Learner Assignment Submission Format

Learner Details

- **Name: Shrayanth S**
- **Enrollment Number: : SU625MR011**
- **Batch / Class: June 2025 MERN**
- **Assignment: (Bridge Course Day 7)**
- **Date of Submission: 02/07/2025**

Problem Solving Activity 1.1

1. Program Statement

Problem 1.1: List Explorer

- Create an integer array of temperatures.
- Print all values.
- Find the sum, average, and highest temperature.

---

2. Algorithm

1. Initialize an integer array with temperature values.
2. Print each value in the array.
3. Initialize variables sum and maxTemp.
4. Iterate over the array:
5. Add each temperature to sum.
6. Check if the current temperature is greater than maxTemp; if so, update maxTemp.
7. Calculate the average by dividing sum by the number of elements.
8. Print the sum, average, and highest temperature.

---

3. Pseudocode

START

Declare array temperatures = [28, 32, 31, 30, 29, 33, 34]

```
  Print "Temperature values:"
  FOR each temp in temperatures
    Print temp
  END FOR

  sum ← 0
  maxTemp ← temperatures[0]

  FOR each temp in temperatures
    sum ← sum + temp
    IF temp > maxTemp THEN
      maxTemp ← temp
    END IF
  END FOR

  average ← sum / number of elements in temperatures

  Print "Sum: ", sum
  Print "Average: ", average
  Print "Highest Temperature: ", maxTemp
END
```

---

4. Program Code

ListExplorer.java

Share Run

```java
public class ListExplorer {
    public static void main(String[] args) {
        int[] temperatures = {28, 32, 31, 30, 29, 33, 34};

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

        int sum = 0;
        int maxTemp = temperatures[0];

        for (int temp : temperatures) {
            sum += temp;
            if (temp > maxTemp) {
                maxTemp = temp;
            }
        }

        double average = (double) sum / temperatures.length;

        System.out.println("\n\nSum of temperatures: " + sum);
        System.out.println("Average temperature: " + average);
        System.out.println("Highest temperature: " + maxTemp);
    }
}

```


5. Test Cases

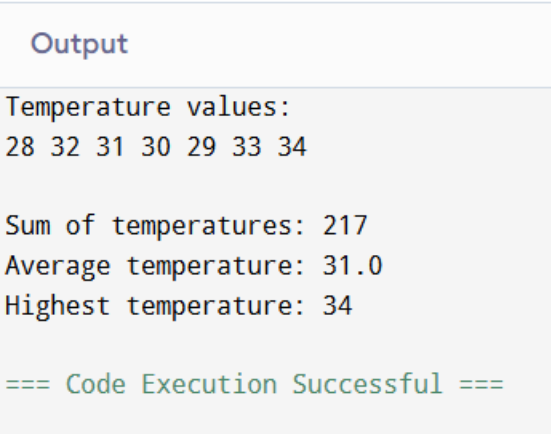
A table of test cases you used to validate your program. Include a mix of regular, boundary, and edge cases.

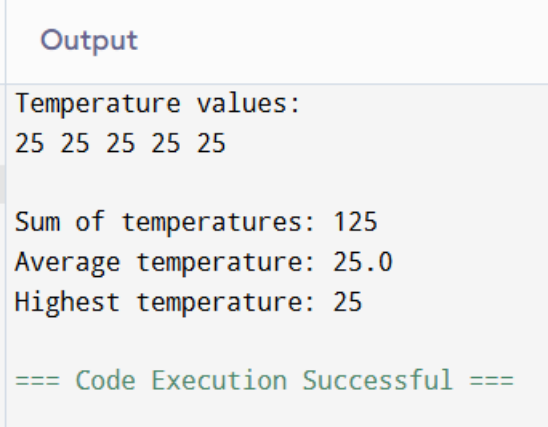
Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	[28, 32, 31, 30, 29, 33, 34]	Sum: 217, Avg: 31.0, Max: 34	Sum: 217, Avg: 31.0, Max: 34	Pass

