Temperature:
30
32
28
35
31
29
33
sum of temperature:218
Average temperatur:31.142857142857142
Highest temperature:35

```

## 7. Observation / Reflection

This program helps in understanding how to use arrays and loops in Java. It shows how to go through all the elements in the array to find the total sum and the highest temperature. It also teaches how to calculate the average correctly using double for more accurate results. The program can be improved further by taking input from the user, sorting the temperatures, or also finding the minimum temperature.

## Problem Solving Activity 1.2 : Product of Evennumbers

**1. Program Statement:** The program is about to store array numbers and print the product of all even numbers:

- Stores numbers from 1 to 10 in an array.
- Finds and prints the product of all even numbers (2, 4, 6, 8, 10).

### 2. Algorithm:

Step 1: start program

Step 2: Create an integer array from 1 to 10.

Step 3: Initialize a variable product as 1.

Step 4: Go through each number in the array.

Step 5 : If the number is even, multiply it with product.

Step 6: Print the final value of product.

Step 7: End the program

### 3. Pseudocode

Start

BEGIN

SET product = 1

CREATE array = [1, 2, 3, ..., 10]

FOR each number in array

IF number is even

product = product \* number

ENDFOR

PRINT product

END

End



## 4. Program Code

```

J ProductofEvennumbers.java X
Assignment 7 > J ProductofEvennumbers.java > ProductofEvennumbers > main(String[])
1  public class ProductofEvennumbers {
    Run | Debug
2      public static void main(String[] args) {
3          int[] numbers={1,2,3,4,5,6,7,8,9,10};
4          int product=1;
5          for (int i=0;i<numbers.length;i++){
6              if(n int product - ProductofEvennumbers.main(String[])
7                  product*=numbers[i];
8              }
9          }
10         System.out.println("Produc of even numbers from 1 to 10 is:"+product);
11     }
12 }
13
  
```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	{1,2,3,4,5,6,7,8,9,10}	3840	3840	Pass
2	{2, 4, 6}	48	48	Pass
3	{1, 3, 5, 7, 9}	1	1	Pass

## 6. Output

```

highest temperature: 33
● if ($?) { java ProductofEvennumbers }
  Produc of even numbers from 1 to 10 is:3840
  
```

## 7. Observation / Reflection

This program helps in learning how to use arrays and loops in Java. It shows how to check if a number is even and multiply only those numbers. The logic is simple and useful for understanding conditions and iterations in programming.

## Problem Solving Activity 1.3 : Reverse my list

**1. Program Statement:** The program is about to enter string items and need to reverse the items

- Creates a string array of items.
- Prints the items in **reverse order**.

## 2. Algorithm

Step 1: Start program

Step 2: Create a string array with some items (e.g., fruits or objects).

Step 3: Loop through the array from the last item to the first.

Step 4: Print each item

Step 5 :End the program

## 3. Pseudocode

Start

BEGIN

CREATE array = ["apple", "banana", "cherry", "date"]

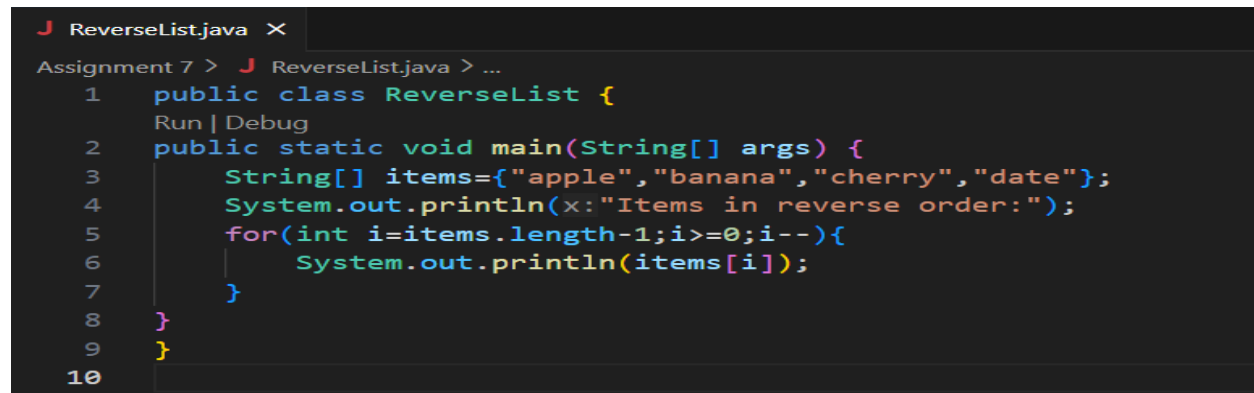
FOR i = length of array - 1 to 0

PRINT array[i]

ENDFOR

END End

## 4. Program Code



