Bit Stuff:

```
#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]);

    }

}
```

Byte Stuff:

```
#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");

}

}
```

CRC:

```
#include<stdio.h>

#include<conio.h>

int rem(int,int);

void main()

{

  int i,j,k,dl,dil;

  int data[10],div[5],newdata[15],crc[5],datacrc[15],revdata[15],remd[5];

  printf("\n Enter the data length= ");

  scanf("%d",&dl);

  printf("\n Enter the divisor  length= ");

  scanf("%d",&dil);

  printf("\n Enter the data : ");

  for(i=0;i<dl;i++)

   scanf("%d",&data[i]);

  printf("\n Enter the divisor : ");
```

```
 for(i=0;i<dil;i++)
  scanf("%d",&div[i]);
 printf("\n The new data is : ");
 for(i=0;i<(dl+dil-1);i++)
 {
   if(i<dl)
    newdata[i]=data[i];
   else
    newdata[i]=0;
   printf("%d",newdata[i]);
 }
for(j=0;j<=dl;j++)
{
           for(i=0;i<dil;i++)
           {
            crc[i]=newdata[i+j];
            if(crc[0]==1)
             newdata[i+j]=rem(newdata[i+j],div[i]);
            else
             newdata[i+j]=rem(newdata[i+j],0);
           }
printf("\n The Crc is : ");
for(i=0;i<dil-1;i++)
 printf("%d",crc[i]);
}


printf("\n The data to be send is : ");
for(i=0;i<(dl+dil-1);i++)
{
 if(i<dl)
```

```
  datacrc[i]=data[i];
 else
  datacrc[i]=crc[i-dl];
 printf("%d",datacrc[i]);
}
printf("\n Enter the receiver side data : ");
for(i=0;i<(dl+dil-1);i++)
 scanf("%d",&revdata[i]);
 for(j=0;j<=dl;j++)
{
          for(i=0;i<dil;i++)
          {
           remd[i]=revdata[i+j];
           if(remd[0]==1)
            revdata[i+j]=rem(revdata[i+j],div[i]);
           else
            revdata[i+j]=rem(revdata[i+j],0);
          }
printf("\n The reminder is : ");
k=0;
for(i=0;i<dil-1;i++)
 {
 printf("%d",remd[i]);
  if(remd[i]==0)
   k++;
 }
}
if(k==dil-1)
printf("\n There is no error found.");
else
```

```
 printf("\n There is error found.");
 getch();
}

int rem(int x, int y)
{
 if(x==y)
  return 0;
 else
  return 1;
}
```

TCP Client:

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
	char buff[MAX];
	int n;
	for (;;) {
		bzero(buff, sizeof(buff));
		printf("Enter the string : ");
		n = 0;
		while ((buff[n++] = getchar()) != '\n')
```

```
                ;
            write(sockfd, buff, sizeof(buff));
            bzero(buff, sizeof(buff));
            read(sockfd, buff, sizeof(buff));
            printf("From Server : %s", buff);
            if ((strncmp(buff, "exit", 4)) == 0) {
                printf("Client Exit...\n");
                break;
            }
        }
}

int main()
{
        int sockfd, connfd;
        struct sockaddr_in servaddr, cli;

        // socket create and varification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
            printf("socket creation failed...\n");
            exit(0);
        }
        else
            printf("Socket successfully created..\n");
        bzero(&servaddr, sizeof(servaddr));

        // assign IP, PORT
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
	servaddr.sin_port = htons(PORT);


	// connect the client socket to server socket
	if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
		printf("connection with the server failed...\n");
		exit(0);
	}
	else
		printf("connected to the server..\n");


	// function for chat
	func(sockfd);


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

TCP Server:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr


// Function designed for chat between client and server.
```

```
void func(int sockfd)

{

        char buff[MAX];

        int n;

        // infinite loop for chat

        for (;;) {

                bzero(buff, MAX);


                // read the message from client and copy it in buffer

                read(sockfd, buff, sizeof(buff));

                // print buffer which contains the client contents

                printf("From client: %s\t To client : ", buff);

                bzero(buff, MAX);

                n = 0;

                // copy server message in the buffer

                while ((buff[n++] = getchar()) != '\n')

                        ;


                // and send that buffer to client

                write(sockfd, buff, sizeof(buff));


                // if msg contains "Exit" then server exit and chat ended.

                if (strncmp("exit", buff, 4) == 0) {

                        printf("Server Exit...\n");

                        break;

                }

        }

}


// Driver function
```

```
int main()

{

        int sockfd, connfd, len;

        struct sockaddr_in servaddr, cli;


        // socket create and verification

        sockfd = socket(AF_INET, SOCK_STREAM, 0);

        if (sockfd == -1) {

                printf("socket creation failed...\n");

                exit(0);

        }

        else

                printf("Socket successfully created..\n");

        bzero(&servaddr, sizeof(servaddr));


        // assign IP, PORT

        servaddr.sin_family = AF_INET;

        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

        servaddr.sin_port = htons(PORT);


        // Binding newly created socket to given IP and verification

        if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {

                printf("socket bind failed...\n");

                exit(0);

        }

        else

                printf("Socket successfully binded..\n");


        // Now server is ready to listen and verification

        if ((listen(sockfd, 5)) != 0) {
                printf("Listen failed...\n");
                exit(0);
        }
        else
                printf("Server listening..\n");
        len = sizeof(cli);

        // Accept the data packet from client and verification
        connfd = accept(sockfd, (SA*)&cli, &len);
        if (connfd < 0) {
                printf("server acccept failed...\n");
                exit(0);
        }
        else
                printf("server acccept the client...\n");

        // Function for chatting between client and server
        func(connfd);

        // After chatting close the socket
        close(sockfd);
}
```

Udp 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;

}
```

UDP 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;

}
```

Distance vector:

```
#include<stdio.h>

struct node

{

    unsigned dist[20];

    unsigned from[20];

}rt[10];
```

```
int main()

{

   int costmat[20][20];

   int nodes,i,j,k,count=0;

   printf("\nEnter the number of nodes : ");

   scanf("%d",&nodes);//Enter the nodes

   printf("\nEnter the cost matrix :\n");

   for(i=0;i<nodes;i++)

   {

      for(j=0;j<nodes;j++)

      {

         scanf("%d",&costmat[i][j]);

         costmat[i][i]=0;

         rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix

         rt[i].from[j]=j;

      }

   }

      do

      {

         count=0;

         for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the
node i to k using the cost matrix

         //and add the distance from k to node j

         for(j=0;j<nodes;j++)

         for(k=0;k<nodes;k++)

            if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

            {//We calculate the minimum distance

               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<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
            for(j=0;j<nodes;j++)
            {
                printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
            }
        }
    printf("\n\n");
    return 0;
}
```

Leaky bucket:

```
#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]);
        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%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;
        }
```