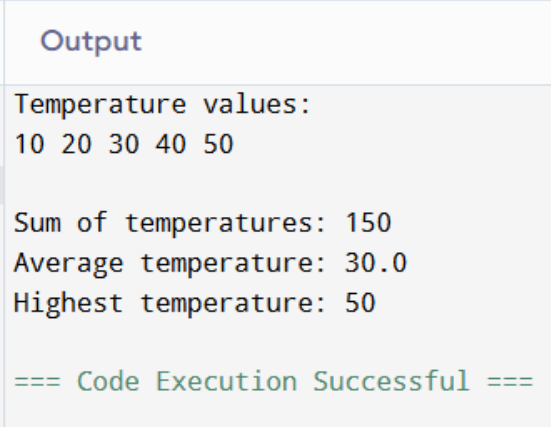
2	[25, 25, 25, 25, 25]	Sum: 125, Avg: 25.0, Max: 25	Sum: 125, Avg: 25.0, Max: 25	Pass
3	[10, 20, 30, 40, 50]	Sum: 150, Avg: 30.0, Max: 50	Sum: 150, Avg: 30.0, Max: 50	Pass

---

6. Screenshots of Output







---

7. Observation / Reflection

1. **Challenges Faced:** Initially, identifying the best way to track the maximum value and calculate the average required careful attention to index handling and division logic.
2. **Lessons Learned:** I learned how to iterate through arrays efficiently and calculate statistical values using loops.
3. **Improvements:** In the future, I would add user input for dynamic testing, and include error checks for empty arrays to make the program more robust.

Problem Solving Activity 1.2

1. Program Statement

Problem 1.2: Product of Evens

- Create an integer array from 1 to 10.
- Calculate and print the product of all even numbers.

---

2. Algorithm

1. Create an integer array with values from 1 to 10.
2. Initialize a variable product = 1.
3. Loop through the array:

   If an element is even (i.e., divisible by 2), multiply it with product.
4. Print the final value of product.

---

3. Pseudocode

```
START
 Declare array numbers = [1, 2, 3, ..., 10]
 product ← 1
 FOR each number in numbers
  IF number % 2 == 0 THEN
   product ← product * number
  END IF
 END FOR
 Print "Product of even numbers:", product
END.
```

---

4. Program Code

ProductOfEvens.java

Share Run

```java
public class ProductOfEvens {
    public static void main(String[] args) {
        int[] numbers = {1,2,3,4,5,6,7,8,9,10};
        int product = 1;

        for (int num : numbers) {
            if (num % 2 == 0) {
                product *= num;
            }
        }

        System.out.println("Product of even numbers from 1 to 10 is: " + product
            );
    }
}
```

5. Test Cases

A table of test cases you used to validate your program. Include a mix of regular, boundary, and edge cases.

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	[1 to 10]	2×4×6×8×10 = 3840	2×4×6×8×10 = 3840	Pass
2	[2, 4, 6]	2x4x6 = 48	2x4x6 = 48	Pass
3	[2,4]	2x4=8	2x4=8	Pass

6. Screenshots of Output

```
Output

Product of even numbers from 1 to 10 is: 3840

=== Code Execution Successful ===
```

```
Output

Product of even numbers from 1 to 10 is: 48

=== Code Execution Successful ===
```

```
Output

Product of even numbers from 1 to 10 is: 8

=== Code Execution Successful ===
```

7. Observation / Reflection

1. **Challenge:** Simple logic, but care was needed to ensure multiplication starts from 1, not 0.
2. **Learning:** Reinforced the importance of checking edge conditions like arrays with no even numbers.
3. **Improvements:** Allow user-defined ranges instead of fixed 1–10.

Problem Solving Activity 1.3

1. Program Statement

Problem 1.3: Reverse My List

- Create a string array of items.
- Print the list in reverse order.

---

2. Algorithm

1. Create a string array with a few item names.
2. Loop through the array in reverse using a decrementing loop.
3. Print each item during the loop.

---

3. Pseudocode

START

Declare array items = ["apple", "banana", "cherry"]

FOR i from length of items - 1 to 0

Print items[i]

END FOR

END

---

4. Program Code

ReverseMyList.java

Share Run

```java
public class ReverseMyList {
    public static void main(String[] args) {
        String[] items = {"apple", "banana", "cherry", "date"};

        System.out.println("Items in reverse order:");
        for (int i = items.length - 1; i >= 0; i--) {
            System.out.println(items[i]);
        }
    }
}
```

---

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	["apple", "banana"]	banana apple	banana apple	Pass

---

6. Screenshots of Output

```
Output

Items in reverse order:
date
cherry
banana
apple

=== Code Execution Successful ===
```

```
Output

Items in reverse order:
banana
apple

=== Code Execution Successful ===
```

---

7. Observation / Reflection

1. **Challenge:** Remembering to loop from the end toward 0.
2. **Learning:** Practice with array indexing in reverse.
3. **Improvement:** We can use Collections.reverse if converting to list in future.

