[2CEIT503 COMPUTER NETWORKS]

Practical: 5

AIM- **Write a program to implement various Error Detection Mechanisms.**

a. find minimum hamming distance

b. Checksum

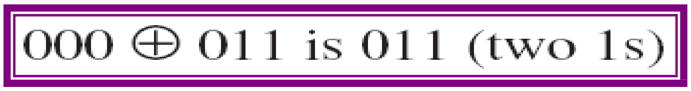
c. CRC

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
U.V. Patel College of Engineering

Department of Computer Engineering/Information Technology

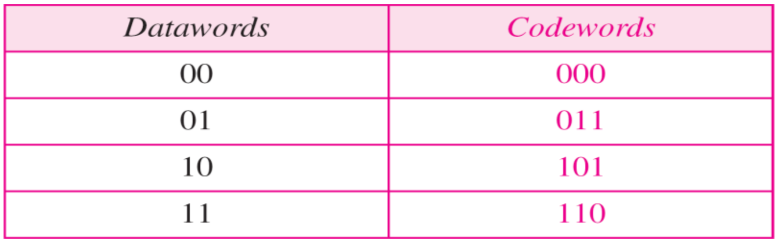
Hamming Distance

- The Hamming distance between two words is the number of differences between corresponding bits.
- Hamming distance between two words x and y as d(x,y)
- The Hamming distance d(000, 011) is 2 because

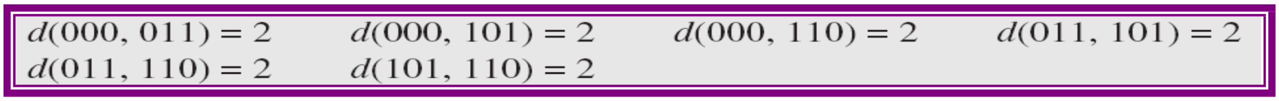


$000 \oplus 011$ is 011 (two 1s)

- The **minimum Hamming distance** is the smallest Hamming distance between all possible pairs in a set of words.



Datawords	*Codewords*
00	000
01	011
10	101
11	110



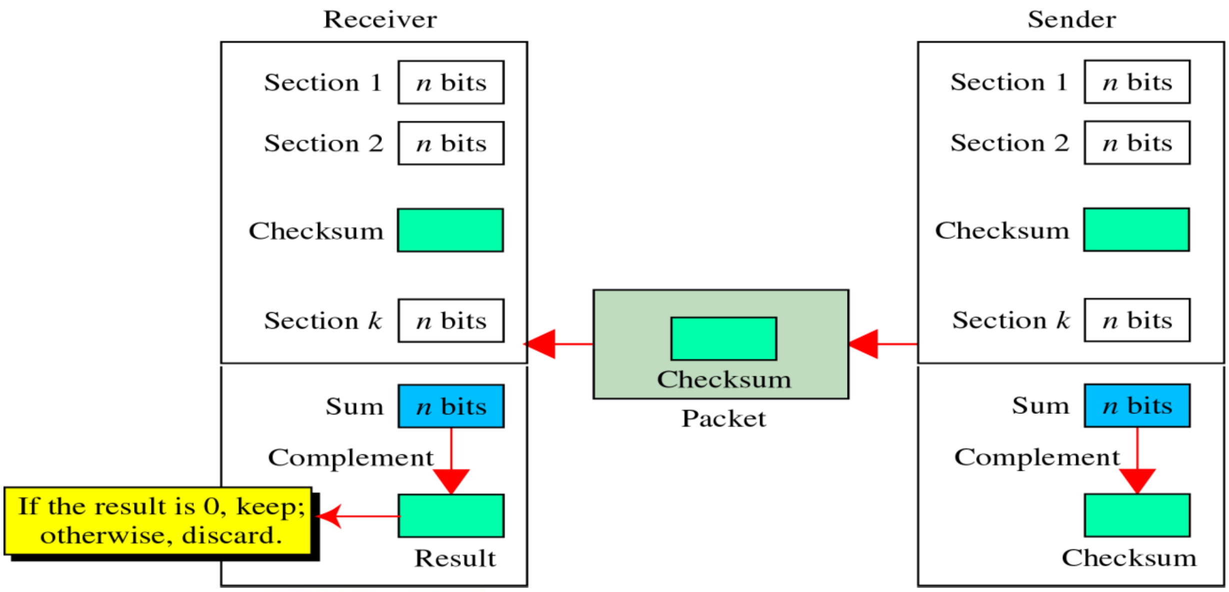
$d(000, 011) = 2$ $d(000, 101) = 2$ $d(000, 110) = 2$ $d(011, 101) = 2$

