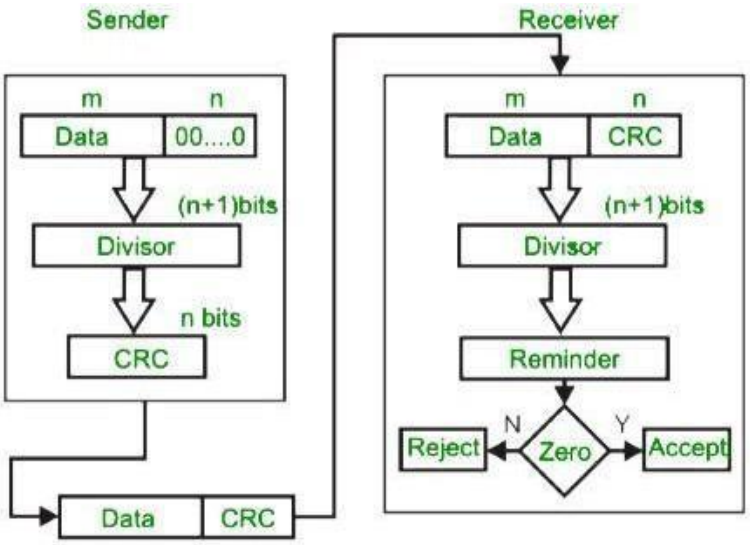
**Practical NO 6A :** CRC
**Name :** Shrikrushna C Gundre
**Roll No :** 12
**PRN :** 2122010191

