COMPUTER NETWORKS LAB SYLLABUS

III Year IT - I Sem

EXPERIMENT NO: 1

1(a)

1. Implement the data link layer farming methods such as character, character-stuffing and bit stuffing.

NAME OF THE EXPERIMENT: **Bit Stuffing**.
OBJECTIVE: Implement the data link layer framing method.
RESOURCE: Turbo C

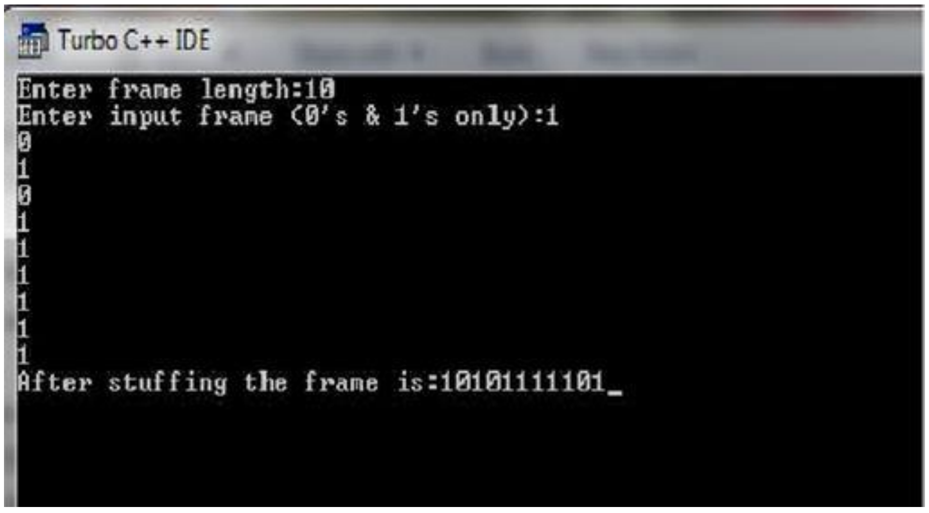
PROGRAM LOGIC:

The new technique allows data frames to contain an arbitrary number if bits and allows character codes with an arbitrary no of bits per character. Each frame begins and ends with special bit pattern, 01111110, called a flag byte. Whenever the sender's data link layer encounters five consecutive ones in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the outgoing character stream before DLE in the data.

SOURCE CODE:

```
// BIT Stuffing
program
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int a[20],b[30],i,j,k,count,n;
clrscr();
printf("Enter frame
length:");scanf("%d",&n);
printf("Enter input frame (0's & 1's
only):");for(i=0;i<n;i++)
scanf("%d",&a[i])
; i=0;
count=1;
j=0;
while(i<n)
{
if(a[i]==1)
{
b[j]=a[i];
for(k=i+1;a[k]==1  &&  k<n  &&count<5;k++)
{
 j++;
b[j]=a[k];
count++;
if(count==5)
{
 j++;
b[j]=0;
}
i=k;
}
}
else
{
b[j]=a[i];
}
i++;
j++;
}
printf("After stuffing the frame
is:");for(i=0;i<j;i++)
printf("%d",b[i]);
getch();
}
```

OUTPUT:



```
Enter frame length:10
Enter input frame (0's & 1's only):1
0
1
0
1
1
1
1
1
1
After stuffing the frame is:10101111101_
```

Viva questions:

1. What is Stuffing?
2. What is use of Stuffing?
3. With bit stuffing the boundary between two frames can be unambiquously recognize by?

4.is a analogous to character stuffing?

