Title: Write a program for error detection and correction for 7/8 bits ASCII codes using CRC.

CRC Bit Technic

Cyclic Redundancy Check (CRC) An error detection mechanism in which a special number is appended to a block of data in order to detect any changes introduced during storage (or transmission). The CRe is recalculated on retrieval (or reception) and compared to the value originally transmitted, which can reveal certain types of error. For example, a single corrupted bit in the data results in a one-bit change in the calculated CRC, but multiple corrupt bits may cancel each other out.

A CRC is derived using a more complex algorithm than the simple CHECKSUM, involving MODULO ARITHMETIC (hence the 'cyclic' name) and treating each input word as a set of coefficients for a polynomial.

• CRC is more powerful than VRC and LRC in detecting errors.

• At the destination, the incoming data unit *i.e.* data + CRC is divided by the same number (predetermined binary divisor).

• If the remainder after division is zero then there is no error in the data unit & receiver accepts it.

• If remainder after division is not zero, it indicates that the data unit has been damaged in transit and therefore it is rejected.

• This technique is more powerful than the parity check and checksum error detection.

• CRC is based on binary division. A sequence of redundant bits called CRC or CRC remainder is appended at the end of a data unit such as byte.

Requirements of CRC:

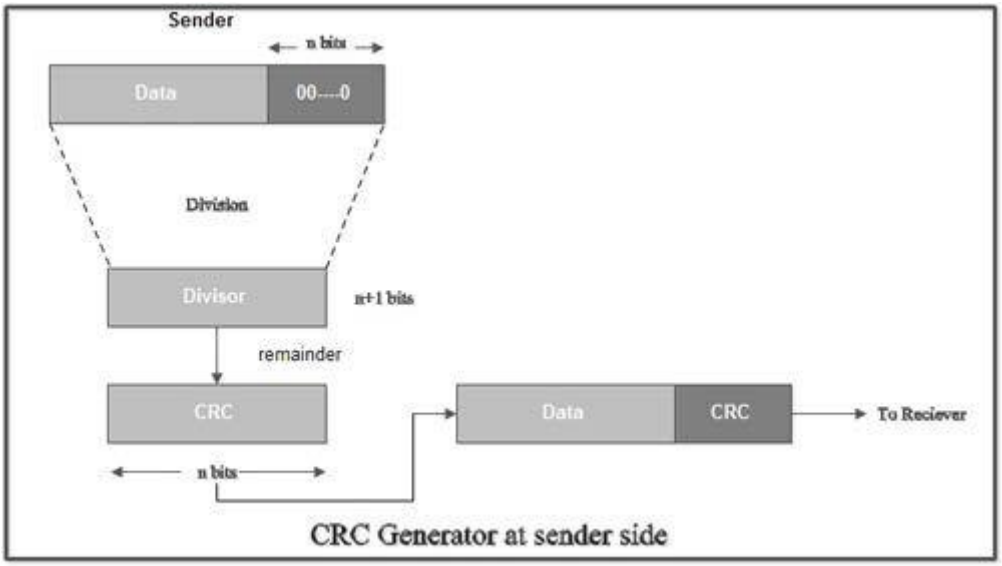
A CRC will be valid if and only if it satisfies the following requirements:

1. It should have exactly one less bit than divisor.
2. Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

• The various steps followed in the CRC method are

1. A string of n as is appended to the data unit. The length of predetermined divisor is n+ 1.

2. The newly formed data unit *i.e.* original data + string of n as are divided by the divisor using binary division and remainder is obtained. This remainder is called CRC.



