1a. Design a C program in which sender module should count the no of bytes in the frame and receive module should display each frame received.

```
#include <stdio.h>
#include <string.h>
void reciever();
char frames[1024];
int main()
{
      int n, len, i;
      char buffer[256], length[10];
      printf("How many frames you want to send: ");
      bzero(buffer, 256);
      scanf("%d", &n);
      for (i = 0; i < n; i++)
      {
            printf("Enter frame\n");
            scanf("%s", buffer);
            printf("String length of buffer is %d\n", strlen(buffer));
            len = strlen(buffer);
            len = len + 1;
            sprintf(length, "%d", len);
            strcat(frames, length);
            strcat(frames, buffer);
      }
      for (i = 0; frames[i] != '\0'; i++)
            printf("%c", frames[i]);
      reciever();
      return 0;
}
void reciever()
{
      int i = 0, framelen, lpvar;
      char leninchar;
      printf("\n\nThis is the reciever\n");
      printf("\nData recieved is %s", frames);
      while (frames[i] != '\0')
      {
            leninchar = frames[i];
            framelen = (int)leninchar - (int)'0';
            printf("\nLength of this frame is %d\n", framelen);
            printf("\nFrame ----->");
            lpvar = i + framelen;
            i = i + 1;
            while (i < lpvar)
            {
                  printf("%c", frames[i++]);
            }
            printf("\n");
      }
}
```

1b. Design a C program to implement bit stuffing, encoding and decoding concept in data link layer.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void sender();
void receiver(int *message,int l2);
int main(void)
{
      sender();
}
void sender()
{
      int i,j,n,count=0,zerocounter=0,zero=0;
      int msg[50];
      int result[50];
      printf("Enter the number of bits of the message\n");
      scanf("%d",&n);
      printf("Enter the bits\n");
      for(i=0;i<n;i++)
      {
            scanf("%d",&msg[i]);
      }
      result[0]=0;
      result[1]=1;
      result[2]=1;
      result[3]=1;
      result[4]=1;
      result[5]=1;
      result[6]=1;
      result[7]=0;
      j=8;
      for(i=0;i<n;i++)
      {
            if(msg[i]==0)
            {
                  result[j]=msg[i];
                  j++;
                  zero=1;
                  count=0;
            }
            else
            {
                  if((count==5)&&(zero==1))
                  {
                        result[j]=0;
                        zerocounter++;
                        j++;
                        result[j]=msg[i];
                        j++;
                        count=0;
                  }
                  else
                  {
                        result[j]=msg[i];
                        j++;
                        count++;
                  }
            }
      }
      result[j++]=0;
      result[j++]=1;
      result[j++]=1;
      result[j++]=1;
      result[j++]=1;
      result[j++]=1;
      result[j++]=1;
      result[j++]=0;
      int l1=16+n+zerocounter;
      printf("The length is: %d\n",l1);
      printf("The frame is\n");
      for(i=0;i<j;i++)
      {
            printf("%d",result[i]);
      }
      receiver(result,l1);
}
void receiver(int *result,int l2)
{
      int i,j,counter,l3;
      int mesg[100];
      l3=l2-8;
      j=0;
      for(i=8;i<l3;i++)
      {
            if(result[i]==0)
            {
                  if(counter==5)
                  {
                        i++;
                        mesg[j]=result[i];
                        j++;
                        counter=0;
                  }
                  else
                  {
                        mesg[j]=result[i];
                        j++;
                        counter=0;
                  }
            }
            else
            {
                  mesg[j]=result[i];
                  j++;
                  counter++;
            }
      }
      printf("\nReciever side message is:");
      for(i=0;i<j;i++)
      {
            printf("%d",mesg[i]);
      }
}
```

2. Design and implement CRC error detection method used in data link layer.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#define N strlen(g)
char t[120],cs[120],g[]="100000111";
int a,c,e;
void xor()
{
 for(c=1;c<N;c++)
 cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc()

{
for(e=0;e<N;e++)
 cs[e]=t[e];
 do
  {
   if(cs[0]=='1')
    xor();
   for(c=0;c<N-1;c++)
cs[c]=cs[c+1];
cs[c]=t[e++];
}
while(e<=a+N-1);
}
void main()
{
printf("enter the polynomial\n");
scanf("%s",t);
printf("generating polynomial is %s\n",g);
a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("modified t[u] is %s\n",t);
crc();
printf("checksum is :%s\n",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];
printf("final codeword is :%s\n",t);
printf("test error detection 0(yes)1(no)?:\n");
scanf("%d",&e);
if(e==0)
 {
 do
  {
   printf("enter position where error has to be inserted\n");
   scanf("%d",&e);
  }
while(e==0 || e>a+N-1);
 t[e-1]=(t[e-1]=='0')?'1':'0';
 printf("errorneous data %s\n",t);
}

crc();

for(e=0;(e<N-1)&&(cs[e]!='1');e++);
 if(e<N-1)
  printf("error detected\n");
 else
  printf("error is not detected\n");
}
```

