Experiment-6

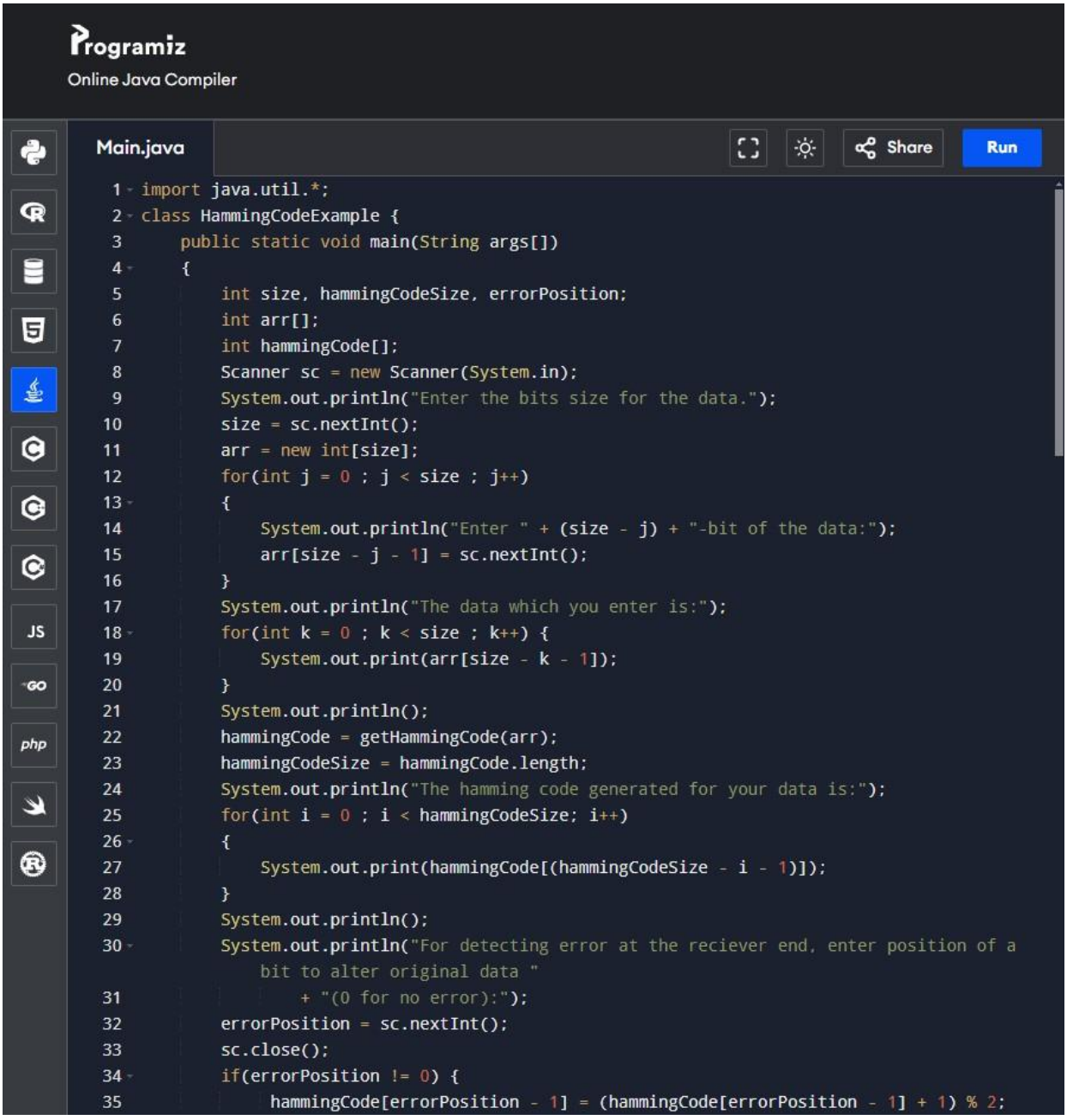
Hamming Code

AIM: Write a program to implement error detection and correction using Hamming code concept. Make a test run to input data stream and verify error correction feature.

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.

CODE:




```java
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;
```

Main.java

Share Run

```
36          }
37          System.out.println("Sent Data is:");
38          for(int k = 0; k < hammingCodeSize; k++) {
39              System.out.print(hammingCode[hammingCodeSize - k - 1]);
40          }
41          System.out.println();
42          receiveData(hammingCode, hammingCodeSize - arr.length);
43      }
44      static int[] getHammingCode(int data[]) {
45          int returnData[];
46          int size;
47          int i = 0, parityBits = 0 ,j = 0, k = 0;
48          size = data.length;
49          while(i < size) {
50              if(Math.pow(2, parityBits) == (i + parityBits + 1)) {
51                  parityBits++;
52              }
53              else {
54                  i++;
55              }
56          }
57          returnData = new int[size + parityBits];
58          for(i = 1; i <= returnData.length; i++) {
59              if(Math.pow(2, j) == i) {
60
61                  returnData[(i - 1)] = 2;
62                  j++;
63              }
64              else {
65                  returnData[(k + j)] = data[k++];
66              }
67          }
68          for(i = 0; i < parityBits; i++) {
69
70              returnData[((int) Math.pow(2, i)) - 1] = getParityBit(returnData, i);
71          }
```