5. The senders data link layer encounters ......no of 1’s consecutively

EXPERIMENT NO: 1

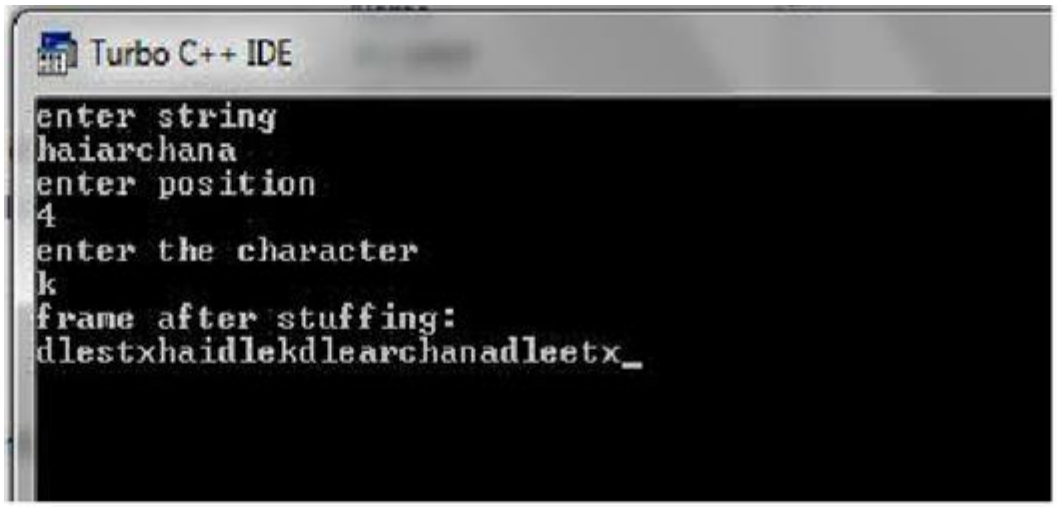
1(b)

NAME OF THE EXPERIMENT: Character Stuffing.
OBJECTIVE: Implement the data link layer framing methods.
RESOURCE: Turbo C

PROGRAM LOGIC:
The framing method gets around the problem of resynchronization after an error by ha vi ng ea ch fra me sta r t with th e ASCII c ha ra cter s equ en ce D L E ST X a nd the sequence DLE ETX. If the destination ever losses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link la yer on t he recei vin g en d re mo ve s th e D LE b efore th e da ta a re giv en t o th e network layer.This technique is called character stuffing.
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:
```
//PROGRAM FOR CHARACTER STUFFING
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<process.h>
void main()
{
int i=0,j=0,n,pos;char
a[20],b[50],ch;clrscr();
printf("enter
string\n");
scanf("%s",&a);
n=strlen(a);
printf("enter
position\n");
scanf("%d",&pos);
if(pos>n)
{
printf("invalid position, Enter again
:");scanf("%d",&pos);
}
printf("enter the
character\n");ch=getche();
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
if(i==pos-1)
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]=ch;
b[j+4]='d';
b[j+5]='l';
b[j+6]='e';
j=j+7;
}
if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
 j=j+3;
}
b[j]=a[i];
i++;
j++;
}
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]='e';
b[j+4]='t';
b[j+5]='x';
b[j+6]='\0';
printf("\nframe after stuffing:\n");
printf("%s",b);
getch();
}
```

OUTPUT:



```
enter string
haiarchana
enter position
4
enter the character
k
frame after stuffing:
dlestxhaidlekdlearchanadleetx_
```

Viva Questions:

1. What is Character stuffing?
2. What is the use of character stuffing?
3. What are the delimiters for the character stuffing?
4. Expand DLE STX?
5. Expand DLE ETX?

