Implement the data link layer framing methods.

a. Character Count:

Program:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str[20];
    int n,i,j, count=1, v[20];
    clrscr();
    printf("\n Enter the string: ");
    scanf("%s", str);
    n= strlen(str);
    printf("\n The length of the string: %d", n);
    for(i=0; i<n; i++)
    {
        count=1;
        for(j=i+1; j<n; j++)
        {
```

Output:

Enter the string: malayalam

The length of the string: 9

Number of m's : 2

Number of a's : 4

Number of l's : 2

Number of y's : 1

Expt. No..... Sheet No. 2 Date .....

```
        if(str[i]==str[j])
        {
            v[j]=1;
            count++;
        }
    }
    if(v[i]!=1)
    {
        printf("\n Number of %c's: %d",
                str[i], count);
    }
}
}
```

b. Bit Stuffing:

Theory:

Security and Error detection are the most prominent features that are to be provided by any application which transfers data from one end to the other end. One of a such mechanism in tracking errors which may add up to the original data during transfer is known as Stuffing. It is of two types namely Bit stuffing and the other Character stuffing. Coming to the Bit Stuffing. 01111110 is appended within the original data while transfer of it. The following program describes how it is stuffed at the sender end and de-stuffed at the receiver end.

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[15];
    int i,j,k,n,c=0,pos=0;
```

```
clrscr();
printf("\nEnter the number of bits: ");
scanf("%d", &n);
printf("\nEnter the bits: ");
for(i=0;i<n;i++)
{
    scanf("%d", &a[i]);
}
for(i=0;i<n;i++)
{
    if(a[i]==1)
    {
        c++;
        if(c==5)
        {
            pos=i+1;
            c=0;
            for(j=n; j>=pos; j--)
            {
                k=j+1;
                a[k]=a[j];
            }
            a[pos]=0;
```

Output:

Enter the number of bits: 10

Enter the bits: 0 1 1 1 1 1 1 0 1 1

DATA AFTER STUFFING:

01111110 0111110 1011 01111110

```
            n=n+1;
        }
    }
    else
    {
        c=0;
    }
}
printf("\n DATA AFTER STUFFING:\n");
printf(" 0!!!!!!0 ");
for(i=0;i<n;i++)
{
    printf("%d", a[i]);
}
printf("0!!!!!!0");
}
```

c. Character Stuffing and De-stuffing:

Theory:

Coming to the character stuffing. DLESTX and DLEETX are used to denote start and end of character data with some constraints imposed on repetition of characters as shown in the program below clearly.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void chare();
void main()
{
    int choice;
    clrscr();
    while(1)
    {
        printf("\nMENU:");
        printf("\n1. CHARACTER STUFFING.");
        printf("\n2. EXIT.");
```

```
            printf("\nEnter the choice: ");
            scanf("%d", &choice);
            if (choice > 2)
            printf("\nALERT: Invalid Option..... Please
                    re-enter the choice.");
            switch(choice)
            {
                case 1: charc();
                        break;
                case 2: exit(0);
            }
        }
}
void charc()
{
        char c[50], d[50], t[50];
        int m,i,j;
        printf("\nEnter the number of characters: ");
        scanf("%d", &m);
        printf("\nEnter the characters: ");
```

```
for(i=0;i<m+1;i++)
{
    scanf("%d", &c[i]);
}
printf("\n Original Data: ");
for(i=0;i<m+1;i++)
{
    printf("%c", c[i]);
}
d[0]='d';
d[1]='l';
d[2]='e';
d[3]='s';
d[4]='t';
d[5]='x';
for(i=0;j=6;i<m+1;i++,j++)
{
    d[j]=c[i];
}
m=m+6;
d[++m]='d';
d[++m]='l';
d[++m]='e';
```

Output:

MENU:

1. CHARACTER STUFFING.

2. EXIT.

Enter the choice: 1

Enter the number of characters: 8

Enter the characters: dleleabc

Original Data:

dleleabc

Transmitted Data:

dlestx

dleleabcdleetx

Received Data:

dleleabc

MENU:

1. CHARACTER STUFFING.

2. EXIT.

Enter the choice: 2

```
d[++m] = 'e';
d[++m] = 't';
d[++m] = 'x';
m++;
printf("\nTransmitted Data:\n");
for(i=0; i<m; i++)
{
    printf("%c", d[i]);
}
for(i=6, j=0; i<m-6; i++, j++)
{
    t[j] = d[i];
}
printf("\nReceived Data: ");
for(i=0; i<j; i++)
{
    printf("%c", t[i]);
}
}
```

Implement on a data set of characters the Cyclic Redundancy check (CRC) polynomials.

Theory:

CRC means Cyclic Redundancy Check. It is the most famous and tradionally successful mechanism used in error detection through the parity bits installed within the data and obtaining checksum which acts as the veritier to check whether the data retrieved at the reciever end is genuine or not. Various operations are involved in implementing CRC on a data set through CRC generating polynomials

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    int i,j,keylen,msglen;
    char input[100], key[30], temp[30], quot[100], rem[30],
         key1[80];
```

```
clrscr();
printf("Enter Data:");
gets(input);
printf("Enter Key:");
gets(key);
keylen= strlen(key);
msglen = strlen(input);
strcpy(key1, key);
for(i=0;i<keylen-1;i++)
{
    input[msglen+i]='0';
}
for(i=0; i<keylen;i++)
  temp[i]= input[i];
for(i=0; i<msglen;i++)
{
    quot[i]=temp[0];
    if(quot[i]==='0')
      for(j=0; j<keylen;j++)
        key[j]='0';
```

```
else
    for(j=0;j<keylen;j++)
        key[j]= key1[j]
    for(j=keylen-1;j>0;j--)
    {
        if(temp[j]==key[j])
            rem[j-1]='0';
        else
            rem[j-1]='1';
    }
    rem[keylen-1]=input[i+keylen];
    strcpy(temp,rem);
}
strcpy(rem,temp);
printf("\n Quotient: ");
for(i=0;i<msglen;i++)
    printf("%c",quot[i]);
```

Output:

Enter Data: 1101101

Enter key: 101

Quotient: 1110001

Remainder: 01

Final Data: 110110101

Expt. No..... Sheet No.....13..... Date .....

```
printf("\n Remainder: ");
for (i=0;i<keylen-1 ;i++)
    printf("%c", rem[i]);
printf("\n Final Data: ");
for(i=0; i<msglen;i++)
    printf("%c", input[i]);
for(i=0;i<keylen-1; i++)
    printf("%c", rem[i]);
}
```