3a. Design a C program to implement client server model (TCP) using socket programming.

Client

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<netinet/in.h>
#include<errno.h>
#include<string.h>
int main()
{
      int sock,bytes_recv;
      struct sockaddr_in server_addr;
      char recv_data[1024],send_data[1024];
      struct hostent *host;
      host=gethostbyname("127.0.0.1");
      if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)
      {
            perror("socket");
            exit(1);
      }
      server_addr.sin_family=AF_INET;
      server_addr.sin_port=htons(6119);
      server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
      if(connect(sock,(struct sockaddr *)&server_addr,sizeof(struct
sockaddr))==-1)
      {
            perror("connect");
            exit(1);
      }
            printf("send Filename to send\n");
            gets(send_data);

            if(strcmp(send_data,"q")!=0)
                  send(sock,send_data,strlen(send_data),0);

            while((bytes_recv=recv(sock,recv_data,1024,0))>0)
            {
                  recv_data[bytes_recv]='\0';
                  //printf("%s\n\n", recv_data);
                  //if(strcmp(recv_data,"q")==0)
            //    {
            //    close(sock);
            //    break;
            //    }
                  printf("%s\n", recv_data);
            }
      close(sock);
      return 0;
}
```

Server

```
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<errno.h>
#include<unistd.h>
#include<netinet/in.h>
#include<string.h>
int main()
{
      struct sockaddr_in server_addr;
      struct sockaddr_in client_addr;
      FILE *fptr;
      int sock,connected,bytes_recv;
      char ch,send_data[1024],recv_data[1024];
      int sin_size,flag = 0;


      if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)
      {
            perror("socket");
            exit(1);
      }


      server_addr.sin_family=AF_INET;
      server_addr.sin_port=htons(6119);
      server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

      if(bind(sock,(struct sockaddr *)&server_addr, sizeof(struct sockaddr))==-
1)
      {
            perror("unable to bind");
            exit(1);
      }

      if(listen(sock,5)==-1)
      {
            perror("lsten");
            exit(1);
      }

      printf("tcp server is waiting for client on port XXXX\n");
      sin_size=sizeof(struct sockaddr_in);
      connected=accept(sock,(struct sockaddr *)&client_addr,&sin_size);

      while(1)
      {

            bytes_recv=recv(connected,recv_data,1024,0);
            recv_data[bytes_recv]='\0';

            printf("reciecved data is %s\n\n\n",recv_data);


            fptr=fopen(recv_data,"r");
            if(fptr==NULL)
            {
                  strcpy(send_data,"FILE");
                  send(connected,send_data,strlen(send_data),0);
            }
            ch = fgetc(fptr);
            while(ch != EOF)//this loop searches the for the current word
            {
               // fscanf(fptr,"%s",send_data);
                  send_data[flag] = ch;
                  flag++;
                  ch = fgetc(fptr);
               //send(connected,send_data,strlen(send_data),0);
            }
                  send(connected,send_data,strlen(send_data),0);
                  //send_data[0] = 'q';
                  //strcpy(send_data,"q");
                  //send(connected,send_data,strlen(send_data),0);
                  close(connected);
                  break;
      }
}
```

3b. Design a C program to implement client server model (UDP) using socket programming.

Client

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main(){
  int clientSocket, portNum, nBytes;
  char buffer[1024];
  struct sockaddr_in serverAddr;
  socklen_t addr_size;

  /*Create UDP socket*/
  clientSocket = socket(PF_INET, SOCK_DGRAM, 0);

  /*Configure settings in address struct*/
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_port = htons(8893);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*Initialize size variable to be used later on*/
  addr_size = sizeof serverAddr;

  while(1){
    printf("Type a sentence to send to server:\n");
    fgets(buffer,1024,stdin);
    printf("You typed: %s",buffer);

    nBytes = strlen(buffer) + 1;

    /*Send message to server*/
    sendto(clientSocket,buffer,nBytes,0,(struct sockaddr
*)&serverAddr,addr_size);

    /*Receive message from server*/
                nBytes = recvfrom(clientSocket,buffer,1024,0,NULL, NULL);

    printf("Received from server: %s\n",buffer);

  }

  return 0;
}
```