Main.java

Share Run

```
73          return returnData;
74      }
75      static int getParityBit(int returnData[], int pow) {
76          int parityBit = 0;
77          int size = returnData.length;
78
79          for(int i = 0; i < size; i++) {
80              if(returnData[i] != 2) {
81                  int k = (i + 1);
82                  String str = Integer.toBinaryString(k);
83                  int temp = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
84                  if(temp == 1) {
85                      if(returnData[i] == 1) {
86                          parityBit = (parityBit + 1) % 2;
87                      }
88                  }
89              }
90          }
91          return parityBit;
92      }
93
94      static void receiveData(int data[], int parityBits) {
95          int pow;
96          int size = data.length;
97          int parityArray[] = new int[parityBits];
98          String errorLoc = new String();
99          for(pow = 0; pow < parityBits; pow++) {
100             for(int i = 0; i < size; i++) {
101                 int j = i + 1;
102
103                 String str = Integer.toBinaryString(j);
104
105                 int bit = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
106                 if(bit == 1) {
107                     if(data[i] == 1) {
108                         parityArray[pow] = (parityArray[pow] + 1) % 2;
```

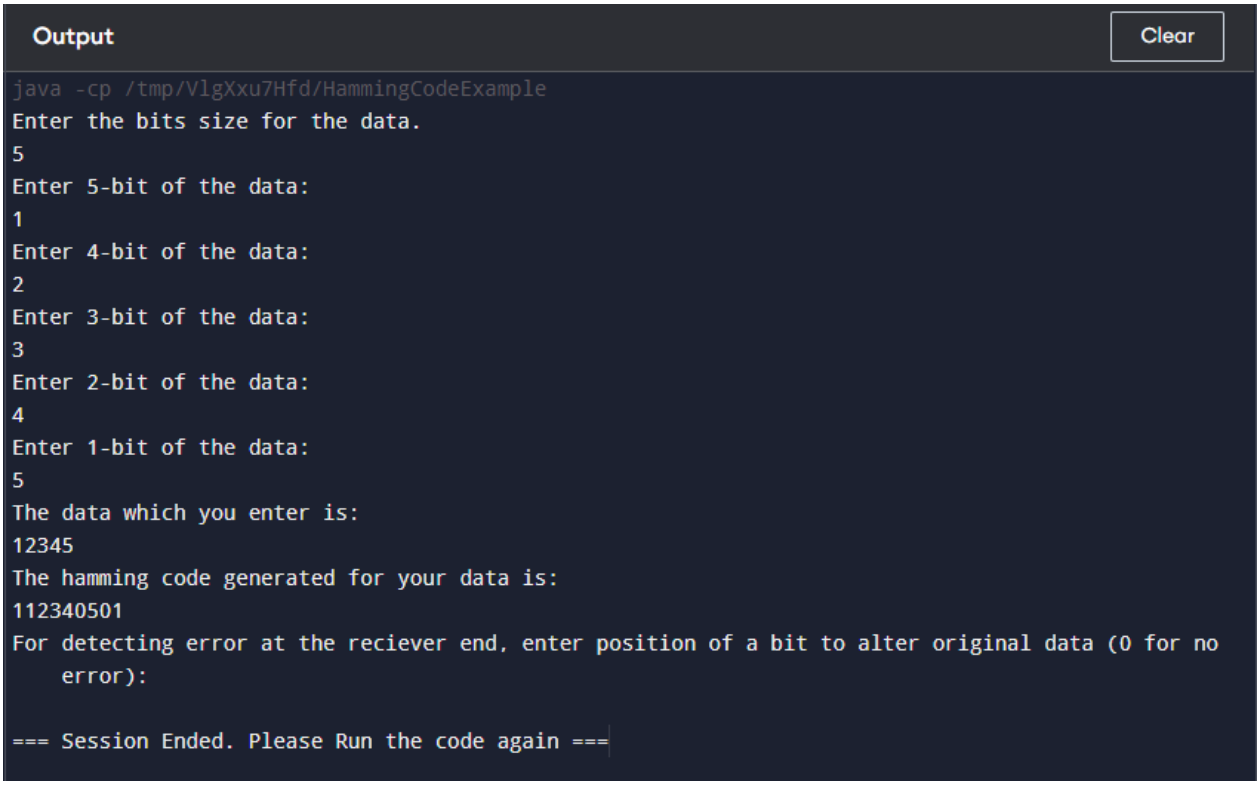
Main.java

Share Run

```
                    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:



Result:

The program to implement error detection and correction using Hamming code concept is executed successfully.