Problem Solving Activity 1.4

1. Program Statement

Problem 1.4: Word Search

- Take a word as input from the user.
- Search for that word in a predefined array.
- Print whether the word was found.

---

2. Algorithm

1. Define a string array with some words.
2. Accept a word from the user.
3. Loop through the array:
4. Compare each word with the user input (case insensitive).
5. If match found, set found = true.
6. Print the result based on found.

---

3. Pseudocode

```
START
 Declare array words = ["apple", "banana", "cherry"]
 Read user input: searchWord
 found ← false

 FOR each word in words
  IF word equalsIgnoreCase searchWord THEN
   found ← true
   BREAK
  END IF
 END FOR

 IF found THEN
  Print "Word found"
```

ELSE

Print "Word not found"

END IF

END

---

4. Program Code

WordSearch.java

Share Run

```java
import java.util.Scanner;

public class WordSearch {
    public static void main(String[] args) {
        String[] words = {"apple", "banana", "cherry", "date"};
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a word to search: ");
        String inputWord = scanner.nextLine();
        boolean found = false;

        for (String word : words) {
            if (word.equalsIgnoreCase(inputWord)) {
                found = true;
                break;
            }
        }

        if (found) {
            System.out.println("Word found.");
        } else {
            System.out.println("Word not found.");
        }

        scanner.close();
    }
}
```

---

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	"banana"	Word found	Word found	Pass
2	“Ram”	Word not found	Word not found	Fail

---

6. Screenshots of Output

```
Output
Enter a word to search: banana
Word found.

=== Code Execution Successful ===
```

```
Output
Enter a word to search: Ram
Word not found.

=== Code Execution Successful ===
```


---

7. Observation / Reflection

1. **Challenge:** Ensuring comparison is case insensitive.
2. **Learning:** How to accept user input and search in arrays.
3. **Improvement:** Use Arrays.asList().contains() or binary search if sorted.

Problem Solving Activity 2.1

1. Program Statement

Problem 2.1: Implement GCD

- Write a Java function using the Euclidean Algorithm.
- Test it with multiple input pairs.

---

2. Algorithm

1. Read two integers a and b.
2. Apply the Euclidean Algorithm:
3. While b ≠ 0
    a. Store a % b in a temporary variable.
    b. Assign b to a.
    c. Assign the temporary variable to b.
4. When b becomes 0, a holds the GCD.
5. Return the value of

---

3. Pseudocode

```
FUNCTION GCD(a, b)
  WHILE b ≠ 0 DO
    temp ← a % b
    a ← b
    b ← temp
  END WHILE
  RETURN a
END FUNCTION
```

---

4. Program Code

GCDExample.java

Share Run

```java
public class GCDExample {
    public static int gcd(int a, int b) {
        while (b != 0) {
            int temp = a % b;
            a = b;
            b = temp;
        }
        return a;
    }

    public static void main(String[] args) {
        System.out.println("GCD of 48 and 18: " + gcd(48, 18));
        System.out.println("GCD of 100 and 25: " + gcd(100, 25));
        System.out.println("GCD of 7 and 3: " + gcd(7, 3));
    }
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	48, 18	6	6	Pass
2	100,25	25	25	Pass
3	7,3	1	1	Pass

6. Screenshots of Output

Output

```
GCD of 48 and 18: 6
GCD of 100 and 25: 25
GCD of 7 and 3: 1

=== Code Execution Successful ===
```

7. Observation / Reflection

1. **Challenge:** Understanding how the Euclidean Algorithm loops until remainder is 0.
2. **Learning:** This method is fast and avoids checking every number.
3. **Improvement:** Add input validation for negative numbers.

Problem Solving Activity 2.2

1. Program Statement

Problem 2.2: Implement LCM

- Reuse your GCD function to compute the LCM.
- Test it with the same pairs.

---

2. Algorithm

1. Reuse the GCD function.
2. Calculate LCM using the formula:

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

3. Return the result.

---

3. Pseudocode

