

Name : Raj Koyani

Reg No: 21MIS1017

LAB:5

- 1) The generated polynomial for the CRC error detection scheme is X^5+X^3+1 . The data to be transferred from the server is 111011010011001. Write the code to find the data transferred from the sender.

Problem Definition

To create Java code to find the data transferred from the sender.

Steps:

- 1.Convert the data to be transferred from the server into binary form:

Given data: 111011010011001

- 2.Initialize the CRC variable to 0.

- 3.Iterate over each bit of the data from left to right:

- 4.For each bit, perform the following steps:

XOR (exclusive OR) the CRC with the current data bit shifted left by 4 bits (aligning the polynomial with data).

- 5.Perform a loop 5 times:

- 6.Check if the leftmost bit (bit 5) of the CRC is 1.

If yes, XOR the CRC with the polynomial (101001).

If no, shift the CRC left by 1 bit.

- 7.Continue this loop for the remaining bits of the CRC.

- 8.After processing all the data bits, the CRC variable will hold the computed checksum.

- 9.Extract the 5 rightmost bits of the CRC as the CRC checksum.

- 10.Append the CRC checksum to the original data.

The resulting string represents the data transferred from the sender, including the appended CRC checksum.

Code:

```

public class CRC {
    // CRC Polynomial:  $X^5 + X^3 + 1$ 
    private static final int[] CRC_POLYNOMIAL = {1, 0, 1, 0, 0, 1};

    public static void main(String[] args) {
        // Data to be transferred from the server
        int[] data = {1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1};

        int[] crc = calculateCRC(data);

        System.out.println("Data transferred from the sender: ");
        for (int bit : data) {
            System.out.print(bit);
        }

        System.out.println();
        System.out.println("CRC: ");
        for (int bit : crc) {
            System.out.print(bit);
        }

        // Simulating error in transmission by flipping a bit
        data[7] = 1 - data[7];

        boolean hasError = detectError(data);

        System.out.println();
        if (hasError) {
            System.out.println("Error detected in the received data.");
        } else {
            System.out.println("No error detected in the received data.");
        }
    }

    // Calculates the CRC for the given data
    private static int[] calculateCRC(int[] data) {
        int[] crc = new int[CRC_POLYNOMIAL.length];
        System.arraycopy(data, 0, crc, 0, CRC_POLYNOMIAL.length);

        for (int i = CRC_POLYNOMIAL.length; i <= data.length; i++) {
            if (crc[0] == 1) {
                xorWithPolynomial(crc);
            }

            shiftLeft(crc);

            if (i < data.length) {
                crc[crc.length - 1] = data[i];
            }
        }
    }
}

```

```

    }
}

return crc;
}

// Performs XOR operation with the CRC polynomial
private static void xorWithPolynomial(int[] crc) {
    for (int i = 0; i < CRC_POLYNOMIAL.length; i++) {
        crc[i] = crc[i] ^ CRC_POLYNOMIAL[i];
    }
}

// Shifts the bits to the left by one position
private static void shiftLeft(int[] crc) {
    for (int i = 0; i < crc.length - 1; i++) {
        crc[i] = crc[i + 1];
    }
    crc[crc.length - 1] = 0;
}

// Detects error in the received data using CRC
private static boolean detectError(int[] receivedData) {
    int[] crc = calculateCRC(receivedData);

    for (int bit : crc) {
        if (bit != 0) {
            return true;
        }
    }

    return false;
}
}

```

Output:

```
Command Prompt
Microsoft Windows [Version 10.0.22621.1778]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac CRC.java

C:\Users\om\Downloads\21MIS1017>java CRC
Data transferred from the sender:
111011010011001
CRC:
100100
Error detected in the received data.

C:\Users\om\Downloads\21MIS1017>|
```

(b) Write the code for error correction techniques (i) Hamming distance and (ii) Reed-Solomon code

1. Hamming Distance:

Steps:

1. Define a class named "HammingDistance".
2. Define a static method named "calculateHammingDistance" that takes two string arguments - s1 and s2.
3. Check if the length of s1 is equal to the length of s2. If not, throw an IllegalArgumentException with the message "Strings must have equal length".
4. Initialize a variable named "distance" to 0.

