CN LAB 3

Name: Rachit Nimje
Class: TY-CS-D
Roll no: 12

Batch: 1
PRN: 12210952

Title: Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Code/CRC.

Code:

```
import java.util.Scanner;

public class CRC {
   private static final int POLYNOMIAL = 0x1021;
   private static final int[] CRC_TABLE = new int[256];

   static {
      for (int i = 0; i < 256; i++) {
         int crc = i << 8;
         for (int j = 0; j < 8; j++) {
            crc = (crc & 0x8000) != 0 ? (crc << 1) ^ POLYNOMIAL : crc << 1;
         }
         CRC_TABLE[i] = crc & 0xFFFF;
      }
   }

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

      System.out.print("Enter a 7-bit ASCII code (binary): ");
      String input = scanner.nextLine();

      if (!isValidInput(input)) {
         System.out.println("Invalid input. Please enter a 7-bit binary
string.");
         return;
      }

      int crc = calculateCRC(input);
      String encodedMessage = input + String.format("%16s",
Integer.toBinaryString(crc)).replace(' ', '0');

      System.out.println("Original ASCII: " + input);
      System.out.println("Calculated CRC: " + String.format("%16s",
Integer.toBinaryString(crc)).replace(' ', '0'));
      System.out.println("Encoded message with CRC: " + encodedMessage);

      String originalEncodedMessage = encodedMessage;
```

```
        System.out.print("Do you want to introduce an error? (y/n): ");
        if (scanner.nextLine().equalsIgnoreCase("y")) {
            encodedMessage = introduceError(scanner, encodedMessage);
        }

        boolean hasError = checkError(encodedMessage);
        System.out.println("Error detected: " + hasError);

        if (hasError) {
            int errorPosition = findErrorPosition(originalEncodedMessage,
encodedMessage);
            System.out.println("Error position: " + (errorPosition != -1 ?
errorPosition : "Not found"));
        }

        scanner.close();
    }

    private static boolean isValidInput(String input) {
        return input.length() == 7 && input.matches("[01]+");
    }

    private static int calculateCRC(String message) {
        // Initial value
        int crc = 0xFFFF;
        for (char c : message.toCharArray()) {
            crc = ((crc << 8) ^ CRC_TABLE[(crc >>> 8) ^ (c - '0')]) & 0xFFFF;
        }
        return crc;
    }

    private static String introduceError(Scanner scanner, String
encodedMessage) {
        System.out.print("Enter the position to flip (1-" +
encodedMessage.length() + "): ");
        int position = scanner.nextInt();
        scanner.nextLine();
        if (position > 0 && position <= encodedMessage.length()) {
            char[] messageArray = encodedMessage.toCharArray();
            messageArray[position - 1] = messageArray[position - 1] == '0' ? '1'
: '0';
            encodedMessage = new String(messageArray);
            System.out.println("Message with error: " + encodedMessage);
        } else {
            System.out.println("Invalid position. No error introduced.");
        }
        return encodedMessage;
    }
```

```
    private static boolean checkError(String encodedMessage) {
        return calculateCRC(encodedMessage) != 0;
    }

    private static int findErrorPosition(String original, String received) {
        if (original.length() != received.length()) {
            return -1;
        }
        for (int i = 0; i < original.length(); i++) {
            if (original.charAt(i) != received.charAt(i)) {
                return i + 1;
            }
        }
        return -1;
    }
}
```

Output:

```
/home/halogen/.jdks/openjdk-22.0.2/bin/java -javaagent:/v
Enter a 7-bit ASCII code (binary): 1000001
Original ASCII: 1000001
Calculated CRC: 0101100110001110
Encoded message with CRC: 100000101011001100011110
Do you want to introduce an error? (y/n): y
Enter the position to flip (1-23): 4
Message with error: 100100101011001100011110
Error detected: true
Error position: 4
```