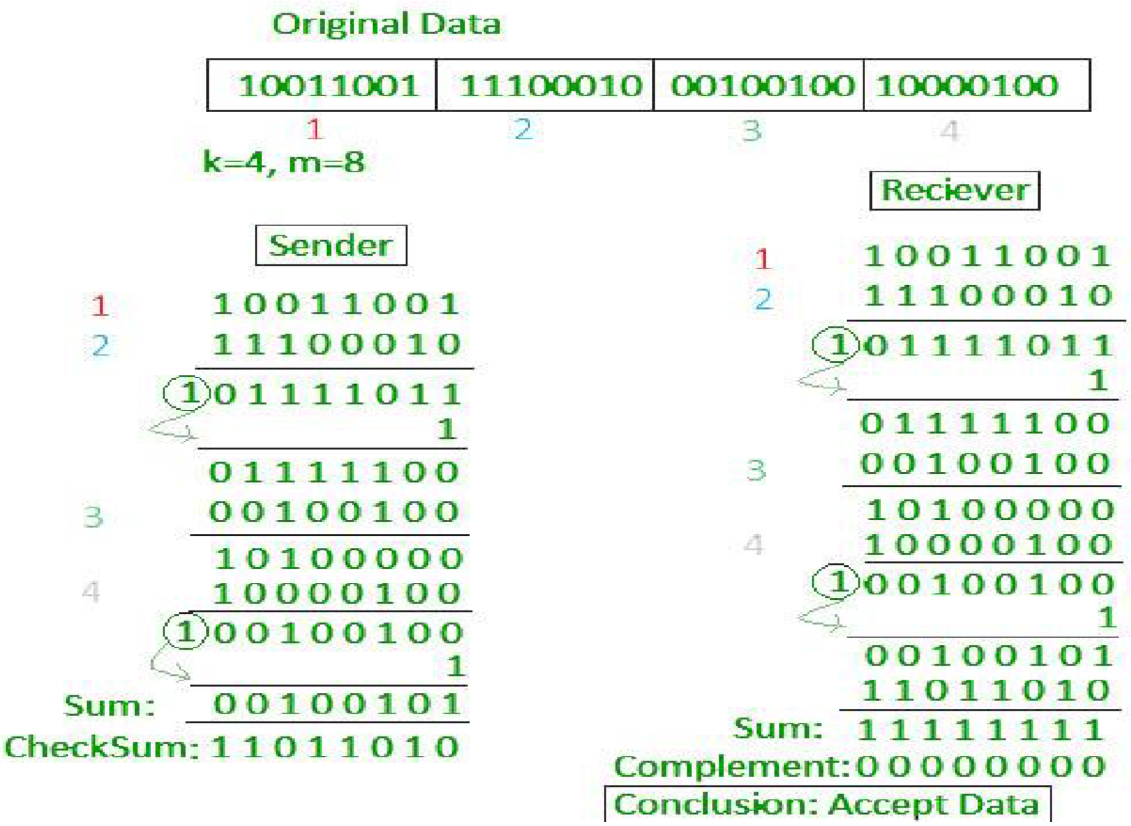
$d(011, 110) = 2$ $d(101, 110) = 2$

The d_{min} in this case is 2.

Checksum

- In checksum error detection scheme, the data is divided into *k* segments each of m bits.
- In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.





Original Data

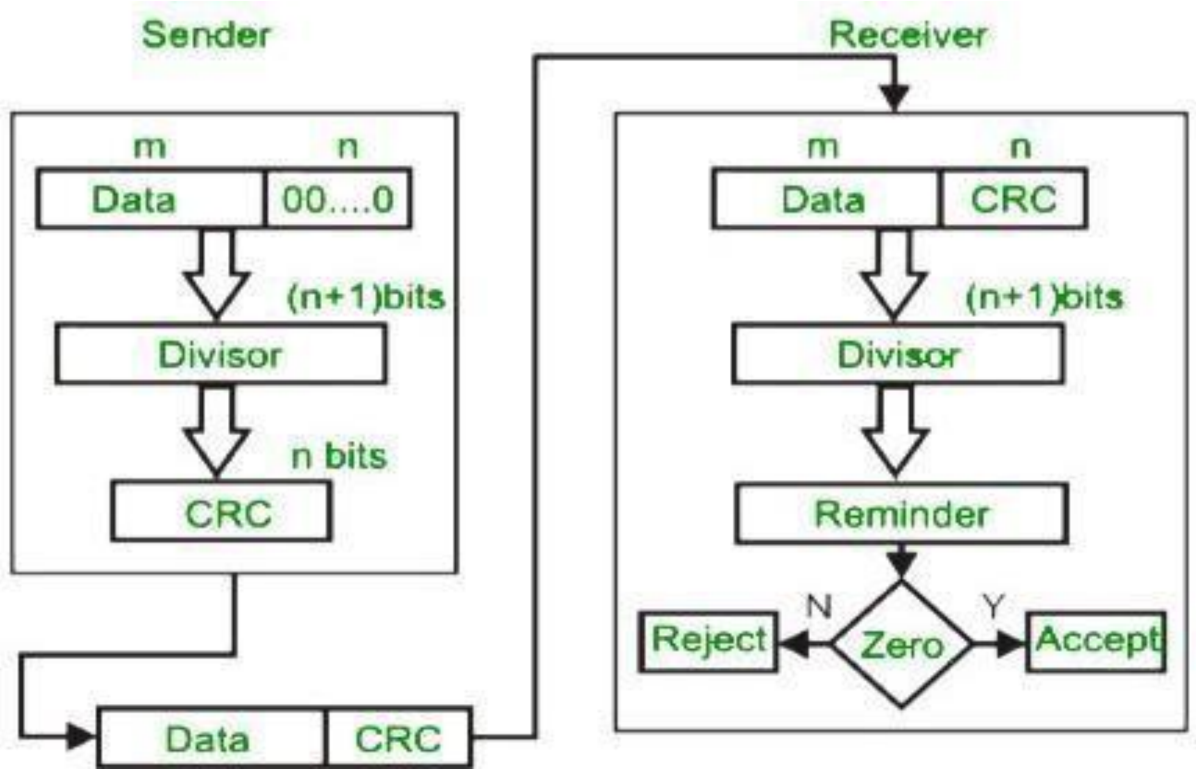
10011001	11100010	00100100	10000100
1	2	3	4