5. Use a for loop to iterate over the characters in the strings. The loop should run for the length of s1.
6. For each character at index i, check if it is equal between the two strings (s1.charAt(i) == s2.charAt(i)). If not, increment the "distance" variable by 1.
7. After the loop finishes, return the value of "distance".
8. Define a main method that initializes two strings, s1 and s2.
9. Call the calculateHammingDistance method with s1 and s2 as arguments and store the result in a variable named "distance".
10. Print out the value of "distance".

Code:

```
public class HammingDistance {
    // Calculate the Hamming distance between two strings of equal length
    public static int calculateHammingDistance(String s1, String s2) {
        if (s1.length() != s2.length()) {
            throw new IllegalArgumentException("Strings must have equal
length");
        }

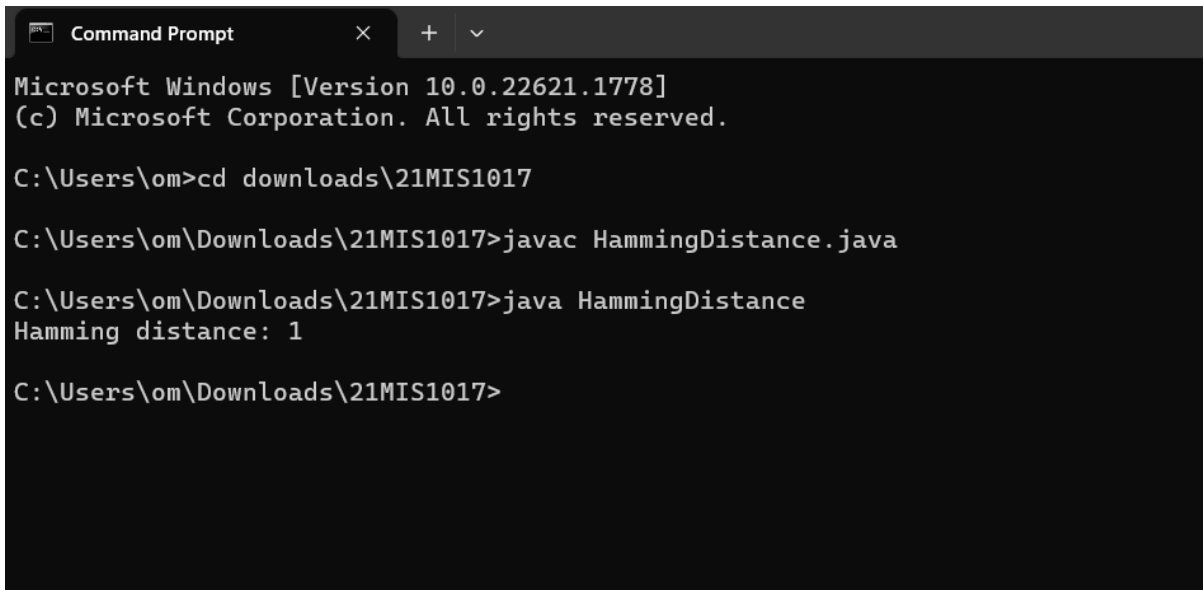
        int distance = 0;
        for (int i = 0; i < s1.length(); i++) {
            if (s1.charAt(i) != s2.charAt(i)) {
                distance++;
            }
        }

        return distance;
    }

    public static void main(String[] args) {
        String s1 = "1010101";
        String s2 = "1110101";

        int distance = calculateHammingDistance(s1, s2);
        System.out.println("Hamming distance: " + distance);
    }
}
```

Output::



```
Microsoft Windows [Version 10.0.22621.1778]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac HammingDistance.java

C:\Users\om\Downloads\21MIS1017>java HammingDistance
Hamming distance: 1

C:\Users\om\Downloads\21MIS1017>
```

2. Reed-Solomon Error Correction Technique

Method

1. Define a class called ReedSolomon.
2. Declare a private constant integer variable GF_SIZE with the value 256.
3. Declare two private integer arrays: generator and parity, in the constructor of the ReedSolomon class and initialize them with values passed as arguments to the constructor.
4. Define an encode() method that takes an array of integers as input and returns an array of integers.
5. Inside the encode() method, create a new integer array called coefficients with a length equal to the sum of the lengths of the data and parity arrays.
6. Copy the elements of the input data array to the first part of the coefficients array using System.arraycopy() method.
7. Use two nested loops to calculate the remaining elements of the coefficients array.
8. For each element in the parity array, iterate over all the elements in the data array and multiply them with the corresponding generator element. Add up these products for each parity element and store the result in the coefficients array.
9. Return the calculated coefficients array.