```
FUNCTION LCM(a, b)
    gcd ← GCD(a, b)
    lcm ← (a × b) / gcd
    RETURN lcm
END FUNCTION
```

---

4. Program Code

LCMExample.java

Share Run

```java
public class LCMExample {

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

    public static int lcm(int a, int b) {
        return (a * b) / gcd(a, b);
    }

    public static void main(String[] args) {
        System.out.println("LCM of 48 and 18: " + lcm(48, 18));
        System.out.println("LCM of 100 and 25: " + lcm(100, 25));
        System.out.println("LCM of 7 and 3: " + lcm(7, 3));
    }
}
```

5. Test Cases

A table of test cases you used to validate your program. Include a mix of regular, boundary, and edge cases.

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	48,18	144	144	Pass
2	100,25	100	100	Pass
3	7,3	21	21	Pass

6. Screenshots of Output

Output

```
LCM of 48 and 18: 144
LCM of 100 and 25: 100
LCM of 7 and 3: 21

=== Code Execution Successful ===
```

7. Observation / Reflection

1. **Challenge:** Multiplying large integers may cause overflow if not careful.
2. **Learning:** Reusing the GCD logic made LCM simple and accurate.
3. **Improvement:** Add check for zero to avoid divide-by-zero errors.

Activity 2.3: GCD/LCM in Real Life

- Describe one scenario where GCD is useful.

  Simplifying Fractions:
  When simplifying a fraction like 60/90, the GCD (30) is used to reduce it to 2/3

- Describe another where LCM is needed.

  Scheduling Events:

  If one event happens every 3 days and another every 5 days, they will both occur together every LCM(3, 5) = 15 days.

Problem Solving Activity 3.1

1. Program Statement

Problem 3.1: Simple Sum Calculator Web Page

- Create an HTML page with a form.
- Add number inputs and a calculate button.
- Use JS to compute and display the sum.

---

2. Algorithm

1. **Create an HTML form with two <input type="number"> fields.**

2. **Add a "Calculate" button.**
3. **Use JavaScript to:**
4. **Fetch the values from both input fields.**
5. **Convert them to numbers.**
6. **Add the two numbers.**
7. **Display the result on the web page.**

---

3. Pseudocode

Display input1, input2, and Calculate button

On button click:

Get input1 and input2 values

Convert inputs to numbers

Calculate sum = input1 + input2

Display sum on page

---

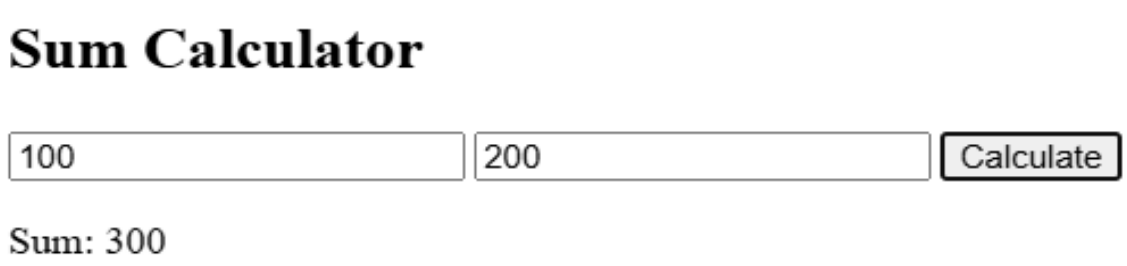
4. Program Code

Untitled
Captain Anonymous

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple Sum Calculator</title>
</head>
<body>
    <h2>Sum Calculator</h2>
    <form onsubmit="calculateSum(event)">
        <input type="number" id="num1" placeholder="Enter first number" required>
        <input type="number" id="num2" placeholder="Enter second number" required>
        <button type="submit">Calculate</button>
    </form>
    <p id="result"></p>

    <script>
        function calculateSum(event) {
            event.preventDefault();
            let a = parseFloat(document.getElementById("num1").value);
            let b = parseFloat(document.getElementById("num2").value);
            let sum = a + b;
            document.getElementById("result").innerText = "Sum: " + sum;
        }
    </script>
</body>
</html>
```

5. Screenshots of Output



Problem Solving Activity 3.2

1. Program Statement

Problem 3.2: Web-based GCD/LCM Calculator

- Extend the form to include two buttons: one for GCD, one for LCM.
- Use JS functions to perform the calculations.