EXPERIMENT NO: 2

2. Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP

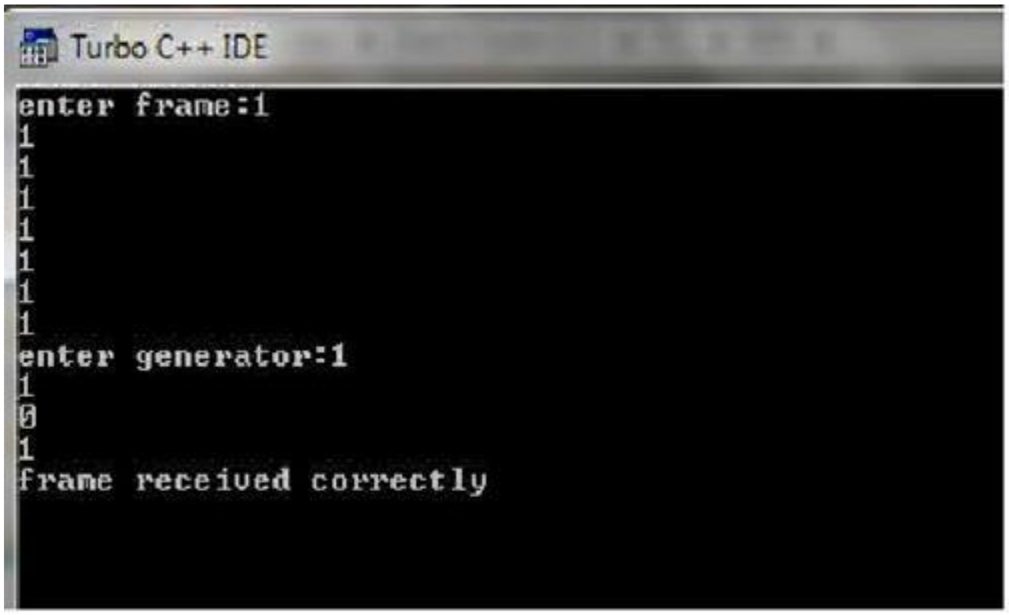
NAME OF THE EXPERIMENT: Cyclic Redundancy Check.

OBJECTIVE: Implement on a data set of characters the three CRC polynomials – CRC 12,CRC 16 and CRC CCIP.
RESOURCE: Turbo C

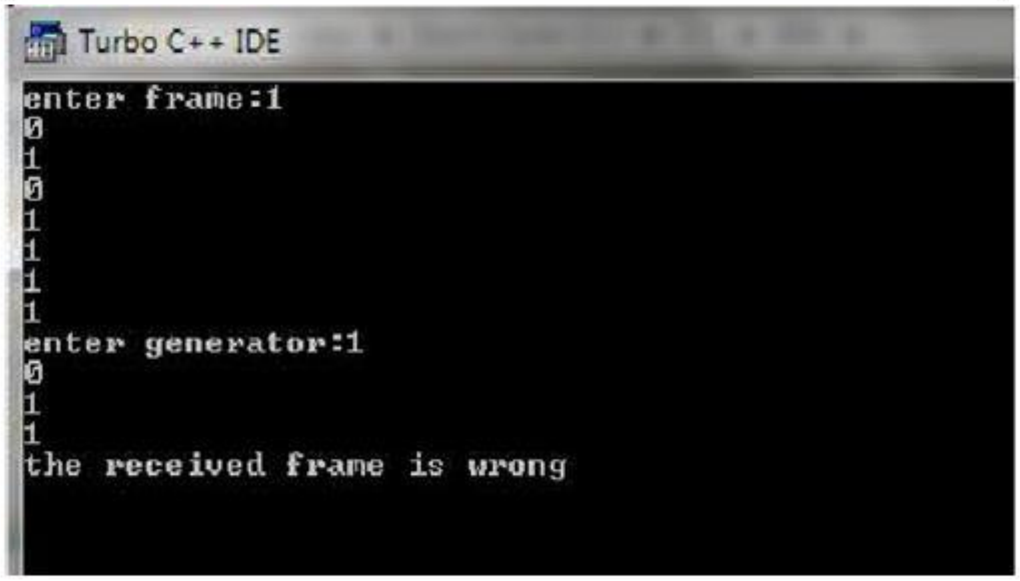
PROGRAM LOGIC:
CRC method can detect a single burst of length n, since only one bit per column will be changed, a burst of length n+1 will pass undetected, if the first bit is inverted, the last bit is inverted and all other bits are correct. If the block is badly garbled by a long burst or by multiple shorter burst, the probability that any of the n columns will have the correct parity that is 0.5. so the probability of a bad block being expected when it should not be 2 power(-n). This scheme sometimes known as Cyclic Redundancy Code
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:
```
//PROGRAM FOR CYCLIC REDUNDENCY CHECK
#include<stdio.h>
#include<conio.h>
int gen[4],genl,frl,rem[4];
void main()
{
int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;
clrscr();
frl=8;
genl=4;
printf("enter frame:");
for(i=0;i<frl;i++)
{
scanf("%d",&fr[i]);
dupfr[i]=fr[i];
}
printf("enter generator:");
for(i=0;i<genl;i++)
scanf("%d",&gen[i]);
tlen=frl+genl-1;
for(i=frl;i<tlen;i++)
{
dupfr[i]=0;
}
remainder(dupfr);
for(i=0;i<frl;i++)
{
recfr[i]=fr[i];
}
for(i=frl,j=1;j<genl;i++,j++)
{
recfr[i]=rem[j];
}
remainder(recfr);
flag=0;
for(i=0;i<4;i++)
{
if(rem[i]!=0)
flag++;
}
if(flag==0)
{
printf("frame received correctly");
}
Else
{
printf("the received frame is wrong");
}
getch();
}
remainder(int fr[])
{
int k,k1,i,j;
for(k=0;k<frl;k++)
{
if(fr[k]==1)
{
k1=k;
for(i=0,j=k;i<genl;i++,j++)
{
rem[i]=fr[j]^gen[i];
}
for(i=0;i<genl;i++)
{
fr[k1]=rem[i];
k1++;
}
}
}
}
```

