1.Write a program for a distance vector algorithm to find a suitable path for transmission.

```
#include<stdio.h>

void dist_vector(int n);

void init(int n);


//creating structure node

struct node

{

        unsigned int dist[20],nexthop[20];

}route[10];


int i,j;

//main function

void main()

{

        int n,i,j;

        printf("Enter the number of router");

        scanf("%d",&n);

        init(n);


        printf("Enter the cost matrix(999 no link)\n");

        for(i=0;i<n;i++)
        for(j=0;j<n;j++)
                scanf("%d",&route[i].dist[j]);
dist_vector(n);
printf("\n------------------------------------");
printf("\nupdated distance vector table\n");
printf("--------------------------------------\n");
for(i=0;i<n;i++)
{
        for(j=0;j<n;j++)
        {
                printf("%d\t",route[i].dist[j]);


        }
printf("\n");
}
printf("\n----------------------------\n");
for(i=0;i<n;i++)
{
        printf("\nRouting table for node %c table\n",65+i);
        printf("-------------------------\n");
        printf("desti\t Cost\t Next hop\n");
        printf("--------------------------\n");
```

```
                    for(int j=0;j<n;j++)
                    if(i!=j)

printf("%c\t%d\t%c\n",65+j,route[i].dist[j],65+route[i].nexthop[j]);
          }
}


//initialization
void init(int n)
{
          int i,j;
          for(i=0;i<n;i++)
          { for(j=0;j<n;j++)

           { if(i!=j)
                    {
                    route[i].dist[j]=999;
                    route[i].nexthop[j]=-20;
                    }
          route[i].dist[i]=0;
          route[i].nexthop[j]=-20;
           }
```

```
        }
}

void dist_vector(int n)
{
int count;
do {
count = 0;
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
for (int k = 0; k < n; k++)
{
if ((route[i].dist[j]) > (route[i].dist[k] + route[k].dist[j]))
{
route[i].dist[j] = route[i].dist[k] + route[k].dist[j];
route[i].nexthop[j] = k;
count = 1;
}
}
}
}
} while (flag);
}
```

2. Using TCP/IP sockets, write a client-server program to make the client send the file name and to make the server send back the contents of the requested file if present.

```
//client
#include<stdio.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<unistd.h>
int main()
{
int soc,n;
int port=5556;
char buffer[1024],fname[50];
struct sockaddr_in addr;

soc=socket(PF_INET,SOCK_STREAM,0);
printf("Client Socket created");
addr.sin_family=AF_INET;
addr.sin_port=port;
addr.sin_addr.s_addr=inet_addr("127.0.0.1");


while(connect(soc,(struct sockaddr*) &addr,sizeof(addr)));
printf("\n CLIENT IS CONNECTED TO SERVER\n");


printf("Enter the file name");
scanf("%s",fname);
send(soc,fname,sizeof(fname),0);


printf("Recieved response\n");
while((n=recv(soc,buffer,sizeof(buffer),0))>0)
printf("%s",buffer);
return 0;
}
```

```
//server
#include<stdio.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<unistd.h>
int main()
{
int server_soc,fd,n,client_soc;
int port=5556;

char buffer[1024],fname[50];
struct sockaddr_in addr;

server_soc=socket(PF_INET,SOCK_STREAM,0);
printf("Socket created....\n");

addr.sin_family=AF_INET;
addr.sin_port=port;
addr.sin_addr.s_addr=inet_addr("127.0.0.1");
```

```
bind(server_soc,(struct sockaddr*)&addr,sizeof(addr));
printf("bind to th port number %d",port);
printf("\nSERVER IS ONLINE\n");


listen(server_soc,5);
printf("Listining....\n");


client_soc=accept(server_soc,NULL,NULL);
printf("server accepted......\n");


recv(client_soc,fname,50,0);
printf("client Requesting for file %s:",fname);
fd=open(fname,O_RDONLY);
if(fd<0)
send(client_soc,"\nFile not found...",25,0);
else
while((n=read(fd,buffer,sizeof(buffer))) > 0)
send(client_soc,buffer,n,0);
close(fd);
```

```
printf("\ndisconnected......");

return 0;


}
```

3. Write a program for Hamming code generation for error detection and correction.

```
#include<stdio.h>

#include<math.h>



int input[5];int messege[20];

int ham_calc(int,int);

void main()

{

        int n,i,par_bits=0;

        printf("Enter the length of messege");

        scanf("%d",&n);

        printf("Enter the Message in bits");

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

        {

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

        }

//redundant bits to add

i=1;

while(pow(2,i)<=n+i+1)
```

```
{
        i++;
        par_bits++;
}
int tot_bits=par_bits+n;
printf("\n total number of bits:%d\n",tot_bits);


//positioning parity
int j=0,k=0;
for(i=0;i<tot_bits;i++)
{
        if(i==((int)pow(2,k)-1))
        {
                messege[i]=0;
                k++;
        }
        else
        {
                messege[i]=input[j];
                j++;
        }
}

for(i=0;i<tot_bits;i++)
{


        printf("Initialized value%d\t",messege[i]);
}
//updating the parity value
for(i=0;i<par_bits;i++)
{
        int position = (int)pow(2,i);
        int value = ham_calc(position,tot_bits);
        messege[position-1]=value;
}

printf("\nThe calculated Code Word is: ");
for(i=0;i<tot_bits;i++)
        printf("%d",messege[i]);
        printf("\n");
        printf("Please enter the received Code Word:\n");
for(i=0;i<tot_bits;i++)
```

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



int error_pos = 0;
for(i=0;i<par_bits;i++)
{
        int position = (int)pow(2,i);
        int value = ham_calc(position,tot_bits);
        if(value != 0)
                error_pos+=position;
}
if(error_pos == 1)
        printf("The received Code Word is correct.\n");
else
        printf("Error at bit position: %d\n",error_pos);



}
```

```
int ham_calc(int position,int tot_bits)
{
        int count=0,i,j;
        i=position-1;
        while(i<tot_bits)
        {
                for(j=i;j<i+position;j++)
                {
                        if(messege[j] == 1)
                        count++;
                }
                i=i+2*position;
        }
        if(count%2 == 0)
                return 0;
        else
                return 1;
}
```

4. Write a program for congestion control using leaky bucket algorithm.

```
#include<stdio.h>
void main()
{
int i, n,buck_size=0,packets[10],rate=0,remai_pack=0,recv=0,sent;
printf("Enter the number of packets");
scanf("%d",&n);
printf("enter the value of packets");
for(i=0;i<n;i++)
{

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

}
printf("Enter the bucket_size");
scanf("%d",&buck_size);
printf("Enter the rate of transmission");
scanf("%d",&rate);
printf("\n--------------------------------------------------------------------------------------------\n");
printf("index\t packet size \t\t accept \tsent\tremaining\n");
printf("\n--------------------------------------------------------------------------------------------
------------\n");

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

{

        if(packets[i]==0)

        {

        recv=-1;

        sent=0;

        }else{

                if(remai_pack+packets[i]>buck_size)

                recv=-1;

                else

                {

                recv=packets[i];

                remai_pack+=packets[i];

                }

                if(remai_pack!=0)

                {

                        if(remai_pack<rate)

                        {

                                sent=remai_pack;

                                remai_pack=0;
                        }
                        else
                        {
                                sent=rate;
                                remai_pack=remai_pack-rate;
                        }
                }
                else
                sent=0;
                }
                if(recv==-1)

printf("\n%d\t\t%d\t\t%s\t\t%d\t\t%d",i,packets[i],"dropped",sent,remai_pack
);
                else

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,packets[i],recv,sent,remai_pack);
}
}
```