2. Algorithm

- Create two input fields.
- Add three buttons: Sum, GCD, LCM.
- On button click:
- Fetch and parse the inputs.
- Use a GCD function with the Euclidean algorithm.
- Use LCM formula: (a * b) / GCD(a, b)

- Display results accordingly.

---

3. Pseudocode

Display input1, input2, buttons: Sum, GCD, LCM

On GCD button click:

Compute using Euclidean algorithm

On LCM button click:

Use formula (a * b) / GCD

Display result.

---

4. Program Code

Untitled
Captain Anonymous

HTML

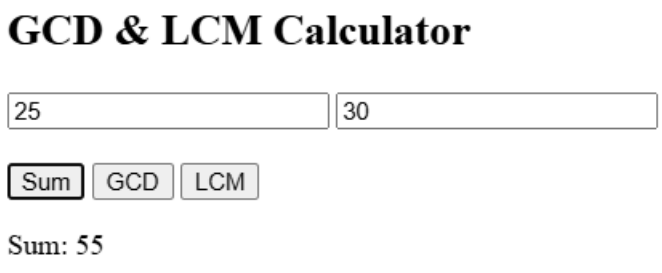
```
<!DOCTYPE html>
<html>
<head>
    <title>GCD/LCM Calculator</title>
</head>
<body>
    <h2>GCD & LCM Calculator</h2>
    <form onsubmit="event.preventDefault();">
        <input type="number" id="num1" placeholder="Enter first number" required>
        <input type="number" id="num2" placeholder="Enter second number" required>
        <br><br>
        <button onclick="calculateSum()">Sum</button>
        <button onclick="calculateGCD()">GCD</button>
        <button onclick="calculateLCM()">LCM</button>
    </form>
    <p id="result"></p>

    <script>
        function getInputValues() {
            let a = parseInt(document.getElementById("num1").value);
            let b = parseInt(document.getElementById("num2").value);
            return [a, b];
        }

        function calculateSum() {
            let [a, b] = getInputValues();
            document.getElementById("result").innerText = "Sum: " + (a + b);
        }

        function gcd(a, b) {
```

```
            while (b != 0) {
                let temp = a % b;
                a = b;
                b = temp;
            }
            return a;
        }

        function calculateGCD() {
            let [a, b] = getInputValues();
            document.getElementById("result").innerText = "GCD: " + gcd(a, b);
        }

        function calculateLCM() {
            let [a, b] = getInputValues();
            let lcm = (a * b) / gcd(a, b);
            document.getElementById("result").innerText = "LCM: " + lcm;
        }
    </script>
</body>
</html>
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	25,30	Sum = 55	Sum = 55	Pass
2	25,30	GCD = 5	GCD = 5	Pass
3	25,30	LCM = 150	LCM = 150	Pass

6. Screenshots of Output



GCD & LCM Calculator

25 30

Sum GCD LCM

GCD: 5

GCD & LCM Calculator

25 30

Sum GCD LCM

LCM: 150

Problem Solving Activity 3.3

1. Program Statement

Activity 3.3: Inspect & Replicate

- Find a form online (e.g., login or contact form).
- Inspect the HTML/CSS.
- Try to replicate the structure and styling.

.

2. Program Code

Untitled

Captain Anonymous

HTML

```html
<!DOCTYPE html>
<html>
<head>
    <title>Login Form</title>
    <style>
        body {
            font-family: Arial;
            background-color: #f0f0f0;
        }
        .login-box {
            width: 300px;
            margin: 100px auto;
            background: white;
            padding: 20px;
            box-shadow: 0px 0px 10px gray;
            border-radius: 10px;
        }
        .login-box input {
            width: 100%;
            padding: 10px;
            margin: 8px 0;
        }
        .login-box button {
            width: 100%;
            padding: 10px;
            background: #007BFF;
            color: white;
            border: none;
        }
    </style>
</head>
<body>
    <div class="login-box">
        <h3>Login</h3>
        <input type="text" placeholder="Username">
        <input type="password" placeholder="Password">
        <button>Login</button>
    </div>
</body>
</html>
```

6. Screenshots of Output