7. Write a program for error detecting code using CRC-CCITT (16-bits).

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The CRC calculation or cyclic redundancy check was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. Also each data block on your hard disk has a CRC value attached to it. Modern computer world cannot do without these CRC calculations. So let's see why they are so widely used. The answer is simple; they are powerful, detect many types of errors and are extremely fast to calculate especially when dedicated hardware chips are used.
The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation— by dividing it by 19 or 10011. Please note that 19 is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.

```
                   1 0 1
          ----------------
10011 /  1 1 0 1 1 0 1
            1 0 0 1 1 |  |
          --------- ----|  |
              1 0 0 0 0 |
              0 0 0 0 0 |
            -------------- |
              1 0 0 0 0 1
                1 0 0 1 1
            -----------------
                   1 1 1 0 = 14 remainder
```

- The message bits are appended with c zero bits; this augmented message is the dividend
- A predetermined c+1-bit binary sequence, called the generator polynomial, is the divisor
- The checksum is the c-bit remainder that results from the division operation

With decimal calculations you can quickly check that 109 divided by 19 gives a quotient of 5 with 14 as the remainder. But what we also see in the scheme is that every bit extra to check only costs one binary comparison and in 50% of the cases one binary subtraction. You can easily increase the number of bits of the test data string—for example to 56 bits if we use our example value "Lammert"—and the result can be calculated with 56 binary comparisons and an average of 28 binary subtractions. This can be implemented in hardware directly with only very few transistors involved. Also software algorithms can be very efficient.
All of the CRC formulas you will encounter are simply checksum algorithms based on modulo-2 binary division where we ignore carry bits and in effect the subtraction will be equal to an exclusive or operation. Though some differences exist in the specifics across different CRC formulas, the basic mathematical process is always the same:
Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder.
So, for example, you‘d use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC - CCITT	**CRC – 16**	**CRC - 32**
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	11000000000000101	100000100110000010001110110110111

International Standard CRC Polynomials

Source Code:

```
import java.util.*;

public class crc {
        void div(int a[],int k)
        {
                int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};
                System.out.print("The divisor is:");
                for(int i=0; i<17;i++)
                        System.out.print(gp[i]);
                int count=0;
                for(int i=0;i<k;i++)
                {
                        if(a[i]==1)
                        {
                                for(int j=i;j<17+i;j++)
                                {
                                        a[j]=a[j]^gp[count++];
                                }
                                count=0;
                        }
                }
        }

        public static void main(String[] args)
        {
                int a[]=new int[100];
                int b[]=new int[100];
                int len,k;
                int flag=0;
                crc ob = new crc();
                System.out.println("Transmitter side");
                System.out.println("------------------");
                System.out.println("Enter the length of Data Frame:");
                Scanner sc = new Scanner(System.in);
                len=sc.nextInt();
                k= len;
                System.out.println("Enter the Message:");
                for(int i=0;i<len;i++)
                {
                        a[i]=sc.nextInt();
                }
                for(int i=0;i<16;i++)
                {
                        a[len++]=0;
                }
                for(int i=0;i<len;i++)
                {
                        b[i]=a[i];
                }
                ob.div(a,k);
                System.out.print("\nThe CRC at the sender:");
                for(int i=k; i<len;i++)
                        System.out.print(a[i]);
                for(int i=0;i<len;i++)
                        a[i]=a[i]^b[i];
                System.out.print("\nData to be transmitted: ");
                for(int i=0;i<len;i++)
                {
                        System.out.print(a[i]+" ");
                }

                System.out.println("\n\nReciever side");
                System.out.println("------------------");
                System.out.println("Enter the Reveived Data: ");
                for(int i=0;i<len;i++)
                {
                        a[i]=sc.nextInt();
                }
                ob.div(a, k);
                System.out.print("\nThe CRC at the reciever:");
                for(int i=k; i<len;i++)
```

```
                         System.out.print(a[i]);
                for(int i=0;i<len;i++)
                {
                        if(a[i]!=0)
                        {
                                flag=1;
                                break;
                        }
                }
                System.out.print("\nResult of CRC error detection : ");
                if(flag==1)
                        System.out.println("Error in the data - Resend the data again");
                else
                        System.out.println("Data is received  successfully");
        }
}
```

OUTPUT

```
cs@cs-desktop:~$ gedit crc.java
cs@cs-desktop:~$ javac crc.java
cs@cs-desktop:~$ java crc
```

<u>Output 1:</u>

```
Transmitter side
------------------
Enter the length of Data Frame:
4
Enter the Message:
1 0 1 1
The divisor is:10001000000100001
The CRC at the sender:1011000101101011
Data to be transmitted: 1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1

Reciever side
------------------
Enter the Reveived Data:
1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1
The divisor is:10001000000100001
The CRC at the reciever:0000000000000000
Result of CRC error detection : Data is received  successfully
```

<u>Output 2:</u>

```
Transmitter side
------------------
Enter the length of Data Frame:
4
Enter the Message:
1 0 0 1
The divisor is:10001000000100001
The CRC at the sender:1001000100101001
Data to be transmitted: 1 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1

Reciever side
------------------
Enter the Reveived Data:
1 0 1 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1
The divisor is:10001000000100001
The CRC at the reciever:0010000001000010
Result of CRC error detection : Error in the data - Resend the data again
```