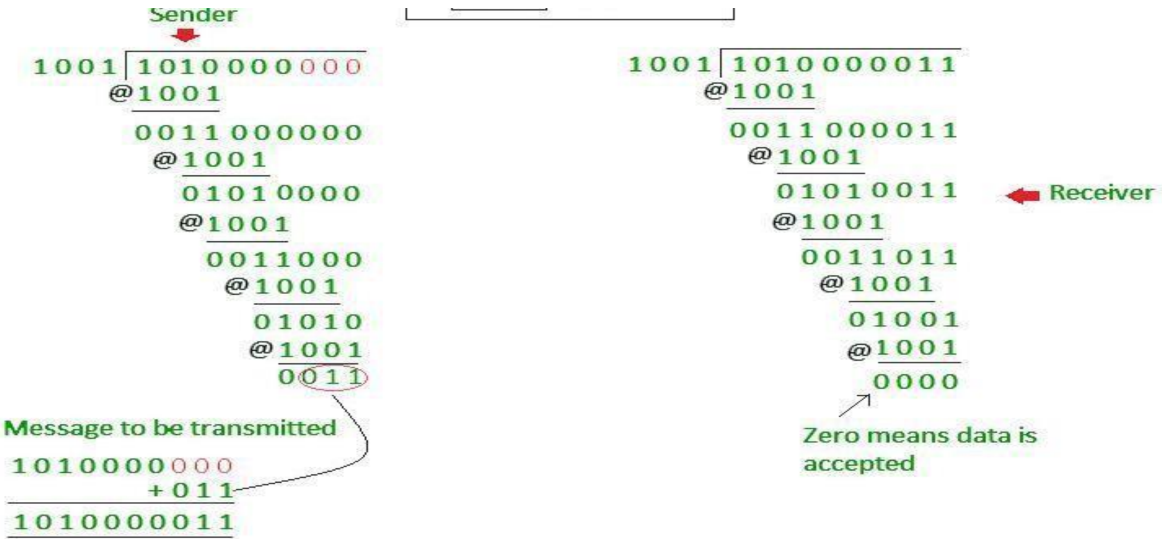
k=4, m=8

```
Sender                              Reciever

                                    1      10011001
                                    2      11100010
1          10011001                      (1)01111011
2          11100010                               1
         (1)01111011                       01111100
                   1                3      00100100
           01111100                        10100000
3          00100100                 4      10000100
           10100000                      (1)00100100
4          10000100                               1
         (1)00100100                       00100101
                   1                       11011010
Sum:       00100101                 Sum:   11111111
CheckSum:  11011010            Complement: 00000000
                               Conclusion: Accept Data
```

CRC

- Unlike checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.




Sender
1001 | 1010000000
@1001
0011000000
@1001
01010000
@1001
0011000
@1001
01010
@1001
0011
Message to be transmitted
1010000000
+011
1010000011
1001 | 1010000011
@1001
0011000011
@1001
01010011
@1001
0011011
@1001
01001
@1001
0000
Receiver
Zero means data is accepted

Aim: Write a program to implement various Error Detection Mechanisms.

