Ex.No - 06
Date - 20/08/2024

HAMMING CODE

AIM:

To write a program to implement error detection and correction using the HAMMING code concept. Make a test run to input data streams and verify error correction features.

Error Correction at Data Link Layer:
Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is transmitted from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

Create a sender program with below features.

1. Input to the sender file should be a text of any length. Program should convert the text to Binary.
2. Apply the hamming code concept on the binary data and add redundant bits to it.
3. Save this output in a file called channel.

Create a receiver program with below features

1. Receiver program should read the input from the Channel file.
2. Apply hamming code on the binary data to check for errors.
3. If there is an error, display the position of the error.
4. Else remove the redundant bits and convert the binary data to ascii and display the output.

CODE:

```
import java.util.*;
class HammingCodeExample {
   public static void main(String args[])
   {
      int size, hammingCodeSize, errorPosition;
      int arr[];
      int hammingCode[];
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the bits size for the data.");
      size = sc.nextInt();
      arr = new int[size];
      for(int j = 0 ; j < size ; j++)
      {
         System.out.println("Enter " + (size - j) + "-bit of the data:");
         arr[size - j - 1] = sc.nextInt();
      }
      System.out.println("The data which you enter is:");
```

```
        for(int k = 0 ; k < size ; k++) {
            System.out.print(arr[size - k - 1]);
        }
        System.out.println();
        hammingCode = getHammingCode(arr);
        hammingCodeSize = hammingCode.length;

        System.out.println("The hamming code generated for your data is:");
        for(int i = 0 ; i < hammingCodeSize; i++)
        {
            System.out.print(hammingCode[(hammingCodeSize - i - 1)]);
        }
        System.out.println();
        System.out.println("For detecting error at the reciever end, enter position of a bit to alter original
data "
                + "(0 for no error):");
        errorPosition = sc.nextInt();
        sc.close();
        if(errorPosition != 0) {
            hammingCode[errorPosition - 1] = (hammingCode[errorPosition - 1] + 1) % 2;
        }
        System.out.println("Sent Data is:");
        for(int k = 0; k < hammingCodeSize; k++) {
            System.out.print(hammingCode[hammingCodeSize - k - 1]);
        }
        System.out.println();   // for next line
        receiveData(hammingCode, hammingCodeSize - arr.length);
    }
    static int[] getHammingCode(int data[]) {
        int returnData[];
        int size;
        int i = 0, parityBits = 0 ,j = 0, k = 0;
        size = data.length;
        while(i < size) {
            if(Math.pow(2, parityBits) == (i + parityBits + 1)) {
                parityBits++;
            }
            else {
                i++;
            }
        }
        returnData = new int[size + parityBits];
        for(i = 1; i <= returnData.length; i++) {
            if(Math.pow(2, j) == i) {
```

```
            returnData[(i - 1)] = 2;
            j++;
        }
        else {
            returnData[(k + j)] = data[k++];
        }
    }
    for(i = 0; i < parityBits; i++) {

        returnData[((int) Math.pow(2, i)) - 1] = getParityBit(returnData, i);
    }

    return returnData;
}
static int getParityBit(int returnData[], int pow) {
    int parityBit = 0;
    int size = returnData.length;

    for(int i = 0; i < size; i++) {

        if(returnData[i] != 2) {

            int k = (i + 1);

            String str = Integer.toBinaryString(k);
            int temp = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
            if(temp == 1) {
                if(returnData[i] == 1) {
                    parityBit = (parityBit + 1) % 2;
                }
            }
        }
    }
    return parityBit;
}
static void receiveData(int data[], int parityBits) {
    int pow;
    int size = data.length;
    int parityArray[] = new int[parityBits];
    String errorLoc = new String();
    for(pow = 0; pow < parityBits; pow++) {
        for(int i = 0; i < size; i++) {
            int j = i + 1;
```

```
            String str = Integer.toBinaryString(j);
            int bit = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
            if(bit == 1) {
               if(data[i] == 1) {
                  parityArray[pow] = (parityArray[pow] + 1) % 2;
               }
            }
         }
         errorLoc = parityArray[pow] + errorLoc;
      }
      int finalLoc = Integer.parseInt(errorLoc, 2);
      if(finalLoc != 0) {
         System.out.println("Error is found at location " + finalLoc + ".");
         data[finalLoc - 1] = (data[finalLoc - 1] + 1) % 2;
         System.out.println("After correcting the error, the code is:");
         for(int i = 0; i < size; i++) {
            System.out.print(data[size - i - 1]);
         }
         System.out.println();
      }
      else {
         System.out.println("There is no error in the received data.");
      }
      System.out.println("The data sent from the sender:");
      pow = parityBits - 1;
      for(int k = size; k > 0; k--) {
         if(Math.pow(2, pow) != k) {
            System.out.print(data[k - 1]);
         }
         else {
            pow--;
         }
      }
      System.out.println();
   }
}
```

OUTPUT :

Enter the bits size for the data.
5
Enter 5-bit of the data:
1 2 3 4 5
Enter 4-bit of the data:
Enter 3-bit of the data:
Enter 2-bit of the data:
Enter 1-bit of the data:
The data which you enter is:
12345
The hamming code generated for your data is:
112340501
For detecting error at the receiver end, enter position of a bit to alter original data (0 for no error):

RESULT:

The hamming code has been successfully run and executed.