3. Now, string of n Os appended to data unit is replaced by the CRC remainder (which is also of n bit).

4. The data unit + CRC is then transmitted to receiver.

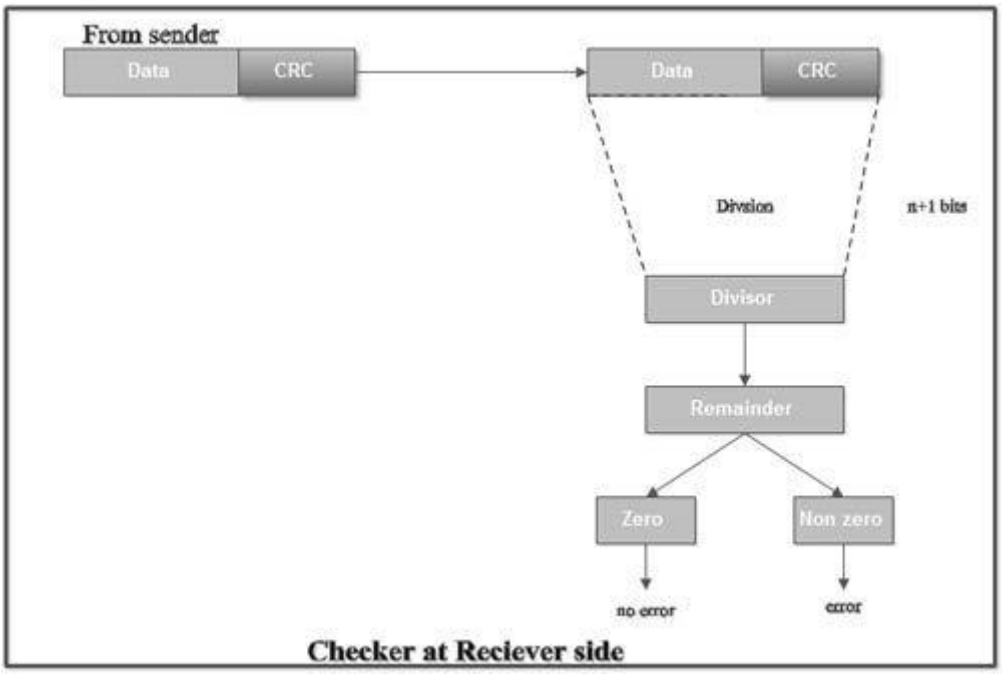
5. The receiver on receiving it divides data unit + CRC by the same divisor & checks the remainder.

6. If the remainder of division is zero, receiver assumes that there is no error in data and it accepts it.

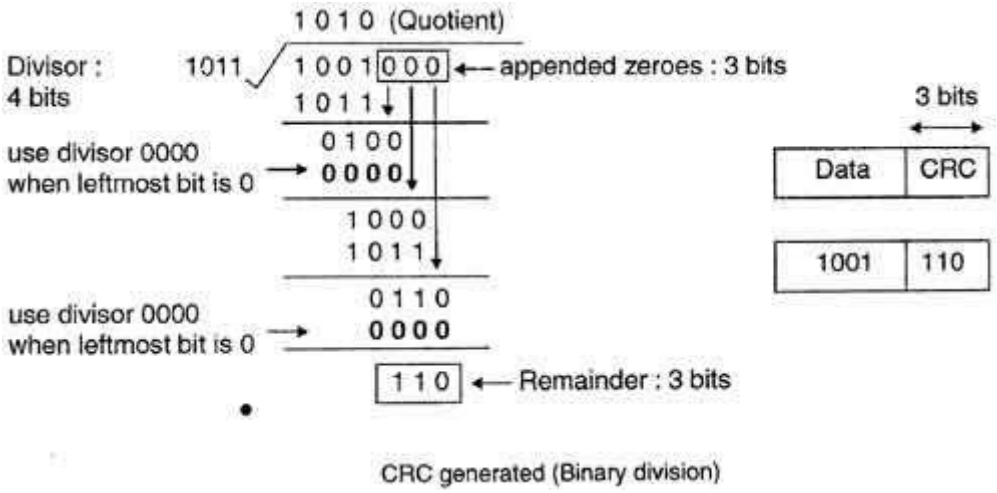
7. If remainder is non-zero then there is an error in data and receiver rejects it.