```
J ReverseList.java ×
Assignment 7 > J ReverseList.java > ...
1 public class ReverseList {
  Run | Debug
2 public static void main(String[] args) {
3     String[] items={"apple","banana","cherry","date"};
4     System.out.println("Items in reverse order:");
5     for(int i=items.length-1;i>=0;i--){
6         System.out.println(items[i]);
7     }
8 }
9 }
10
```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	{"apple", "banana", "cherry", "date"}	date, cherry, banana, apple	date, cherry, banana, apple	pass
2	{"pen", "book", "bag"}	bag, book, pen	bag, book, pen	Pass
3	{"one","two","three","four"}	four,three,two,one	four,three,two,one	Pass

## 6. Output

```

PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 7>
rseList }
Items in reverse order:
date
cherry
banana
apple
  
```

## 7. Observation / Reflection

From the program we come to know the that print the items in a reverse order by using loop statement make it to run easily without any complications of it ,if we want to add some items in the array we can add into it directly .It will reverse the items by using loop condition.

## Problem Solving Activity 1.4 : WordSearch

**Program Statement:** Need to create a predefined array and find the word present are not.

- Takes a word input from the user.
- Checks if that word exists in a predefined array.
- Prints whether the word was found or not.

## 2. Algorithm

Step 1: start program

Step 2 Define a string array with a few words.

Step 3: Ask the user to enter a word

Step 4: Loop through the array to compare each word with the user input.

Step 5: If a match is found, set a flag to true.

Step 6: After the loop, print if the word was found or not.

Step 7: End the program

### 3. Pseudocode

Start

BEGIN

CREATE wordArray = ["apple", "banana", "cherry", "date"]

GET word from user

SET found = false

FOR each item in wordArray

IF item equals user word

SET found = true

ENDFOR

IF found is true

PRINT "Word found"

ELSE

PRINT "Word not found"

End





## 4. Program Code

```

J Wordsearch.java X
Assignment 7 > J Wordsearch.java > Wordsearch > main(String[])
1  import java.util.Scanner;
2
3  public class Wordsearch {
    Run | Debug
4      public static void main(String[] args) {
5          String[] word={"apple", "banana", "cherry"};
6          Scanner input=new Scanner(System.in);
7          System.out.println(x:"Enter a word to search:");
8          String userword=input.nextLine();
9          boolean found=false;
10         for(int i=0;i<word.length;i++){
11             if(word[i].equalsIgnoreCase(userword)){
12                 found=true;
13                 break;
14             }
15         }
16         if(found){
17             System.out.println(x:"word found!");
18         }else{
19             System.out.println(x:"word not found");
20         }
21     }
22 }
23

```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Apple	{"apple", "banana", "cherry"}	Word found!	pass

2	Banana	{"apple", "banana", "cherry"}	Word found!	Pass
3	Indira	{"apple", "banana", "cherry"}	Word not found!	fail

## 6. Output

```

> ▾ TERMINAL
[icon]
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 7> cd "c
java Wordsearch }
Enter a word to search:
apple
word found!
○ PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 7>

```

## 7. Observation / Reflection

This program helps us understand how to take input from the user and check if the entered word is present in a predefined list. A list of data (array) is already given in the program, and the user can type a word to search for. Using a loop and a condition, the program checks if the entered word matches any word in the list. If a match is found, it shows that the word is present; otherwise, it says it's not found. This logic is useful in real-life situations where we need to search for something in a list.

## Problem Solving Activity 2. 1 : Implement GCD

### 1. Program Statement:

Write a Java function that uses the Euclidean Algorithm to find the GCD (Greatest Common Divisor) of two numbers. Test the function with different input pairs.

### 2. Algorithm

Step 1: start program

Step 2: Take two numbers as input.

Step 3: Euclidean method.

Step 4: Repeat until second number becomes 0 and Replace the first number with second number. Replace second number with the remainder.

Step 5: The first number is now the GCD, print the result.

Step 6 :End the program

### 3. Pseudocode

Start

FUNCTION GCD(a, b)

WHILE b is not 0

temp = b


b = a % b

a = temp

ENDWHILE

RETURN a End

### 4. Program Code