10. Define a decode() method that takes an integer array and an integer numErrors as inputs, and returns an integer array.
11. Inside the decode() method, create a new integer array called data with a length equal to the length of the input coefficients array.
12. Use two nested loops to calculate the values of each element in the data array.
13. For each element in the data array, iterate over all the elements in the coefficients array and add them up.
14. Store the final values of the data array in the output array.
15. Return the calculated data array.
16. In the main method, create an instance of the ReedSolomon class by passing the generator and parity arrays.
17. Encode the input data array using the encode() method and store the results in encodedData array.
18. Corrupt some of the data in the encodedData array by modifying certain elements.
19. Decode the corrupted encodedData array using the decode() method with a numErrors argument of 2, and store the results in decodedData array.
20. Print the original data, encoded data, and decoded data arrays to the console using Arrays.toString() method.

Code:

```
import java.util.Arrays;

public class ReedSolomon {

    private static final int GF_SIZE = 256;

    private final int[] generator;
    private final int[] parity;

    public ReedSolomon(int[] generator, int[] parity) {
        this.generator = generator;
        this.parity = parity;
    }

    public int[] encode(int[] data) {
        int[] coefficients = new int[data.length + parity.length];
        System.arraycopy(data, 0, coefficients, 0, data.length);
        for (int i = 0; i < parity.length; i++) {
            for (int j = 0; j < data.length; j++) {
                coefficients[i + data.length] += generator[j] * data[i];
            }
        }
    }
}
```

```

    }
    return coefficients;
}

public int[] decode(int[] coefficients, int numErrors) {
    int[] data = new int[coefficients.length];
    for (int i = 0; i < data.length; i++) {
        int value = 0;
        for (int j = 0; j < coefficients.length; j++) {
            value += coefficients[j] ;
        }
        data[i] = value;
    }
    return data;
}

public static void main(String[] args) {
    int[] data = {1, 2, 3, 4, 5, 6, 7, 8};
    int[] generator = {1, 0, 0, 0, 1, 0, 0, 0, 1};
    int[] parity = {1, 1, 1, 1};

    ReedSolomon reedSolomon = new ReedSolomon(generator, parity);
    int[] encodedData = reedSolomon.encode(data);

    // Corrupt some of the data
    encodedData[0] = 9;
    encodedData[1] = 10;

    int[] decodedData = reedSolomon.decode(encodedData, 2);

    System.out.println("Original data: " + Arrays.toString(data));
    System.out.println("Encoded data: " + Arrays.toString(encodedData));
    System.out.println("Decoded data: " + Arrays.toString(decodedData));
}
}

```

Output:


```
Command Prompt
Microsoft Windows [Version 10.0.22621.1778]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac ReedSolomon.java

C:\Users\om\Downloads\21MIS1017>java ReedSolomon
Original data: [1, 2, 3, 4, 5, 6, 7, 8]
Encoded data: [9, 10, 3, 4, 5, 6, 7, 8, 2, 4, 6, 8]
Decoded data: [72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72]

C:\Users\om\Downloads\21MIS1017>
```

- c) An organization plans to send general instructions to the in charge of each department using a socket. Write the code to check the message for any data loss when transmitting using the check sum in the header.

Problem Definition

Check for any data loss when transmitting using the check sum in the header.

