EXPERIMENT NO. 4

To implement Cyclic Redundancy Check using Java

Aim:

To implement Cyclic Redundancy Check for error detection.

i. Creating a class.

ii. Implementing CRC algorithm to check error in the dataword.

Requirements:

Ubuntu operating system/ Windows operating system, Java

Algorithm:

1. Consider the sequence of data bits consisting of 1's and 0's. (Prompt user to enter number of data bits and actual data bits)
2. Consider a generator polynomial. (Prompt user to enter size of generator and actual generator bits)
3. Append n (size of generator word -1) 0's to the data word. E.g. If generator word is 5-bit long then we have to append 4-zeros to the data word.
4. Divide appended data word by the generator. This is a binary division.
5. The remainder obtained after the division in step 4 is the n bit CRC.
6. This CRC will replace the n 0's appended to the data word in step 3, to get the code word to be transmitted.
7. Display the reminder bits and final code word.
8. For receiver implementation, ask user to input received bit sequence.
9. Divide the received sequence by the generator. This is again a binary division.
10. If reminder is zero, received string has no error. If reminder is non zero, received string has error.

Procedure:

1. **Solve the CRC examples for which program output is shown.** *(To be solved on journal sheets)*

2. Open Notepad/gedit.
3. Type the source code and save it with classname.java.
4. Open Command Prompt(for Windows)/Terminal(for Ubuntu) and run the code using following:
5. To Compile - javac classname.java

6. To Run - java classname
7. Observe output in the terminal / cmd window.

Program output and Observations:

1. Attach complete CRC program with appropriate comments.
2. Produce output of program for at least two examples. (Examples that are solved on journals sheets)
   Choose two examples out of following:
   a) A bit stream 10011101 is transmitted using CRC. The generator bits are 1001. Find the transmitted string. Suppose the third bit from left is inverted during transmission, show that this error is detected at the receiver end. (Ans: 10011101 100)
   b) Consider an error detecting CRC with generator word 10101. Compute the transmitted bit sequence for data bits 1101101. (Ans: 1101101 1011). If the received bit string is 110011001100, is it acceptable?
   c) Find transmitted frame if the data is 1101011011 and generator is 10011. (Ans: 1101011011 1110). Assume one bit error in the frame transmission and show that it is detected by CRC.
   d) Generate the CRC code for data word 110010101. The divisor is 10101. At the receiver side, detect the error in following codewords: 110010101 1011 and 110010010 1011.

Conclusion:

CRC is a systematic cyclic error detection method, where data word and redundancy bits can be identified separately. It is used widely in computer networks especially in Ethernet because of its ease of implementation. CRC is mainly used with backward error correction method.

Post Experimental Exercise: *(To be written on journal sheets)*

1. Why is error detection and correction required?
2. What are different types of error detection and correction methods?
3. State advantages and disadvantages of CRC.

