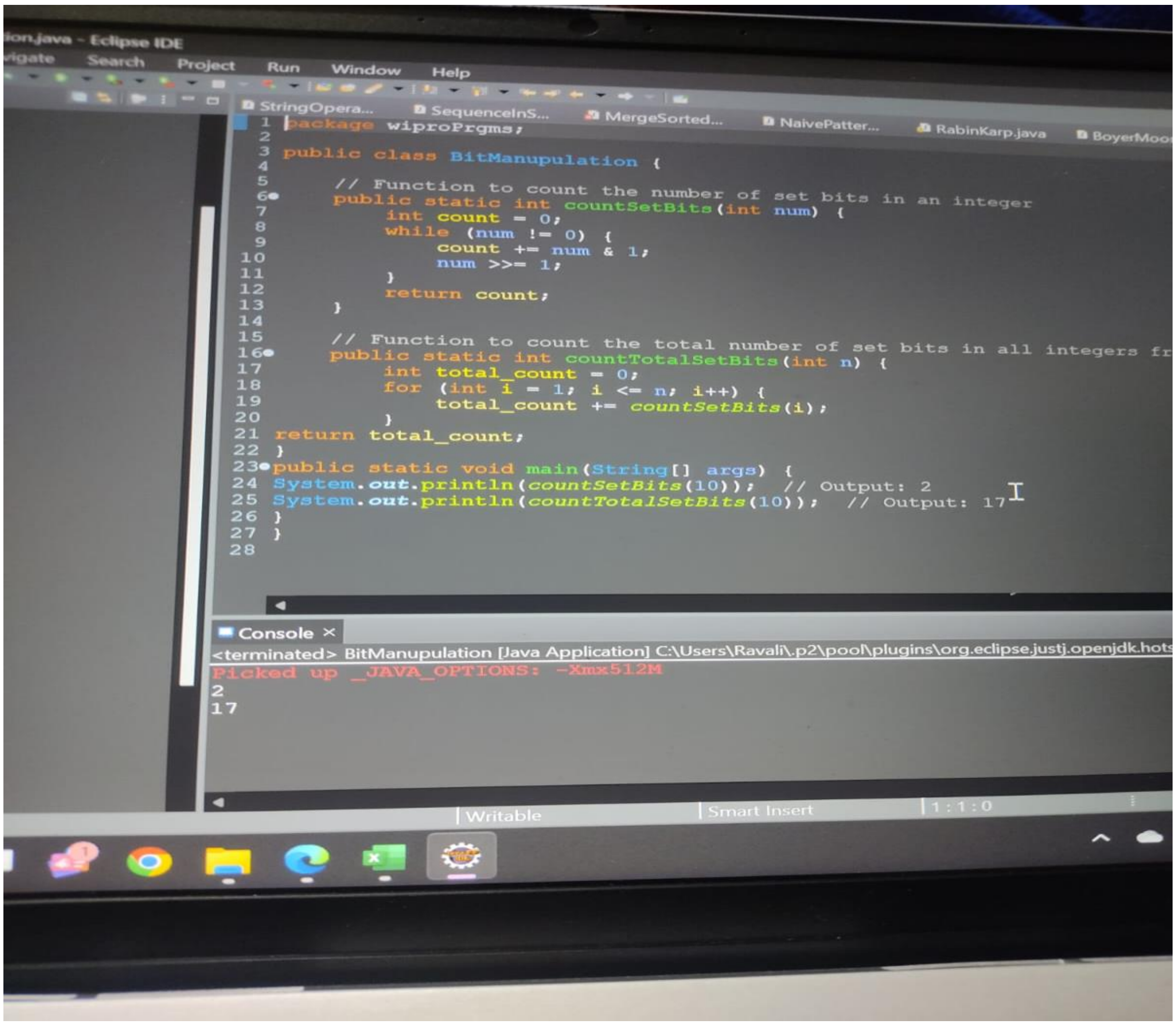
NAME :- Jangili Ravali

EMAIL :- jangiliravali9@gmail.com

Day 12 :-

Task 1: Bit Manipulation Basics

Create a function that counts the number of set bits (1s) in the binary representation of an integer. Extend this to count the total number of set bits in all integers from 1 to n.



```java
package wiproPrgms;

public class BitManupulation {

	// Function to count the number of set bits in an integer
	public static int countSetBits(int num) {
		int count = 0;
		while (num != 0) {
			count += num & 1;
			num >>= 1;
		}
		return count;
	}

	// Function to count the total number of set bits in all integers fr
	public static int countTotalSetBits(int n) {
		int total_count = 0;
		for (int i = 1; i <= n; i++) {
			total_count += countSetBits(i);
		}
return total_count;
}
public static void main(String[] args) {
System.out.println(countSetBits(10));   // Output: 2
System.out.println(countTotalSetBits(10));   // Output: 17
}
}
```

```
<terminated> BitManupulation [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
Picked up _JAVA_OPTIONS: -Xmx512M
2
17
```

This Java code provides the same functionality as the Python code, counting the number of set bits in a single integer and the total number of set bits in all integers from 1 to n.

Task 2: Unique Elements Identification

Given an array of integers where every element appears twice except for two, write a function that efficiently finds these two non-repeating elements using bitwise XOR operations.