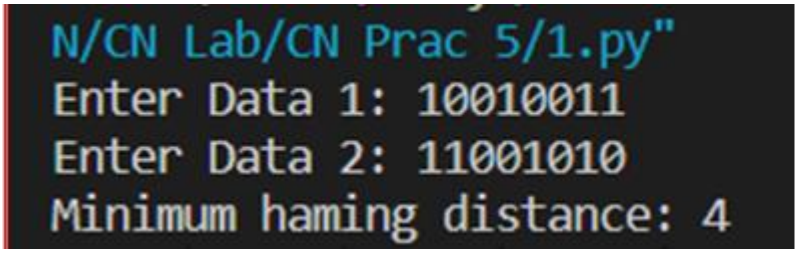
- **a. Find the Minimum Hamming Distance.**
- **b. Checksum**
- **c. CRC**

A. Find the Minimum Hamming Distance.

Program:

```
str1 = input("Enter Data 1: ")
str2 = input("Enter Data 2: ")

count = 0
for i in range(0, len(str1)):
    if str1[i] != str2[i]:
        count += 1

print("Minimum haming distance:", count)
```

Output:

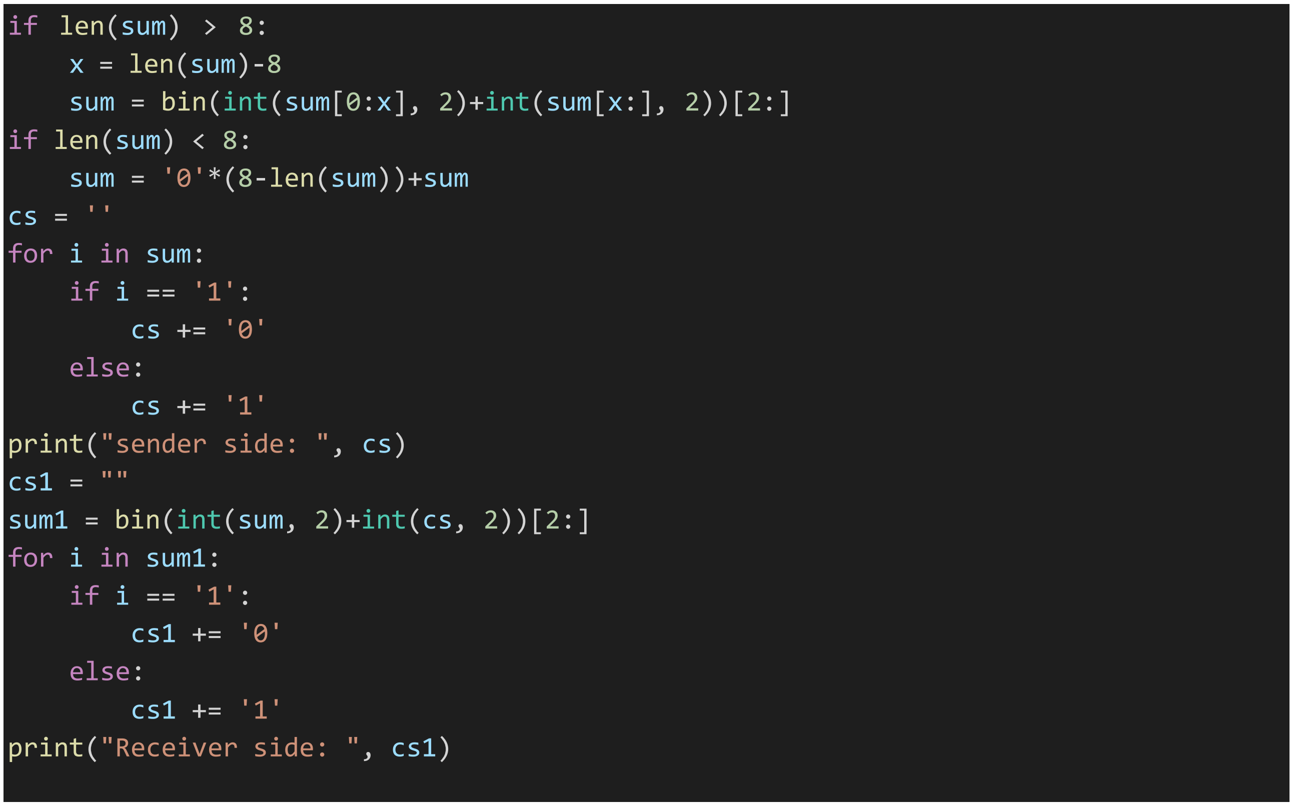


```
N/CN Lab/CN Prac 5/1.py"
Enter Data 1: 10010011
Enter Data 2: 11001010
Minimum haming distance: 4
```

B. Checksum

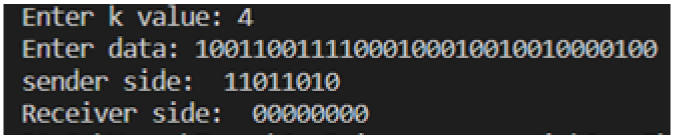
Program:

```
k = int(input("Enter k value: "))
str = input("Enter data: ")
s = len(str)/k
c1 = str[0:k*2]
c2 = str[8:k*4]
c3 = str[16:k*6]
c4 = str[24:k*8]
sum = bin(int(c1, 2)+int(c2, 2) + int(c3, 2)+int(c4, 2))[2:]
```