• For example, if data to be transmitted is 1001 and predetermined divisor is 1011. The procedure given below is used:

1. String of 3 zeroes is appended to 1011 as divisor is of 4 bits. Now newly formed data is 1011000.



1. Data unit 1011000 is divided by 1011.



2. During this process of division, whenever the leftmost bit of dividend or remainder is 0, we use a string of Os of same length as divisor. Thus in this case divisor 1011 is replaced by 0000.

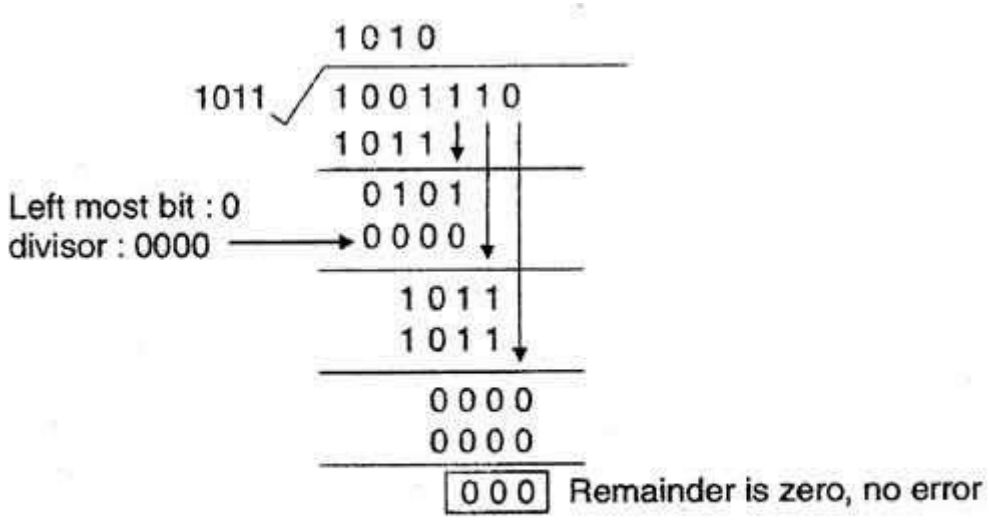
3. At the receiver side, data received is 1001110.

4. This data is again divided by a divisor 1011.

5. The remainder obtained is 000; it means there is no error.

• CRC can detect all the burst errors that affect an odd number of bits.

• The probability of error detection and the types of detectable errors depends on the choice of divisor.



CRC decoded (binary division)

• Thus two major requirement of CRC are:

(a) CRC should have exactly one bit less than divisor.

