Date: 16/09/2025

Lab Practical #15:

Implementation of parity bit check Using C/Java language with example.

Practical Assignment #15:

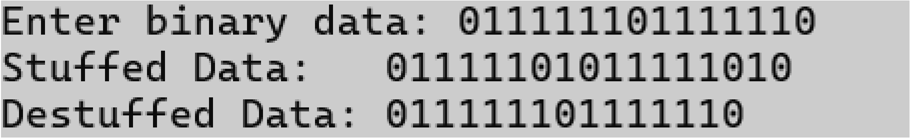
C/Java Program: Implementation of Bit stuffing Using C/Java language.

```
import java.util.Scanner;

public class BitStuffing {

    public static String stuffBits(String data) {
        StringBuilder stuffed = new StringBuilder();
        int count = 0;

        for (char bit : data.toCharArray()) {
            stuffed.append(bit);a
            if (bit == '1') {
                count++;
                if (count == 5) {
                    stuffed.append('0');
                    count = 0;
                }
            } else {
                count = 0;
            }
        }

        return stuffed.toString();
    }

    public static String destuffBits(String stuffedData) {
        StringBuilder destuffed = new StringBuilder();
        int count = 0;
```

```
      for (int i = 0; i < stuffedData.length(); i++) {
        char bit = stuffedData.charAt(i);
        destuffed.append(bit);

        if (bit == '1') {
          count++;
          if (count == 5 && i + 1 < stuffedData.length() && stuffedData.charAt(i +
  1) == '0') {
            i++;
            count = 0;
          }
        } else {
          count = 0;
        }
      }

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

        System.out.print("Enter binary data: ");
        String data = scanner.nextLine();

        String stuffed = stuffBits(data);
        System.out.println("Stuffed Data:   " + stuffed);

        String destuffed = destuffBits(stuffed);
        System.out.println("Destuffed Data: " + destuffed);

        scanner.close();
    }
}
```

1. **Enter the binary data: 011111101111110**

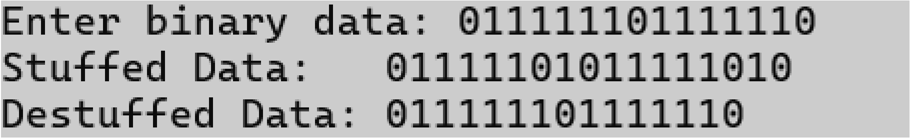
   Bit-stuffed data: 01111101011111010



```
Enter binary data: 011111101111110
Stuffed Data:   01111101011111010
Destuffed Data: 011111101111110
```

2. **Enter the binary data: 111110111111**

   Bit-stuffed data: 1 1 1 1 1 0 0 1 1 1 1 1 0 1



```
Enter binary data: 011111101111110
Stuffed Data:   01111101011111010
Destuffed Data: 011111101111110
```