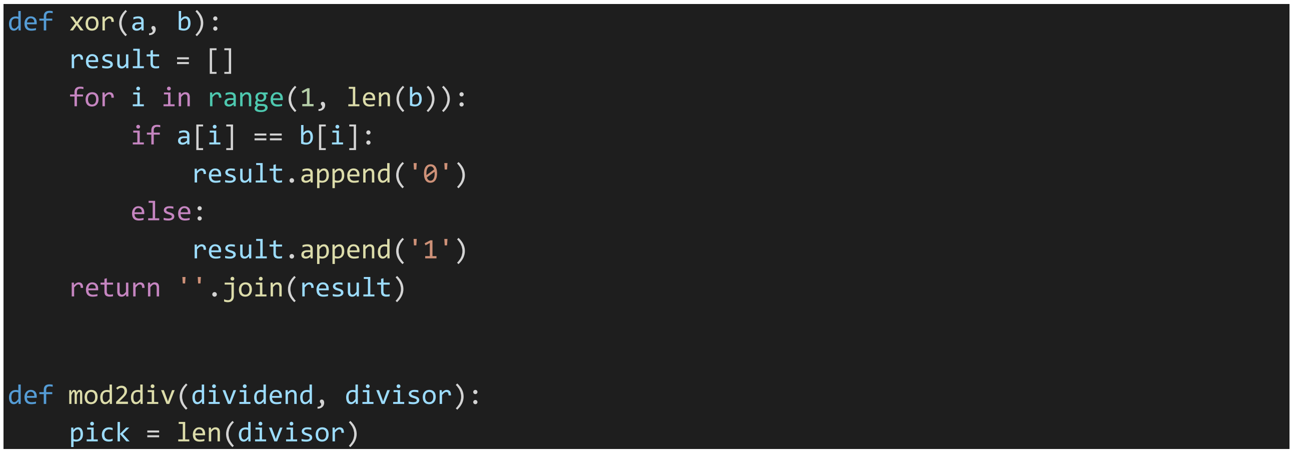
```
if  len(sum)  >  8:
    x = len(sum)-8
    sum = bin(int(sum[0:x], 2)+int(sum[x:], 2))[2:]
if len(sum) < 8:
    sum = '0'*(8-len(sum))+sum
cs = ''
for i in sum:
    if i == '1':
        cs += '0'
    else:
        cs += '1'
print("sender side: ", cs)
cs1 = ""
sum1 = bin(int(sum, 2)+int(cs, 2))[2:]
for i in sum1:
    if i == '1':
        cs1 += '0'
    else:
        cs1 += '1'
print("Receiver side: ", cs1)
```

Output:

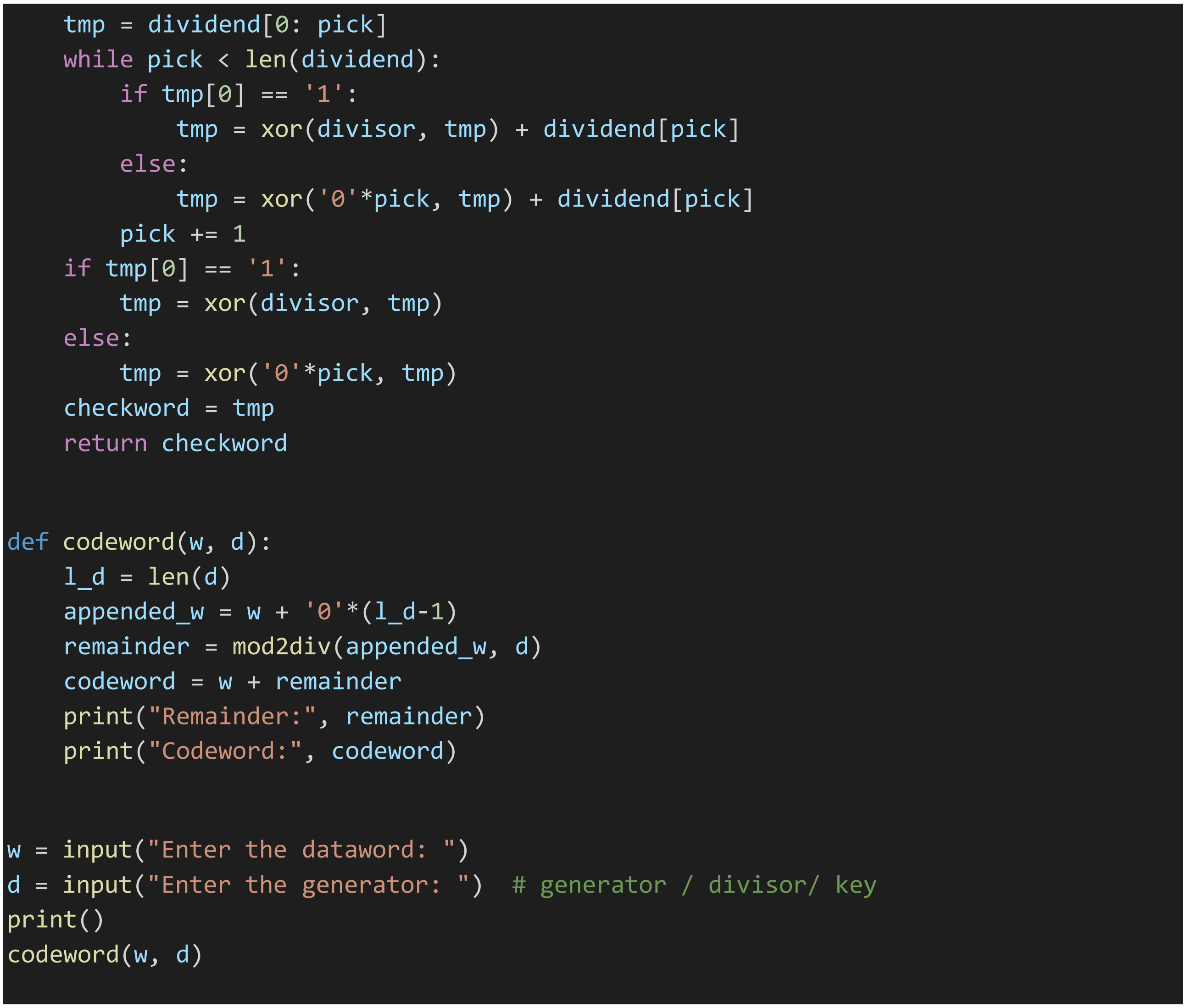


```
Enter k value: 4
Enter data: 10011001111000100010010010000100
sender side:  11011010
Receiver side:  00000000
```

