```
1)


the number of bytes in the frame and receiver module should display each fram
received */

#include<stdio.h>
#include<string.h>
#include<stdlib.h>

char res[100];

void sender()
{
    int n,i,len;
    char frame[100],l[100];
    printf("Enter the number of frames\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter the frame %d\n",i+1);
        scanf("%s",&frame);
        len=strlen(frame);
        printf("Number of bytes in the frame %d = %d\n",i+1,len);
        sprintf(l,"%d",len);
        strcat(l,frame);
        strcat(res,l);
    }

}

void reciever()
{
    int len,i,j;
    printf("Received frame \n");
    for(i=0;i<strlen(res);i++)
    {
        len=res[i]-'0';
        for(j=i+1;j<=(i+len);j++)
            printf("%c",res[j]);
            i=i+len;
        printf("\n");
    }
}

void main()
{
    sender();
```

```
    reciever();
    return 0;
}
```

2)

```
/* Program to implement Bit Stuffing concept in datalink layer */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

void receiver(int *frame, int l2)
{
    int i, j, counter, l3;
    int msg[100];
    l3=l2-8;
    j=0;
    for(i=8;i<l3;i++)
    {
        if(frame[i]==0)
        {
            if(counter==5)
            {
                msg[j]=frame[i];
                 j++;
                counter=0;
            }
            else
            {
                msg[j]=frame[i];
                j++;
                counter++;
            }
        }
        else
        {
            msg[j]=frame[i];
            j++; counter++;
        }
    }
    printf("Received message is: \n");
    for(i=0;i<j;i++)
        printf("%d",msg[i]);
    printf("\n");
}
```

```
void sender()
{
    int data[100], frame[100], framelen=0, n, i, j=8;
    int count, zeroadded=0, zero;
    printf("Enter the number of bits\n");
    scanf("%d",&n);
    printf("Enter data for bits\n");
    for(i=0;i<n;i++)
        scanf("%d",&data[i]);
    frame[0]=0;
    frame[1]=1;
    frame[2]=1;
    frame[3]=1;
    frame[4]=1;
    frame[5]=1;
    frame[6]=1;
    frame[7]=0;
    for(i=0;i<n;i++)
    {
        if(data[i]==0)
        {
            frame[j]=data[i];
            j++;
            count=0;
            zero=1;
        }
        else
        {
            if((count==5)&&(zero==1))
            {
                frame[j]=0;
                j++;
                zeroadded++;
                frame[j]=data[i];
                j++;
                count=0;
            }
            else
            {
                frame[j]=data[i];
                 j++;
                count++;
            }
        }
    }
    frame[j++]=0;
    frame[j++]=1;
    frame[j++]=1;
```

```
    frame[j++]=1;
    frame[j++]=1;
    frame[j++]=1;
    frame[j++]=1;
    frame[j++]=0;
    framelen=n+16+zeroadded;
    printf("length of frame sent %d \n",framelen);
    printf("Frame sent: ");
    for(i=0;i<framelen;i++)
        printf("%d",frame[i]);
    printf("\n");
    receiver(frame,framelen);
}

void main()
{
    sender();
}
```