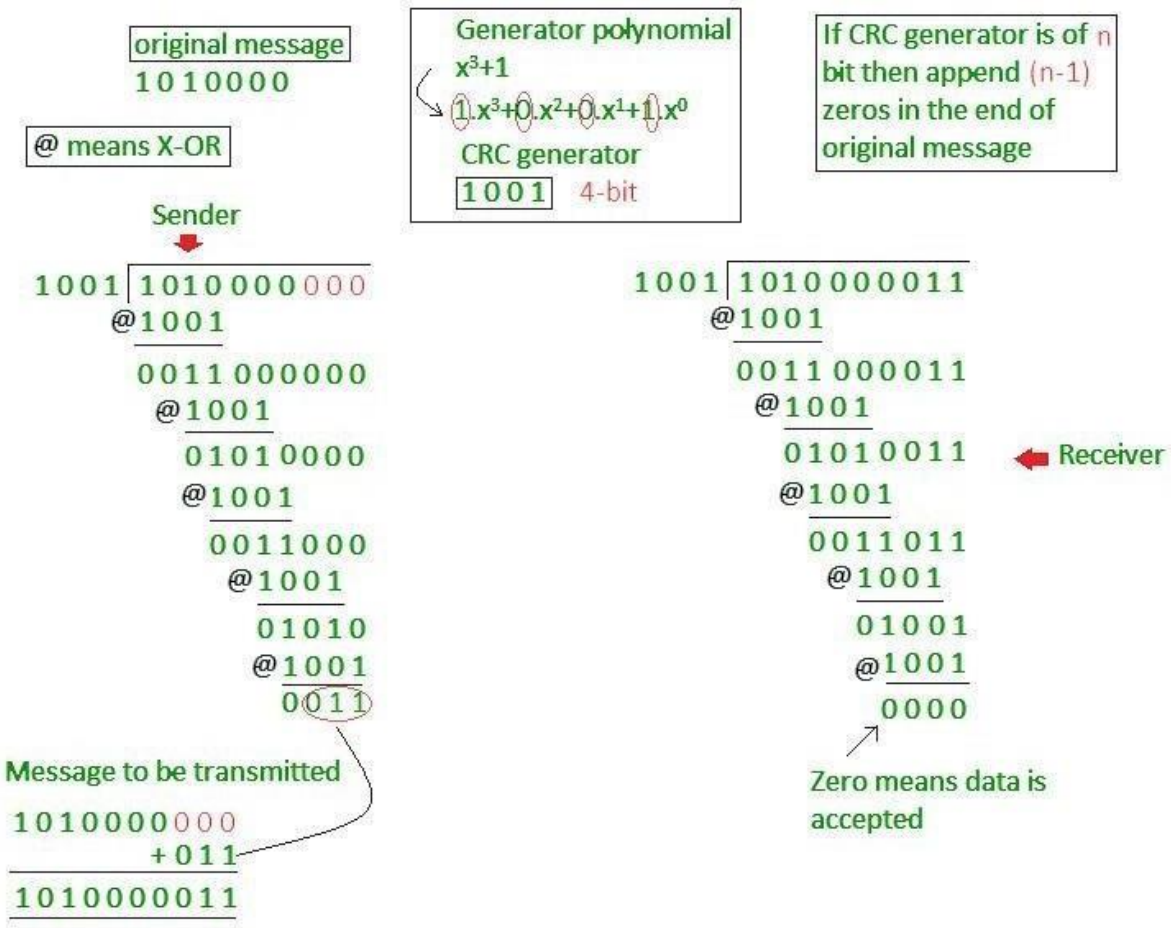
**CRC (Cyclic Redundancy Check)**

CRC is a different approach to detect if the received frame contains valid data. This technique involves binary division of the data bits being sent. The divisor is generated using polynomials. The sender performs a division operation on the bits being sent and calculates the remainder. Before sending the actual bits, the sender adds the remainder at the end of the actual bits. Actual data bits plus the remainder is called a code word. The sender transmits data bits as code words.

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.






**n:**

**Number of bits in data to be sent from sender side.**

**k: Number of bits in the key obtained from generator polynomial.**

**Advantages of CRC**

1. CRC can detect all the burst errors that affect an odd number of bits.
2. It is easy to implement.
3. Easy to view single bit errors and multiple burst errors. **Disadvantages of CRC**

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

**Code**

```
#include <stdio.h>
#include <stdlib.h>
void calculate(int);
```

```
int main() {     int
ch;     do
   {
       printf("\n1.Encode\n2.Decode\n3.Exit\nYour choice:");
scanf("%d", &ch);         switch (ch)
       {         case 1:
calculate(0);
break;         case 2:
calculate(1);
break;         default:
           printf("wrong choice/Exit");
       }
   } while (ch != 3);
return 0;
}

void calculate(int op)
{     int i, j, k = 0;     int flag = 1, a[16], gen[16], r[20],
div[16], n, m, ch;     printf("Enter Data in only 0 and
1 \n");     printf("Enter total no. of bits of generator :
");     scanf("%d", &n);     printf("\nEnter the
generator bit by bit : ");     n = n - 1;     for (i = 0; i <=
n; i++)
   {
       scanf("%d", &gen[i]);
   }     printf("\nEnter total no. of bits of data
: ");     scanf("%d", &m);     m = m - 1;
printf("Enter the data bit by bit : ");     for (i
= 0; i <= m; i++)
   {         scanf("%d",
&a[i]);
   }
```

```
   if (m < n || (gen[0] && gen[n]) == 0)
   {        printf("Not a proper generator
\n");        exit(0);
   }

   for (i = m + 1; i <= m + n; i++)
   {        a[i] = 0;     }
for (j = 0; j <= n; j++)
   {        r[j] =
a[j];
   }     for (i = n; i <= m + n;
i++)
   {
      if (i > n)        {
for (j = 0; j < n; j++)
{              r[j] = r[j + 1];
}            r[j] = a[i];        }
if (r[0])        {
div[k++] = 1;
      }        else
{            div[k++] =
0;            continue;
      }        for (j = 0; j <=
n; j++)
      {            r[j] = r[j] ^
gen[j];
      }
   }

   printf("\n\tQuotient : ");
for (j = 0; j < k; j++)     {
printf("%d ", div[j]);
```

```
    }
    printf("\n\tRemainder : ");
for (i = 1; i <= n; i++)     {
printf("%d ", r[i]);
    }

    if (op == 0)
    {         printf("\n\tTransmitted frame is : ");
for (i = m + 1, j = 1; i <= m + n; i++, j++)
        {
a[i] = r[j];
        }
        for (i = 0; i <= m + n; i++)
        {             printf("%d
", a[i]);
        }
printf("\n");     }
if (op == 1)
    {         for (i = 1; i <= n;
i++)
        {             if
(r[i])
flag = 0;
        }          if (flag)
printf("\n\tNo Error\n");
else             printf("\n\tError");
    }
}
```

**Output**

```
1.Encode
2.Decode
3.Exit
Your choice:1
Enter Data in only 0 and 1
Enter total no. of bits of generator : 4

Enter the generator bit by bit : 1 1 0 1

Enter total no. of bits of data : 6
Enter the data bit by bit : 1 0 0 1 0 0

        Quotient : 1 1 1 1 0 1
        Remainder : 0 0 1
        Transmitted frame  is : 1 0 0 1 0 0 0 0 1

1.Encode
2.Decode
3.Exit
Your choice:2
Enter Data in only 0 and 1
Enter total no. of bits of generator : 4

Enter the generator bit by bit : 1 1 0 1

Enter total no. of bits of data : 9
Enter the data bit by bit : 1 0 0 1 0 0 0 0 1

        Quotient : 1 1 1 1 0 1 0 0 0
        Remainder : 0 0 0
        No Error
```