OUTPUT:



```
enter frame:1
1
1
1
1
1
1
1
enter generator:1
1
0
1
frame received correctly
```



```
enter frame:1
0
1
0
1
1
1
1
enter generator:1
0
1
1
the received frame is wrong
```

Viva Questions:

1. What is CRC?
2. What is the use of the CRC?
3. Name the CRC standards?
4. Define Checksum?
5. Define generator polynomial?

EXPERIMENT NO: 3

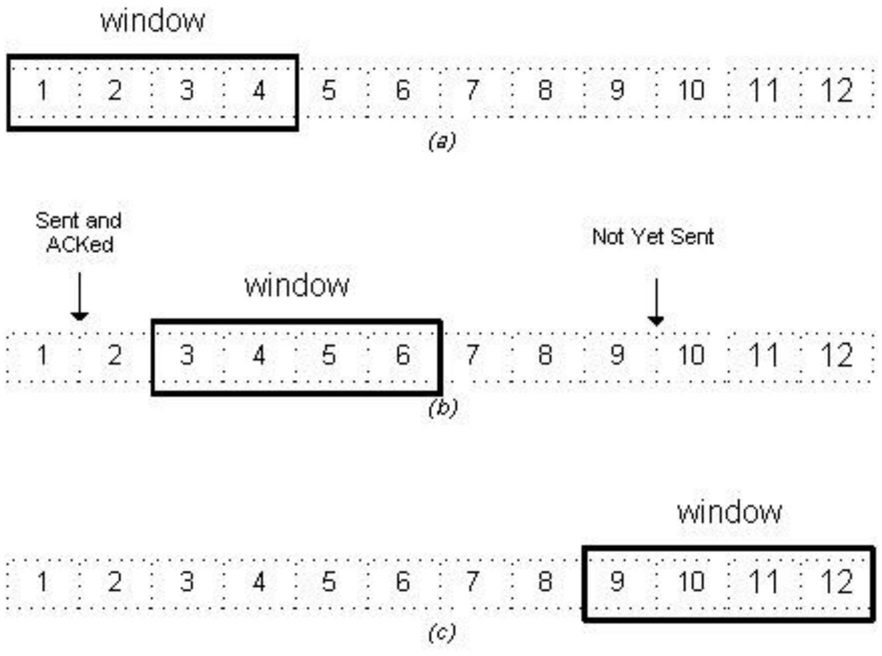
3. Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.