C. CRC
Program:

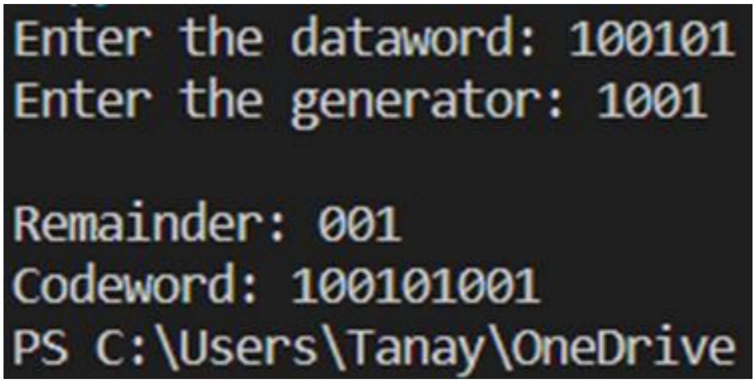


```
def xor(a, b):
    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)


def mod2div(dividend, divisor):
    pick = len(divisor)
```



```
    tmp = dividend[0: pick]
    while pick < len(dividend):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[pick]
        else:
            tmp = xor('0'*pick, tmp) + dividend[pick]
        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0'*pick, tmp)
    checkword = tmp
    return checkword


def codeword(w, d):
    l_d = len(d)
    appended_w = w + '0'*(l_d-1)
    remainder = mod2div(appended_w, d)
    codeword = w + remainder
    print("Remainder:", remainder)
    print("Codeword:", codeword)


w = input("Enter the dataword: ")
d = input("Enter the generator: ")  # generator / divisor/ key
print()
codeword(w, d)
```

Output:



```
Enter the dataword: 100101
Enter the generator: 1001

Remainder: 001
Codeword: 100101001
PS C:\Users\Tanay\OneDrive
```

