**DCCN Lab 9**

**Error Detection and Correction**

**CRC**

import java.util.*\**;

class CRC{

    public static void main(String args[]) {

        Scanner scan = new Scanner(System.in);

        int size;

        System.out.println("Enter the size of the data array: ");

        size = scan.nextInt();

        int data[] = new int[size];

        System.out.println("Enter data bits in the array one by one: ");

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

            System.out.println("Enter bit " + (1+i) + ":");

            data[i] = scan.nextInt();

        }

        System.out.println("Enter the size of the divisor array:");

        size = scan.nextInt();

        int divisor[] = new int[size];

        System.out.println("Enter divisor bits in the array one by one: ");

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

            System.out.println("Enter bit " + (1+i) + ":");

            divisor[i] = scan.nextInt();

        }

        int rem[] = divideDataWithDivisor(data, divisor);

        for(int i = 0; i < rem.length-1; i++) {

            System.out.print(rem[i]);

        }

        System.out.println("\nGenerated CRC code is: ");

        for(int i = 0; i < rem.length-1; i++) {

            System.out.print(rem[i]);

        }

        System.out.println("\nData + CRC Code is :");

        for(int i = 0; i < data.length; i++) {

            System.out.print(data[i]);

        }

        for(int i = 0; i < rem.length-1; i++) {

            System.out.print(rem[i]);

        }

        System.out.println();

        scan.close();

    }

    static int[] divideDataWithDivisor(int oldData[], int divisor[]) {

        int rem[] = new int[divisor.length];

        int i;

        int data[] = new int[oldData.length + divisor.length];

        System.arraycopy(oldData, 0, data, 0, oldData.length);

        System.arraycopy(data, 0, rem, 0, divisor.length);

        for(i = 0; i < oldData.length; i++) {

            System.out.println((i+1) + ".) First data bit is : "+ rem[0]);

            System.out.print("Remainder : ");

            if(rem[0] == 1) {

                for(int j = 1; j < divisor.length; j++) {

                    rem[j-1] = exorOperation(rem[j], divisor[j]);

                    System.out.print(rem[j-1]);

                }

            }

            else {

                for(int j = 1; j < divisor.length; j++) {

                    rem[j-1] = exorOperation(rem[j], 0);

                    System.out.print(rem[j-1]);

                }

            }

            rem[divisor.length-1] = data[i+divisor.length];

            System.out.println(rem[divisor.length-1]);

        }

        return rem;

    }

    static int exorOperation(int x, int y) {

        if(x == y) {

            return 0;

        }

        return 1;

    }

}

**Checksum**

def checksum(data, bit):

    m = (1 << bit) - 1

    cs = 0

    while data > 0:

        cs += data & m

        data >>= bit

    cs = (cs & m) + (cs >> bit)

    r\_cs = 0

    while cs > 0:

        r\_cs <<= 1

        r\_cs |= cs & 1

        cs >>= 1

    return bin(r\_cs)

data = 0b100010001100

print("Checksum: ", cs(data, 4))

**Hamming Code**

import java.util.*\**;

public class HammingCode

{

    public static void main(String[] args)

    {

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter number of bits of the message");

        int n=Integer.parseInt(sc.nextLine());

        int a[]=new int[n];

        for(int i=0;i<n;i++)

        {

            System.out.println("Enter bit number "+(i+1));

            a[i]=Integer.parseInt(sc.nextLine());

        }

        int r=1;

        while(Math.pow(2,r)<(n+r+1))

        {

            r++;

        }

        int b[]=new int[n+r];

        int k=0;

        while(Math.pow(2,k)<(n+r))

        {

            k++;

        }

        k--;

        int t=n;

        for(int i=0;i<(n+r);i++)

        {

            int flag=0;

            for(int j=0;j<=k;j++)

            {

                if(i==Math.pow(2,j)-1)

                flag=1;

            }

            if(flag==0)

            b[i]=a[--t];

        }

        t=k;

        for(int i=0;i<=k;i++)

        {

            int c=0;

            for(int j=(int)Math.pow(2,i);j<=(n+r);j++)

            {

                String ch=Integer.toBinaryString(j);

                if(Integer.toBinaryString(j).length()<Integer.toBinaryString(n+r).length())

                {

                    for(int l=Integer.toBinaryString(n+r).length();l>=ch.length();l--)

                    ch="0"+ch;

                }

                if(ch.substring(ch.length()-i-1,ch.length()-i).equals("1")&&b[j-1]==1)

                {

                    c++;

                }

            }

            if(c%2==0)

            b[(int)Math.pow(2,i)-1]=0;

            else

            b[(int)Math.pow(2,i)-1]=1;

        }

        for(int i=n+r-1;i>=0;i--)

        {

            System.out.print(b[i]+" ");

        }

    }

}