(b) Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

Code:-

```
import java.util.*;
class CRC{
    public static void main(String args[]){
        int frame[]=new int[20];
        int generator[]=new int[20];
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter Frame Size: ");
        int frame_size=sc.nextInt();
        System.out.print("\nEnter Frame Elements \n");
        for(int i=0;i<frame_size;i++){
            System.out.print("Enter Frame "+(i+1)+":- ");
            frame[i]=sc.nextInt();
        }
        System.out.print("\nEnter Generator Size:- ");
        int generator_size=sc.nextInt();
        System.out.print("Enter Generator Elements \n");
        for(int i=0;i<generator_size;i++){
            System.out.print("Enter Generator "+(i+1)+":- ");
            generator[i]=sc.nextInt();
        }

        System.out.println("\nSender Side: ");
        System.out.print("Frame:-  ");
        for(int i=0;i<frame_size;i++){
            System.out.print(frame[i]);
        }
        System.out.print("\nGenerator:-  ");
        for(int i=0;i<generator_size;i++){
            System.out.print(generator[i]);
        }

        System.out.println("Number of 0's to append:- "+(generator_size-1));
        System.out.print("Dividend(Modified data):- ");
        for(int i=frame_size;i<frame_size+generator_size-1;i++){
            frame[i]=0;
        }
        for(int i=0;i<frame_size+generator_size-1;i++){
            System.out.print(frame[i]);
        }
        System.out.println();
        for(int j=0;j<generator_size;j++){
              System.out.print(generator[j]);
        }
        System.out.print("_|");
        for(int j=0;j<frame_size+generator_size-1;j++){
```

```
            System.out.print(frame[j]);
        }
        int remainder[]=new int[4];
        int counter=0;
        String spacing="      ";
        for(int i=0;i<frame_size+generator_size-1;i++){
            if(counter>3){

                if(remainder[0]==generator[0] && remainder[0]!=0){

                    System.out.print("\n"+spacing);
                    System.out.print("-");
                    System.out.print("\n"+spacing);
                    for(int j=0;j<counter;j++){
                        System.out.print(generator[j]);
                    }
                    System.out.print("\n"+spacing+"----");

                    System.out.print("\n"+spacing);


                    for(int j=0;j<4;j++){

                        if(remainder[j]==generator[j]){
                            remainder[j]=0;
                            System.out.print("0");
                        }
                        else{
                            remainder[j]=1;
                            System.out.print("1");
                        }


                    }
                    remainder[0]=remainder[1];
                    remainder[1]=remainder[2];
                    remainder[2]=remainder[3];
                    remainder[3]=frame[i];
                    System.out.print(remainder[3]);
                }else{
                    //int temp[]=new int[4];
                    for(int j=0;j<counter-1;j++){
                        remainder[j]=remainder[j+1];

                    }
                   // remainder=temp;
                    remainder[3]=frame[i];

                    System.out.print("\n"+spacing);
```

```
                System.out.print("-");
                System.out.print("\n"+spacing+"0");
                System.out.print("\n"+spacing+"----");
                spacing+=" ";
                System.out.print("\n"+spacing);
                for(int j=0;j<counter;j++){
                    System.out.print(remainder[j]);
                }
            }
        }
        else{
            remainder[counter]=frame[i];
            counter++;
            System.out.print("\n"+spacing);
            System.out.print("-");
            System.out.print("\n"+spacing+"0");
            System.out.print("\n"+spacing+"----");

            System.out.print("\n"+spacing);
            for(int j=0;j<counter;j++){
                System.out.print(remainder[j]);
            }

        }
    }
    System.out.print("\n\nRemainder is : ");
    for(int i=1;i<remainder.length;i++){
        System.out.print(remainder[i]);
    }
    int j=1;
    for(int i=frame_size;i<frame_size+remainder.length-1;i++){
        frame[i]=remainder[j];
        j++;
    }
    System.out.print("\nData after apending reaminder :-" );
    for(int i=0;i<frame_size+remainder.length-1;i++){
        System.out.print(frame[i]);
    }

    System.out.print("\nDate Sent to Reciever\n\n\n");




    System.out.println("Reciever Side\n\n");
    System.out.print("Data Recived:- ");
    for(int i=0;i<frame_size+remainder.length-1;i++){
        System.out.print(frame[i]);
```

```
        }

        boolean result=checkData(frame,generator,frame_size,generator_size);
        if(!result){
            System.out.println("\nData is Corrupted");
        }else{
            System.out.println("\nData is Correct");
        }

    }



 static boolean checkData(int frame[],int generator[],int frame_size,int
generator_size){
    System.out.println();
    for(int j=0;j<generator_size;j++){
          System.out.print(generator[j]);
    }
    System.out.print("_|");
    for(int j=0;j<frame_size+generator_size-1;j++){
          System.out.print(frame[j]);
    }
        int remainder[]=new int[4];
        int counter=0;
        String spacing="      ";
        for(int i=0;i<frame_size+generator_size;i++){
            if(counter>3){

                if(remainder[0]==generator[0] && remainder[0]!=0){

                    System.out.print("\n"+spacing);
                    System.out.print("-");
                    System.out.print("\n"+spacing);
                    for(int j=0;j<counter;j++){
                        System.out.print(generator[j]);
                    }
                    System.out.print("\n"+spacing+"----");

                    System.out.print("\n"+spacing);


                    for(int j=0;j<4;j++){

                        if(remainder[j]==generator[j]){
                            remainder[j]=0;
                            System.out.print("0");
                        }
                        else{
```

```
                        remainder[j]=1;
                        System.out.print("1");
                    }


                }
                remainder[0]=remainder[1];
                remainder[1]=remainder[2];
                remainder[2]=remainder[3];
                remainder[3]=frame[i];
                System.out.print(remainder[3]);
            }else{
                //int temp[]=new int[4];
                for(int j=0;j<counter-1;j++){
                    remainder[j]=remainder[j+1];

                }
               // remainder=temp;
                remainder[3]=frame[i];

                System.out.print("\n"+spacing);
                System.out.print("-");
                System.out.print("\n"+spacing+"0");
                System.out.print("\n"+spacing+"----");
                spacing+=" ";
                System.out.print("\n"+spacing);
                for(int j=0;j<counter;j++){
                    System.out.print(remainder[j]);
                }
            }
        }
        else{
            remainder[counter]=frame[i];
            counter++;
            System.out.print("\n"+spacing);
            System.out.print("-");
            System.out.print("\n"+spacing+"0");
            System.out.print("\n"+spacing+"----");

            System.out.print("\n"+spacing);
            for(int j=0;j<counter;j++){
                System.out.print(remainder[j]);
            }

        }
    }

    System.out.print("\n\nRemainder is : ");
    boolean flag=true;
```

```
        for(int i=1;i<remainder.length;i++){
            if(remainder[i]==1){
                flag=false;
            }
            System.out.print(remainder[i]);
        }
        return flag;
    }
}
```