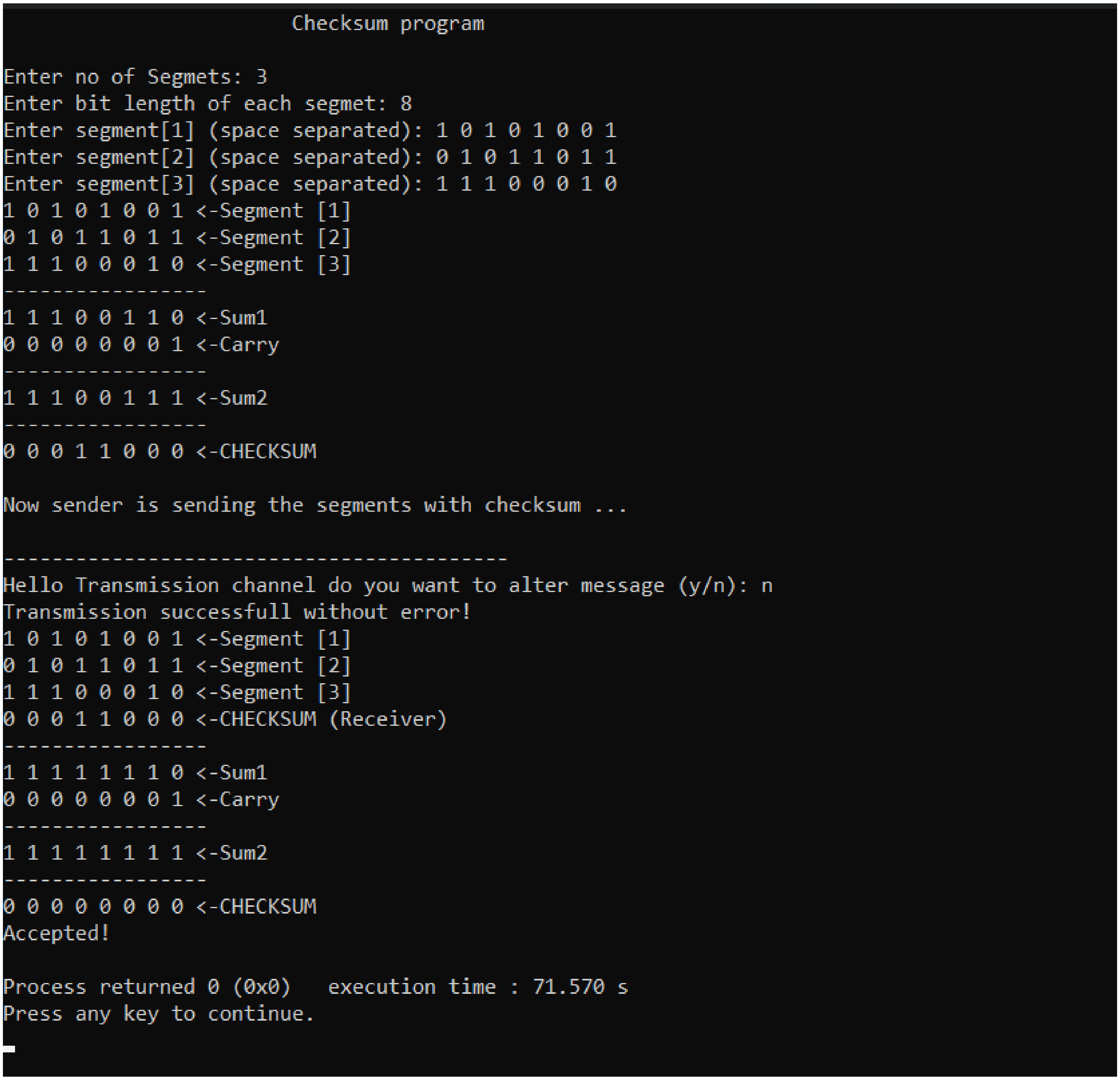
Checksum:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

int* decToBin(int dec,int n)
{
    int * bin=(int *)calloc(n,sizeof(int));
    for(int i=n-1; i>=0; i--)
    {
        bin[i]=dec%2;
        dec/=2;
    }
    return bin;
}
void checksumGen(int **data,int n,int k)
{
    int  carrynum=0;
    for(int j=k-1; j>=0; j--)
    {
        int sum=0;
        for(int i=0; i<n; i++)
        {
            sum+=data[i][j];
        }
        sum+=carrynum;
        data[n][j]=sum%2;
        carrynum=sum>0 ? sum>>1 : 0;
    }
    int *carry=decToBin(carrynum,k);
    for(int i=0; i<n; i++)
    {
        for(int j=0; j<k; j++)
        {
            printf("%d ",data[i][j]);
        }
        printf("<-Segment [%d] \n",(i+1));
    }
    printf("_________________________\n");
    for(int j=0; j<k; j++)
        printf("%d ",data[n][j]);
    printf("<-Sum1\n");
    carrynum=0;
    for(int i=k-1; i>=0; i--)
    {
        int sum=carrynum+carry[i]+data[n][i];
```

```
            data[n][i]=sum%2;
            carrynum=sum>0 ? sum>>1 : 0;
        }
        for(int i=0; i<k; i++)
            printf("%d ",carry[i]);
        printf("<-Carry\n");
        printf("______________________\n");
        for(int i=0; i<k; i++)
        {
            printf("%d ",data[n][i]);
            data[n][i]= data[n][i]==0 ? 1 : 0;
        }
        printf("<-Sum2\n");
        printf("______________________\n");
        for(int i=0; i<k; i++)
        {
            printf("%d ",data[n][i]);
        }
        printf("<-CHECKSUM\n");
    }
    void checksumChk(int **data,int n,int k)
    {
        int *chkBucket=(int *)calloc(k,sizeof(int));
        int carrynum=0;
        for(int j=k-1; j>=0; j--)
        {
            int sum=0;
            for(int i=0; i<=n; i++)
            {
                sum+=data[i][j];
            }
            sum+=carrynum;
            chkBucket[j]=sum%2;
            carrynum=sum>0 ? sum>>1 : 0;
        }
        int *carry=decToBin(carrynum,k);
        for(int i=0; i<n; i++)
        {

            for(int j=0; j<k; j++)
            {
                printf("%d ",data[i][j]);
            }
            printf("<-Segment [%d] \n",(i+1));
        }
        for(int i=0; i<k; i++)
        {
```