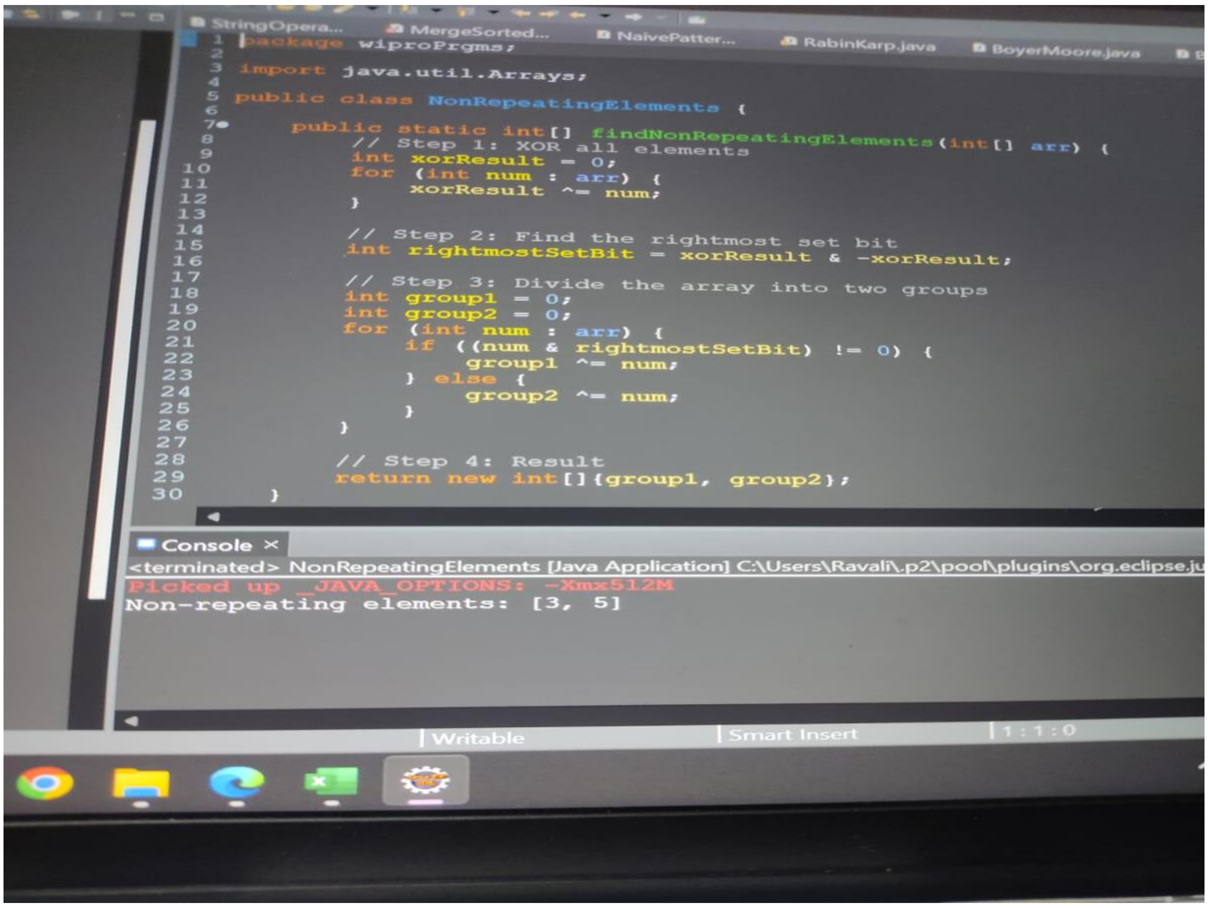
To solve this problem efficiently using bitwise XOR operations, you can follow these steps:

1. XOR all the elements in the array. This will give you the XOR of the two non-repeating elements.

2. Find the rightmost set bit in the XOR result. You can do this by performing XOR with a number having only the rightmost set bit (e.g., 1, 2, 4, 8, etc.) until you get a non-zero result.

3. Divide the elements of the array into two groups based on whether the corresponding bit is set or not.

4. XOR all the elements in each group separately. The result of each XOR operation will be one of the non-repeating elements.



```
package wiproPrgms;

import java.util.Arrays;

public class NonRepeatingElements {

    public static int[] findNonRepeatingElements(int[] arr) {
        // Step 1: XOR all elements
        int xorResult = 0;
        for (int num : arr) {
            xorResult ^= num;
        }

        // Step 2: Find the rightmost set bit
        int rightmostSetBit = xorResult & -xorResult;

        // Step 3: Divide the array into two groups
        int group1 = 0;
        int group2 = 0;
        for (int num : arr) {
            if ((num & rightmostSetBit) != 0) {
                group1 ^= num;
            } else {
                group2 ^= num;
            }
        }

        // Step 4: Result
        return new int[]{group1, group2};
    }
```

```
<terminated> NonRepeatingElements [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.ju
Picked up _JAVA_OPTIONS: -Xmx512M
Non-repeating elements: [3, 5]
```

This Java code will efficiently find the two non-repeating elements in the given array.