```
J GCD.java X
Assignment 7 > J GCD.java > GCD > main(String[])
1 public class GCD {
2     public static int findGCD(int a,int b) {
3         while (b != 0) {
4             int temp = b;
5             b = a % b;
6             a = temp;
7         }
8         return a;
9     }
10
11     public static void main(String[] args) {
12         System.out.println("GCD of 12 and 18 is: " + findGCD(a:12, b:18));
13         System.out.println("GCD of 25 and 15 is: " + findGCD(a:25, b:15));
14         System.out.println("GCD of 100 and 10 is: " + findGCD(a:100, b:10));
15         System.out.println("GCD of 7 and 3 is: " + findGCD(a:7, b:3));
16     }
}
```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	12,18	6	6	Pass
2	25,15	5	5	Pass
3	100,10	10	10	Pass

## 6. Output

```

• GCD of 12 and 18 is: 6
  GCD of 25 and 15 is: 5
  GCD of 100 and 10 is: 10
  GCD of 7 and 3 is: 1
  
```

## 7. Observation / Reflection

This program teaches how to use a loop and modulo operator to find the GCD using the Euclidean Algorithm. It's efficient and works well for any pair of positive integers. The function can be reused with any inputs and is helpful for problems involving number theory or simplification.

## Problem Solving Activity 2.2 : Implement LCM

### 1. Program Statement:

Use the previously created GCD function to compute the LCM (Least Common Multiple) of two numbers. Test it using the same input pairs.

### 2. Algorithm

Step 1: start program

Step 2: Use the formula:

$$\text{LCM}(a, b) = (a \times b) / \text{GCD}(a, b)$$

Step 3: Reuse the GCD function from the previous problem.

Step 4: Calculate and print the LCM.

Step 5: Repeat for different pairs.

Step 6:End the program

### 3. Pseudocode

Start

FUNCTION GCD(a, b)

WHILE  $b \neq 0$

temp = b

b = a % b

a = temp


RETURN a

FUNCTION LCM(a, b)

RETURN (a \* b) / GCD(a, b)

End

### 4. Program Code



```
ssignment 7 > J LCM.java > ...
1  public class LCM {
2      public static int findGCD(int a,int b) {
3          while(b!=0){
4              int temp=b;
5              b=a%b;
6              a=temp;
7          }return a;
8      }
9      public static int findLCM(int a,int b){
10         return (a*b)/findGCD(a,b);
11     }
12     Run | Debug
13     public static void main(String[] args){
14         System.out.println("LCM of 12 and 18:"+findLCM(a:12,b:18));
15         System.out.println("LCM of 12 and 18:"+findLCM(a:25,b:15));
16         System.out.println("LCM of 12 and 18:"+findLCM(a:100,b:10));
17         System.out.println("LCM of 12 and 18:"+findLCM(a:7,b:3));
18     }
19 }
20
```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	12,18	GCD:6,LCM:36	GCD:6,LCM:360	Pass
2	25,15	GCD:15,LCM:5	GCD:15,LCM:5	Pass
3	100,10	GCD:10,LCM:10	GCD:10,LCM:10	Pass

## 6. Output