3)CRC error check

```
#include<stdio.h>
#include<stdlib.h>
main()
{
    int    c[50],  b[50],  a[17]={1,  0,  0};
    int  i,  j,  m,  n=3,  q,  r,  x,  y,  z,  e,  pos,  fail=1;
    printf  ("enter  no  of  bits  for  messages:\n");
    scanf  ("%d",&m);
    printf  ("enter  the  message  to  be  transmitted:\n");
    for  (i=0;  i<m;  i++)
    {
            scanf  ("%d",  &b[i]  );
                        c[i]  =  b[i];
    }
    for  (i=m;  i<m+n-1;  i++) // append n-1 zeros at the end of message
                    b[i]  =  0;
    if  (  m<n  )
    {
          printf  ("error!!!  enter  bits  again");
          exit  (0);
    }
    else
    {
          y=0;
          for  (i=0;i<m;i++)
```

```
        {
              if  (b[y]==1)
                              for  (x=y,j=0;  x<y+n;  x++,  j++)
                 b[x]  =  b[x]  ^  a[j];
              else
             for  (x=y;  x<y+n;  x++)
                 b[x]  =  b[x]  ^  0;
              y++;
        }
  }
  for  (i=m;  i<m+n-1;  i++)
                    c[i]  =  b[i];
  printf  ("message  to  be  sent  is:\n");
  for  (i=0;  i<m+n-1;  i++)
                 printf  ("%d",  c[i]);
  printf  ("\nintroduce  error??  yes  or  no[1  or  0]:\n");
  scanf  ("%d",  &e);
  if  (e==1)
  {
                 printf("enter  the  position  to  be  changed:");
                 scanf("%d",&pos);
      if(  pos>m)
          printf  ("\ninvalid  position!!");
      else
          if(  c[pos-1]==0)
                                  c[pos-1]=1;
          else
              c[pos-1]=0;
  }
  printf  ("message  received  at  receiver  site:\n");
  for  (i=0;  i<m+n-1;  i++)
      printf  ("%d",  c[i]);
  z  =  0;
  for  (i=0;  i<m;  i++)
  {
        if  (c[z]==a[0])
      {
                              for  (x=z,j=0;  x<z+n;  x++,  j++)
              c[x]  =  c[x]  ^  a[j];
      }
      else
      {
                              for  (x=z;  x<z+n;  x++)
              c[x]  =  c[x]  ^  0;
      }
      z++;
  }
  for  (i=0;  i<m+n-1;  i++)
```

```
    {
          if  (c[i]==1)
        {
            printf  ("\n error  in  the  message!!!\n");
            fail  =  0;
            break;
        }

    }
    if (fail)
        printf ("\n successful    transfer    of    message\n");
}
```

4)Leaky bucket

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,j,qs,ns,t,count,size,a,choice,p[10],cap,rate,delay,flag=1,t1,t2;
    printf("enter the queue size:");
    scanf("%d",&size);
    count=size;
    printf("enter leaky bucket capacity:");
    scanf("%d",&cap);
    qs=cap;
    printf("enter the size of the packets in the queue:");
    for(i=0;i<size;i++)
    {
        scanf("%d",&a);
        if(a>cap)
        {
            i--;
            printf("packets cannot be entered");
        }
        else
            p[i]=a;
    }
    printf("enter the output rate:");
    scanf("%d",&rate);
    delay=cap/rate;
    printf("delay=%d\n",delay);
    while(flag)
```

```
    {
        qs=cap;
        while(qs>=p[0]&&count>0)
        {
            printf("\npacket of size %d is put into the bucket\n",p[0]);
            qs=qs-p[0];
            printf("\navailable space %d\n",qs);
            count--;
            for(i=0;i<count;i++)
                p[i]=p[i+1];
        }
        t=delay;
        long int t1=(long)time(NULL);
        long int t2=(long)time(NULL);
        while((t2-t1)<delay)
        {
            t2=(long)time(NULL);
            if((delay-t)==(t2-t1))
            {
                printf("\ntransmitting packets in the leaky bucket:%d
seconds\n",t);
                t--;
            }
        }
        printf("\npackets in the queue:\n");
        for(i=0;i<count;i++)
            printf("%d\t",p[i]);
        printf("\ndo u want to enter more packets in the queue? (1 or 0)\n");
        scanf("%d",&choice);
        while(choice&&count<size)
        {
            printf("enter the no of packets (<=%d)\n",size-count);
            scanf("%d",&ns);
            if(ns>(size-count))
                printf("\nexceeding queue size\n");
            else
            {
                printf("\nenter the size of the packets to put in the
queue:\n");
                for(i=0;i<ns;i++)
                {
                    scanf("%d",&a);
                    if(a>cap)
                        printf("packets cannot be entered");
                    else
                        p[count++]=a;
                }
            }
```

```
            printf("\ndo u want to enter more? (0 or 1)\n");
            scanf("%d",&choice);
            if(choice!=0)
                if(count>=size)
                    printf("queue is full");
        }
        if(count==0)
            flag=0;
    }
}
```

5)Distance vector

```
#include<stdio.h>
struct node
{
    int dist[20];
    int from[20];
}rt[10];
int main()
{
    int  dmat [20] [20];
    int n, i, j, k, count=1;
    printf ("\nEnter the number of nodes :\n");
    scanf ("%d", &n);
    printf ("\nEnter the cost matrix :\n");
    for (i=1; i<=n; i++)
        for (j=1; 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
    {
        for (i=1; i<=n; i++)
            for (j=1; j<=n; j++)
                for (k=1; 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 < n);
```