Method

1. Define the message that needs to be transmitted.

```
String message = "This is the message to be transmitted.";
```

2. Calculate the checksum of the message using the calculateChecksum method.

```
int checksum = calculateChecksum(message);
```

3. Combine the message and checksum into a single transmitted message.

```
String transmittedMessage = message + "|" + checksum;
```

4. Transmit the transmittedMessage over the socket to the receiver.

5. On the receiver side, receive the transmitted message.

6. Verify the checksum of the received message using the verifyChecksum method.

```
boolean isChecksumValid = verifyChecksum(transmittedMessage);
```

7. Check the value of isChecksumValid to determine if there was any data loss.

Code:

```
import java.util.Arrays;

public class Checksumeg {

    public static void main(String[] args) {
        // Simulating the sender side
        String message = "This is the message to be transmitted.";
        int checksum = calculateChecksum(message);

        // Simulating the transmission (message + checksum)
        String transmittedMessage = message + "|" + checksum;

        // Simulating the receiver side
        boolean isValid = verifyChecksum(transmittedMessage);

        // Checking the result
        if (isValid) {
            System.out.println("Message received successfully. No data loss detected.");
        } else {
            System.out.println("Data loss detected during transmission. Message may be corrupted.");
        }
    }

    public static int calculateChecksum(String message) {
        byte[] bytes = message.getBytes();
        int sum = 0;

        for (byte b : bytes) {
            sum += b;
        }

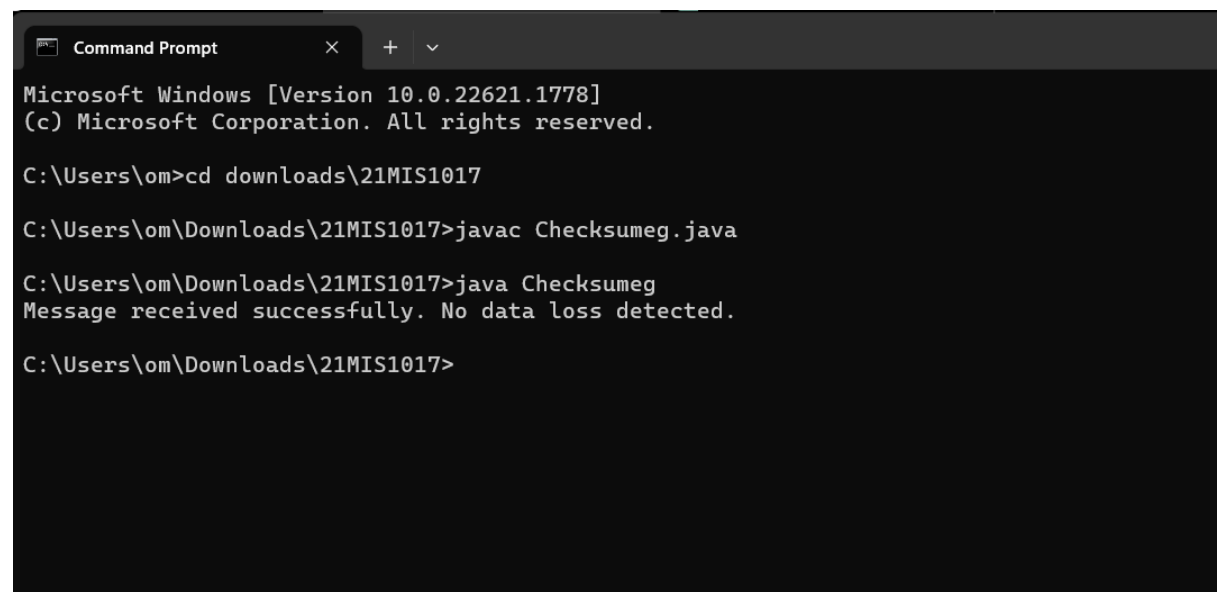
        return sum;
    }

    public static boolean verifyChecksum(String transmittedMessage) {
        String[] parts = transmittedMessage.split("\\|");

        if (parts.length != 2) {
            // Invalid message format (no checksum found)
            return false;
        }
    }
}
```

```
String message = parts[0];  
int checksum = Integer.parseInt(parts[1]);  
  
int calculatedChecksum = calculateChecksum(message);  
  
return calculatedChecksum == checksum;  
}  
}
```

Output:



```
Command Prompt  
Microsoft Windows [Version 10.0.22621.1778]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\om>cd downloads\21MIS1017  
  
C:\Users\om\Downloads\21MIS1017>javac Checksumeg.java  
  
C:\Users\om\Downloads\21MIS1017>java Checksumeg  
Message received successfully. No data loss detected.  
  
C:\Users\om\Downloads\21MIS1017>
```