```
        printf("%d ",data[n][i]);
    }
    printf("<-CHECKSUM (Receiver)\n");
    printf("_________________________\n");
    for(int j=0; j<k; j++)
        printf("%d ",chkBucket[j]);
    printf("<-Sum1\n");
    carrynum=0;
    for(int i=k-1; i>=0; i--)
    {
        int sum=carrynum+carry[i]+chkBucket[i];
        chkBucket[i]=sum%2;
        carrynum=sum>0 ? sum>>1 : 0;
    }
    for(int i=0; i<k; i++)
        printf("%d ",carry[i]);
    printf("<-Carry\n");
    printf("_________________________\n");
    for(int i=0; i<k; i++)
    {
        printf("%d ",chkBucket[i]);
        chkBucket[i]=chkBucket[i]==0 ? 1 : 0;
    }
    printf("<-Sum2\n");
    printf("_________________________\n");
    bool accept=true;
    for(int i=0; i<k; i++)
    {
        printf("%d ",chkBucket[i]);
        if(chkBucket[i]!=0)
            accept=false;
    }
    printf("<-CHECKSUM\n");
    printf("%s",(accept ? "Accepted!" : "Rejected!"));
    printf("\n");
}
int main()
{
    printf("\t\t\tChecksum program\n\n");
    int n,k;
    printf("Enter no of Segmets: ");
    scanf("%d",&n);
    printf("Enter bit length of each segmet: ");
    scanf("%d",&k);
    int len=(n+1)*sizeof(int *)+ (n+1)*(k)*sizeof(int);
    int **data=(int **)malloc(len);
    int * ptr=(int *)(data+n+1);
    for(int i=0; i<n+1; i++)
        data[i]=(ptr+k*i);
    for(int i=0; i<n; i++)
    {
        printf("Enter segment[%d] (space separated): ",(i+1));
        for(int j=0; j<k; j++)
        {
            scanf("%d",&data[i][j]);
        }
    }
    checksumGen(data,n,k);
    printf("\nNow sender is sending the segments with checksum ... \n\n");
    printf("------------------------------------------------------------\n");
    printf("Hello Transmission channel do you want to alter message (y/n): ");
    char choice;
    scanf(" %c",&choice);
    switch(choice)
    {
    case 'y':
        printf("You are supposed to enter %d segments of %d length!\n",n,k);
        for(int i=0; i<n; i++)
        {
            printf("Enter segment[%d] (space separated): ",(i+1));
            for(int j=0; j<k; j++)
            {
                scanf("%d",&data[i][j]);
            }
        }
        printf("Enter the checksum (space separated): ");
        for(int j=0; j<k; j++)
            scanf("%d",&data[n][j]);
        break;
    case 'n':
        printf("Transmission successfull without error!\n");
        break;
    }
    checksumChk(data,n,k);
}
```

Output:



```
                        Checksum program

Enter no of Segmets: 3
Enter bit length of each segmet: 8
Enter segment[1] (space separated): 1 0 1 0 1 0 0 1
Enter segment[2] (space separated): 0 1 0 1 1 0 1 1
Enter segment[3] (space separated): 1 1 1 0 0 0 1 0
1 0 1 0 1 0 0 1 <-Segment [1]
0 1 0 1 1 0 1 1 <-Segment [2]
1 1 1 0 0 0 1 0 <-Segment [3]
-----------------
1 1 1 0 0 1 1 0 <-Sum1
0 0 0 0 0 0 0 1 <-Carry
-----------------
1 1 1 0 0 1 1 1 <-Sum2
-----------------
0 0 0 1 1 0 0 0 <-CHECKSUM

Now sender is sending the segments with checksum ...

------------------------------------------
Hello Transmission channel do you want to alter message (y/n): n
Transmission successfull without error!
1 0 1 0 1 0 0 1 <-Segment [1]
0 1 0 1 1 0 1 1 <-Segment [2]
1 1 1 0 0 0 1 0 <-Segment [3]
0 0 0 1 1 0 0 0 <-CHECKSUM (Receiver)
-----------------
1 1 1 1 1 1 1 0 <-Sum1
0 0 0 0 0 0 0 1 <-Carry
-----------------
1 1 1 1 1 1 1 1 <-Sum2
-----------------
0 0 0 0 0 0 0 0 <-CHECKSUM
Accepted!

Process returned 0 (0x0)   execution time : 71.570 s
Press any key to continue.
```