Output:

```
PS C:\Users\Admin\Documents\AIDS-B-75> cd "c:\Users\Admin\Documents\AIDS-B-75\" ; if ($?) { javac CRC.java } ; if ($?)
C }
Enter Frame Size: 10

Enter Frame Elements
Enter Frame 1:- 1
Enter Frame 2:- 1
Enter Frame 3:- 1
Enter Frame 4:- 1
Enter Frame 5:- 0
Enter Frame 6:- 1
Enter Frame 7:- 0
Enter Frame 8:- 1
Enter Frame 9:- 1
Enter Frame 10:- 0

Enter Generator Size:- 4
Enter Generator Elements
Enter Generator 1:- 1
Enter Generator 2:- 0
Enter Generator 3:- 1
Enter Generator 4:- 1

Sender Side:
Frame:-  1111010110
Generator:-  1011Number of 0's to append:- 3
Dividend(Modified data):- 1111010110000
```

```
1011_|1111010110000
      -
      0
      ----
      1
      -
      0
      ----
      11
      -
      0
      ----
      111
      -
      0
      ----
      1111
      -
      1011
      ----
      01000
      -
      1011
      ----
      00111
      -
      0
      ----
       1110
       -
       1011
       ----
       01011
       -
       1011
       ----
       00001
       -
       0
       ----
        0010
        -
        0
        ----
         0100
         -
         0
         ----
          1000
          -
          1011
          ----
          00110

Remainder is : 110
```

```
Remainder is : 110
Data after apending reaminder :-1111010110110
Date Sent to Reciever
```

```
Reciever Side


Data Recived:- 1111010110110
```

```
1011_|1111010110110
      -
      0
      ----
      1
      -
      0
      ----
      11
      -
      0
      ----
      111
      -
      0
      ----
      1111
      -
      1011
      ----
      01000
      -
      1011
      ----
      00111
      -
      0
      ----
       1110
       -
       1011
       ----
       01011
       -
       1011
       ----
       00001
       -
       0
       ----
        0010
        -
        0
        ----
         0101
         -
         0
         ----
          1011
          -
          1011
          ----
          00000
          -
          0
          ----
           0000

Remainder is : 000
Data is Correct
```