```

PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignm
M }
LCM of 12 and 18:36
LCM of 25 and 15:75
LCM of 100 and 10:100
LCM of 7 and 3:21
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignm

```

## 7. Observation / Reflection

This program shows how we can reuse a GCD function to find the LCM using a simple formula. It teaches how to break problems into small reusable parts, which is very helpful in programming. The solution works for any pair of positive integers and is useful in many real-world calculations.

## Problem Solving Activity 2.3 : Refactor Repetitive Code

### 1. Program Statement:

1. Real-life scenario where GCD is useful:

Suppose you have 24 red flowers and 36 yellow flowers, and you want to make flower bouquets with equal number of flowers in each without mixing colors. The GCD of 24 and 36 helps you find the largest number of bouquets you can make such that each has the same number of flowers.

- GCD helps in dividing items equally into groups.

## 2. Real-life scenario where LCM is needed:

Imagine two buses arrive at a bus stop — one every 6 minutes, and another every 8 minutes. To find when both buses will arrive together again, you find the LCM of 6 and 8, which is 24 minutes.

- LCM helps in finding common times or events that repeat together.

## Problem Solving Activity 3.1 :Simple Sum calculator Web Page

### 1. Program Statement:

The Simple Sum Calculator Web Page, explained simply and clearly:

### 2. Algorithm

Step 1: start program

Step 2: Create a web page with two number input boxes and a button.

Step 3: When click on buttons it should calculate the both numbers and add

Step 4: It should show the result on the webpage.

Step 5: End the program

### 3. Pseudocode

Start

BEGIN

DISPLAY two number input boxes and a button

WHEN button is clicked

READ number1 and number2

CALCULATE  $\text{sum} = \text{number1} + \text{number2}$

DISPLAY sum

End

## 4. Program Code

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Simple Sum Calculator</title>
5  </head>
6  <body>
7
8  <h2>Simple Sum Calculator</h2>
9
10 <form onsubmit="calculateSum(); return false;">
11 <label>Enter first number:</label>
12 <input type="number" id="num1"><br><br>
13
14 <label>Enter second number:</label>
15 <input type="number" id="num2"><br><br>
16
17 <button type="submit">Calculate</button>
18 </form>
19
20 <p id="result"></p>
21
22 <script>
23   function calculateSum() {
24
25     var n1 = document.getElementById("num1").value;
26     var n2 = document.getElementById("num2").value;
27
28     var sum = Number(n1) + Number(n2);
29
30     document.getElementById("result").innerText = "Sum = " + sum;
31   }
32 </script>
33
34 </body>
35 </html>
36

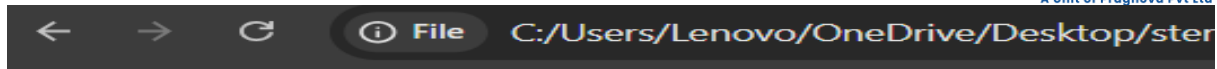
```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	5,3	Sum:8	Sum:8	Pass
2	10,0	Sum:10	Sum:10	Pass
3	-2,7	Sum:5	Sum:5	pass

## 6. Output





## Simple Sum Calculator

Enter first number:

Enter second number:

Sum = 8

### 7. Observation / Reflection

This program helps in understanding how HTML forms and JavaScript work together. It teaches how to take inputs from users, process them, and display results on a web page. This is a basic example of using JS for real-time user interaction without refreshing the page.

### Problem Solving Activity 3.2 :Web-based GCD/LCM Calculator

#### 1. Program Statement:

Web-based GCD/LCM Calculator using HTML + JavaScript, with clear explanation

#### Algorithm

Step 1: start program

Step 2: Display two input boxes for numbers.

Step 3: Add two buttons: one for GCD, one for LCM.

Step 4: When GCD button is clicked:

- Use Euclidean Algorithm to find GCD.

Step 6: When LCM button is clicked:

- Use formula  $LCM = (a \times b) / GCD$ .

Step 7: End the program

### 3. Pseudocode

Start

BEGIN

GET two input numbers from user

IF "GCD" button clicked

    COMPUTE GCD using Euclidean Algorithm

    DISPLAY result

IF "LCM" button clicked

    CALCULATE GCD

    COMPUTE  $LCM = (a \times b) / GCD$

    DISPLAY result

End



## 4. Program Code

```

Assignment 7 > GCDLCM.html > html > body > script > calculateGCD
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>GCD/LCM Calculator</title>
5  </head>
6  <body>
7    <h2>GCD and LCM Calculator</h2>
8
9    <form onsubmit="return false;">
10     <label>Enter first number:</label>
11     <input type="number" id="num1"><br><br>
12
13     <label>Enter second number:</label>
14     <input type="number" id="num2"><br><br>
15
16     <button onclick="calculateGCD()">Calculate GCD</button>
17     <button onclick="calculateLCM()">Calculate LCM</button>
18   </form>
19   <p id="result"></p>
20   <script>
21     function gcd(a, b) {
22       while (b !== 0) {
23         let temp = b;
24         b = a % b;
25         a = temp;
26       }
27       return a;
28     }
29
30     function calculateGCD() {
31       let n1 = parseInt(document.getElementById("num1").value);
32       let n2 = parseInt(document.getElementById("num2").value);
33       let result = gcd(n1, n2);
34       document.getElementById("result").innerText = "GCD = " + result;
35     }
36
37     function calculateLCM() {
38       let n1 = parseInt(document.getElementById("num1").value);
39       let n2 = parseInt(document.getElementById("num2").value);
40       let result = (n1 * n2) / gcd(n1, n2);
41       document.getElementById("result").innerText = "LCM = " + result;
42     }
43   </script>
44 </body>
45 </html>