Server

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <stdlib.h>

int main(){
  int udpSocket, nBytes;
  char buffer[1024];
  struct sockaddr_in serverAddr, clientAddr;
  struct sockaddr_storage serverStorage;
  socklen_t addr_size, client_addr_size;
  int i;

  /*Create UDP socket*/
  udpSocket = socket(PF_INET, SOCK_DGRAM, 0);

  /*Configure settings in address struct*/
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_port = htons(8893);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*Bind socket with address struct*/
  bind(udpSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));

  /*Initialize size variable to be used later on*/
  addr_size = sizeof serverStorage;

  while(1){
    /* Try to receive any incoming UDP datagram. Address and port of
 *       requesting client will be stored on serverStorage variable */
    nBytes = recvfrom(udpSocket,buffer,1024,0,(struct sockaddr *)&serverStorage,
&addr_size);

    /*Convert message received to uppercase*/
    for(i=0;i<nBytes-1;i++)
      buffer[i] = toupper(buffer[i]);

    /*Send uppercase message back to client, using serverStorage as the
address*/
    sendto(udpSocket,buffer,nBytes,0,(struct sockaddr
*)&serverStorage,addr_size);
  }

  return 0;
}
```

4. Design and implement a C program to route the packet in a network using distance vector algorithm.

```
#include<stdio.h>
struct node
{
      unsigned dist[20];
      unsigned from[20];
}rt[10];
int main()
{
int dmat[20][20];
int n,i,j,k,count=0;
printf("\nEnter the number of nodes: ");
scanf("%d",&n);
printf("\nEnter the cost matrix\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
      scanf("%d",&dmat[i][j]);
      dmat[i][i]=0;
      rt[i].dist[j]=dmat[i][j];
      rt[i].from[j]=j;
}
do
{
count=0;
for(i=0;i<n;i++)
{
      for(j=0;j<n;j++)
      {
            for(k=0;k<n;k++)
            {
                  if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
                  {
                        rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                        rt[i].from[j]=k;
                        count++;
                  }
            }
      }
}
}while(count!=0);
for(i=0;i<n;i++)
{
      printf("\n\nState value for router %d is \n",i+1);
      printf("\nNode \t Via \t Dist. ");
      for(j=0;j<n;j++)
      {
            printf("\n%d \t %d \t %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
      }
}
printf("\n\n");
}
```

5. Design a C program for congestion control using leaky bucket algorithm.

```
#include<stdio.h>
#include<stdlib.h>
#define MIN(x,y) (x>y)?y:x
int main()
{
      int orate,drop=0,cap,x,count=0,inp[10]={0},i=0,nsec,ch;
      printf("\n enter bucket size : ");
      scanf("%d",&cap);
      printf("\n enter output rate :");
      scanf("%d",&orate);
      do{
      printf("\n enter number of packets coming at second %d :",i+1);
      scanf("%d",&inp[i]);
      if(inp[i]>cap)
      {
            printf("Bucket overflow\n");
            printf("Packet Discarded\n");
            exit(0);
      }
      i++;
      printf("\n enter 1 to contiue or 0 to quit.........");
      scanf("%d",&ch);
}
while(ch);
nsec=i;
printf("\n Second \t Recieved \t Sent \t Dropped \tRemained \n");
for(i=0;count || i<nsec;i++)
{
      printf("  %d",i+1);
      printf(" \t\t%d\t ",inp[i]);
      printf(" \t%d\t ",MIN((inp[i]+count),orate));
      if((x=inp[i]+count-orate)>0)
      {
            if(x>cap)
            {
                  count=cap;
                  drop=x-cap;
            }
            else
            {
                  count=x;
                  drop=0;
            }
      }
      else
      {
            drop=0;
            count=0;
      }
      printf(" \t %d\t %d \n",drop,count);
}
return 0;
}
```