```
    for (i=1; i<=n; i++)
    {
        printf ("\nDistance Table for router %c is \n", i+64);
        for (j=1; j<=n; j++)
            printf ("\tNode %d Via %d, Distance : %d\n", j, rt[i].from[j],
rt[i].dist[j]);
    }
    return 0;
}
```

6)**UDP client**

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>

#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
      char buffer[100];
      char *message = "Hello Server";
      int sockfd, n;
      struct sockaddr_in servaddr;

      // clear servaddr
      bzero(&servaddr, sizeof(servaddr));
      servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
      servaddr.sin_port = htons(PORT);
      servaddr.sin_family = AF_INET;

      // create datagram socket
      sockfd = socket(AF_INET, SOCK_DGRAM, 0);

      // connect to server
      if(connect(sockfd, (struct sockaddr *)&servaddr,
sizeof(servaddr)) < 0)
      {
            printf("\n Error : Connect Failed \n");
            exit(0);
      }
      // request to send datagram
      // no need to specify server address in sendto
```

```
      // connect stores the peers IP and port
      sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)NULL,
sizeof(servaddr));
      // waiting for response
      recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct
sockaddr*)NULL, NULL);

      puts(buffer);
    printf("\nMessage recieved from client\n");

      // close the descriptor
      close(sockfd);
}
```

UDP server

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
      char buffer[100];
      char *message = "Hello Client";
      int listenfd, len;
      struct sockaddr_in servaddr, cliaddr;
      bzero(&servaddr, sizeof(servaddr));


      // Create a UDP Socket
      listenfd = socket(AF_INET, SOCK_DGRAM, 0);
      servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
      servaddr.sin_port = htons(PORT);
      servaddr.sin_family = AF_INET;

      // bind server address to socket descriptor

      bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
    printf("Waiting for client request....\n");

      //receive the datagram
      len = sizeof(cliaddr);
      int n = recvfrom(listenfd, buffer, sizeof(buffer),
                  0, (struct sockaddr*)&cliaddr,&len); //receive
message from server
      buffer[n] = '\0';
      puts(buffer);

      // send the response
```

```
      sendto(listenfd, message, MAXLINE, 0,
            (struct sockaddr*)&cliaddr, sizeof(cliaddr));
}
```

7)TCP client

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#define PORT_ID 8000
Int main()
{

   Char buf[30000];
   Int fd1,n;
   Struct sockaddr_in s;
   System("clear");
   Printf("Enter the filename to be sent to the server\n");
   Scanf("%s",buf);

   s.sin_family=AF_INET;
   s.sin_port=htons(PORT_ID);
   s.sin_addr.s_addr=inet_addr("127.0.0.1");

   fd1=socket(AF_INET,SOCK_STREAM,0);

   if((connect(fd1,(struct socketaddr *)&s,sizeof(struct sockaddr)))==-1)
      printf("Error in socket binding\n");

   send(fd1,buf,strlen(buf),0);

   printf("****Contents of the requested file is **** \n");
   while((n=recv(fd1,buf,sizeof(buf),0))>0)
   {
      Buf[n]='\0';
      Printf("%s",buf);
   }
   Printf("\n");
   Close(fd1);
   Return 0;

}
```

TCP server

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

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#define PORT_ID 8000


Int main()
{
   Char buf[300];
   Int fd1,fd2,size,n;
   Struct sockaddr_in s;
   System("clear");
   Printf("Server is getting ready\n");

   s.sin_family=AF_INET;
   s.sin_port=htons(PORT_ID);
   s.sin_addr.s_addr=inet_addr("127.0.0.1");

   fd1=socket(AF_INET,SOCK_STREAM,0);

   if((bind(fd1,(struct sockaddr *)&s,sizeof(struct sockaddr)))==-1)
      printf("Error in socket binding\n");
   if((listen(fd1,5))==-1)
   printf("Error in listening\n");
   printf("Waiting for client request\n");
   size=sizeof(struct sockaddr);

   fd2=accept(fd1,(struct sockaddr *)&s,&size);
   size=recv(fd2,buf,sizeof(buf),0);
   buf[size]='\0';
   printf("Filename received is %s\n",buf);
   if((fd1=open(buf,O_RDONLY))!=-1)
   {
      While((n=read(fd1,buf,sizeof(buf)))>0)
         Send(fd2,buf,n,0);

   }
   Else
      Send(fd2,"File not found",20,0);

      Close(fd1);
      Close(fd2);
      Printf("Server terminated");
      Return 0;


}
```