Code:

```
import java.util.Scanner;

public class EXP4code {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Input: Number of data bits
    System.out.println("Enter the number of data bits:");
    int dataSize = scanner.nextInt();

    // Input: Data bits
    System.out.println("Enter the data bits:");
    int[] data = new int[dataSize];
    for (int i = 0; i < dataSize; i++) {
      data[i] = scanner.nextInt();
    }

    // Input: Number of generator bits
    System.out.println("Enter the number of generator bits:");
    int generatorSize = scanner.nextInt();

    // Input: Generator bits
    System.out.println("Enter the generator bits:");
    int[] generator = new int[generatorSize];
    for (int i = 0; i < generatorSize; i++) {
      generator[i] = scanner.nextInt();
    }

    // CRC Encoding
    int[] encodedData = encodeCRC(data, generator);

    // Output: Appended data word
    System.out.println("The Appended data word is");
    printArray(encodedData);

    // Output: Transmitted code word
    System.out.println("Transmitted code word is:");
    printArray(encodedData);

    // CRC Decoding (Receiver side)
    System.out.println("At Receiver side, ");
    System.out.println("Enter the received code word:");
```

```
    int[] receivedData = new int[encodedData.length];
    for (int i = 0; i < encodedData.length; i++) {
      receivedData[i] = scanner.nextInt();
    }

    int[] remainder = divideCRC(receivedData, generator);

    // Output: Remainder after division
     System.out.println("The remainder after dividing received codeword by
generator bits is:");
    printArray(remainder);

    // Output: Check for errors
    if (isZeroArray(remainder)) {
       System.out.println("As remainder is zero, received code word has no
errors.");
    } else {
       System.out.println("As remainder is non zero, the received code word
has errors.");
    }
  }

  // CRC Encoding Algorithm
  private static int[] encodeCRC(int[] data, int[] generator) {
    int dataSize = data.length;
    int generatorSize = generator.length;
    int appendedSize = dataSize + generatorSize - 1;

    int[] appendedData = new int[appendedSize];

    // Copy data bits to the front of appendedData
    System.arraycopy(data, 0, appendedData, 0, dataSize);

    // Append zeros to the end of appendedData
    for (int i = dataSize; i < appendedSize; i++) {
      appendedData[i] = 0;
    }

    // Perform CRC Division
    int[] remainder = divideCRC(appendedData, generator);

    // Replace zeros with CRC remainder
```

```
     System.arraycopy(remainder, 0, appendedData, dataSize, generatorSize -
1);

    return appendedData;
  }

  // CRC Division Algorithm
  private static int[] divideCRC(int[] data, int[] generator) {
    int dataSize = data.length;
    int generatorSize = generator.length;

    int[] remainder = new int[generatorSize];
    System.arraycopy(data, 0, remainder, 0, generatorSize);

    for (int i = 0; i <= dataSize - generatorSize; i++) {
      if (remainder[0] == 1) {
        // XOR operation
        for (int j = 0; j < generatorSize; j++) {
          remainder[j] = remainder[j] ^ generator[j];
        }
      }

      // Shift remainder to the right
      for (int j = 0; j < generatorSize - 1; j++) {
        remainder[j] = remainder[j + 1];
      }

      // Insert next bit from data
      if (i < dataSize - generatorSize) {
        remainder[generatorSize - 1] = data[i + generatorSize];
      }
    }

    return remainder;
  }

  // Utility function to check if an array is all zeros
  private static boolean isZeroArray(int[] array) {
    for (int value : array) {
      if (value != 0) {
        return false;
      }
    }
```

```
    return true;
  }

  // Utility function to print an array
  private static void printArray(int[] array) {
    for (int value : array) {
      System.out.println(value);
    }
  }
}
```

A) A bit stream 10011101 is transmitted using CRC. The generator bits are 1001. Find the transmitted string. Suppose the third bit from left is inverted during transmission, show that this error is detected at the receiver end.

Output of A :

```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Network Lab> javac
.\EXP4code.java

PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Network Lab> java EXP4code

Enter the number of data bits:
8
Enter the data bits:
1
0
0
1
1
1
0
1
Enter the number of generator bits:
4
Enter the generator bits:
1
0
0
1
The Appended data word is
1
0
0
1
1
1
0
1
1
0
0
```

```
Transmitted code word is:
1
0
0
1
1
1
0
1
1
0
0
At Receiver side,
Enter the received code word:
1
0
1
1
1
1
0
1
1
0
0
The remainder after dividing received codeword by generator bits is:
1
0
0
0
As remainder is non zero, the received code word has errors.
```

C] Find transmitted frame if the data is 1101011011 and generator is 10011. (Ans: 1101011011 1110). Assume one bit error in the frame transmission and show that it is detected by CRC.

Output of C:

```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Network Lab> javac
.\EXP4code.java

PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Network Lab> java EXP4code

Enter the number of data bits:
10
Enter the data bits:
1
1
0
1
0
1
1
0
1
1
Enter the number of generator bits:
5
Enter the generator bits:
1
0
0
1
1
The Appended data word is
1
1
0
1
0
1
1
0
1
1
```

1
1
1
0
Transmitted code word is:
1
1
0
1
0
1
1
0
1
1
1
1
1
0
At Receiver side,
Enter the received code word:
1
1
0
1
0
1
1
0
1
1
1
1
0
1
The remainder after dividing received codeword by generator bits is:
0
0
1
1
1
As remainder is non zero, the received code word has errors.