```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	12,2	GCD:12,LCM:2	GCD:12,LCM:2	Pass
2	8,20	GCD:4,LCM:40	GCD:4,LCM:40	Pass
3	5,7	GCD:1,LCM:35	GCD:1,LCM:35	pass

## 6. Output

---

### GCD and LCM Calculator

Enter first number:

Enter second number:

LCM = 12

## 7. Observation / Reflection

This program teaches how to use HTML with JavaScript to make a working calculator. It helps understand logic building, input handling, and function reuse. It is useful for learning how math operations can be automated on web pages.

### Problem Solving Activity 3.3 :Trace the Flow

#### 1. Program Statement:

Inspect & Replicate, where you practice inspecting and copying a basic online form like a login form.

#### 2. Algorithm

Step 1: Start the program.

Step 2: Create a basic HTML structure using `<html>`, `<head>`, and `<body>`

Step 3: Design a login box with a title using `<div>` and `<h2>`.

Step 4: Add two input fields: one for username and one for password. Add a "Login" button.

Step 5: Use CSS to style the form (center it, add color, padding, etc.). Display the final login form on the webpage.

Step 6:End the program.

### 3. Pseudocode

Start

BEGIN

CREATE webpage structure (HTML)

ADD container for login box

INSIDE login box:

DISPLAY "Login" title

ADD input field for username

ADD input field for password

ADD login button

STYLE using CSS

END

End

### 4. Program Code



```

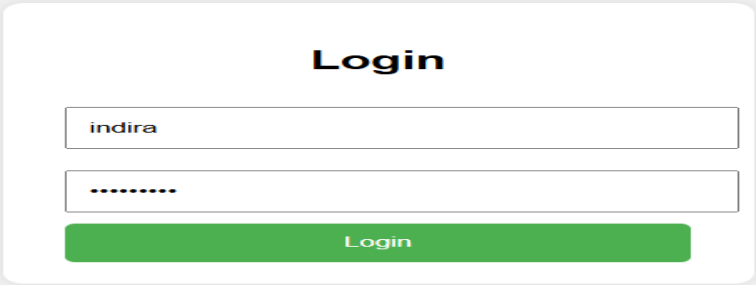
Sumcalculator.html  Replicate.html X  GCDLCM.html
Assignment 7 > Inspect > Replicate.html > html > body > div.login-box
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Login Form</title>
5    <style>
6      body {
7        font-family: Arial;
8        background-color: #f2f2f2;
9        display: flex;
10       justify-content: center;
11       align-items: center;
12       height: 100vh;
13     }
14     .login-box {
15       background-color: white;
16       padding: 20px 30px;
17       border-radius: 10px;
18       box-shadow: 0 0 10px rgba(0,0,0,0.1);
19       width: 300px;
20     }
21     .login-box h2 {
22       text-align: center;
23       margin-bottom: 20px;
24     }
25     .login-box input {
26       width: 100%;
27       padding: 10px;
28       margin: 10px 0;
29     }
30     .login-box button {
31       width: 100%;
32       padding: 10px;
33       background-color: #4CAF50;
34       color: white;
35       border: none;
36       border-radius: 5px;
37       cursor: pointer;
38     }
39     .login-box button:hover {
40       background-color: #45a049;
41     }
42   </style>
43 </head>
44 <body>
45   <div class="login-box">
46     <h2>Login</h2>
47     <input type="text" placeholder="Username" />
48     <input type="password" placeholder="Password" />
49     <button>Login</button>
50   </div>
51 </body>
52 </html>
53

```

## 5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Indira	Indira@234	Indira@234	Pass
2	Vidya	V@w9u8	V@w9u8	Pass
3	Siri	Siri@2235	Siri@2235	Pass

## 6. Output



The screenshot shows a login form with the title "Login" centered at the top. Below the title are two input fields: the first contains the text "indira" and the second contains a series of dots, indicating a password field. Below these fields is a green button labeled "Login".

## 7. Observation / Reflection

This activity helps in learning how to design a simple and clean login form using HTML and CSS. By replicating a structure, we learn how web forms are built and styled. It improves understanding of layout, input types, styling elements, and creating good-looking UI components using basic web technologies.