Aim:

In computer networks sliding window protocol is a method to transmit data on a network. Sliding window protocol is applied on the Data Link Layer of OSI model. At data link layer data is in the form of frames. In Networking, Window simply means a buffer which has data frames that needs to be transmitted.

Both sender and receiver agrees on some window size. If window size=w then after sending w frames sender waits for the acknowledgement (ack) of the first frame.

As soon as sender receives the acknowledgement of a frame it is replaced by the next frames to be transmitted by the sender. If receiver sends a collective or cumulative acknowledgement to sender then it understands that more than one frames are properly received, for eg:- if ack of frame 3 is received it understands that frame 1 and frame 2 are received properly.



In sliding window protocol the receiver has to have some memory to compensate any loss in transmission or if the frames are received unordered.

Efficiency of Sliding Window Protocol

$\eta = (W*t_x)/(t_x+2t_p)$

W = Window Size

t_x = Transmission time

t_p = Propagation delay

Sliding window works in full duplex mode

It is of two types:-

1. Selective Repeat: Sender transmits only that frame which is erroneous or is lost.

2. Go back n: Sender transmits all frames present in the window that occurs after the error bit including error bit also.

Sliding Window Protocol Program in C

```
#include<stdio.h>
 int main()
{
    int w,i,f,frames[50];
     printf("Enter window size: ");
    scanf("%d",&w);
     printf("\nEnter number of frames to transmit: ");
    scanf("%d",&f);
     printf("\nEnter %d frames: ",f);
     for(i=1;i<=f;i++)
        scanf("%d",&frames[i]);
     printf("\nWith sliding window protocol the frames will be sent in the following manner (assuming
no corruption of frames)\n\n");
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by the
receiver\n\n",w);
     for(i=1;i<=f;i++)
    {
        if(i%w==0)
        {
            printf("%d\n",frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
        else
            printf("%d ",frames[i]);
    }

    if(f%w!=0)
        printf("\nAcknowledgement of above frames sent is received by sender\n");
     return 0;
}
```

Output

```
Enter window size: 3
Enter number of frames to transmit: 5
Enter 5 frames: 12 5 89 4 6
With sliding window protocol the frames will be sent in the following manner (assuming no
corruption of frames)
After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver
```

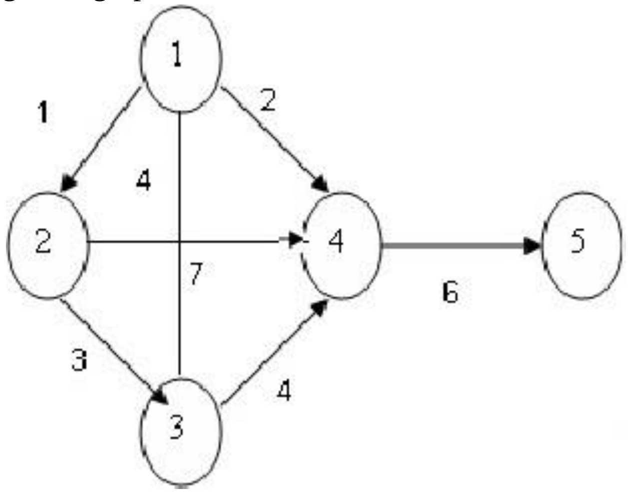
```
12 5 89
Acknowledgement of above frames sent is received by sender
4 6
Acknowledgement of above frames sent is received by sender
```

EXPERIMENT NO: 4

4. Implement Dijsktra’s algorithm to compute the shortest path through a network

NAME OF THE EXPERIMENT: Shortest Path.

OBJECTIVE: Implement Dijkstra‘s algorithm to compute the Shortest path thru a given graph.



RESOURCE: Turbo C

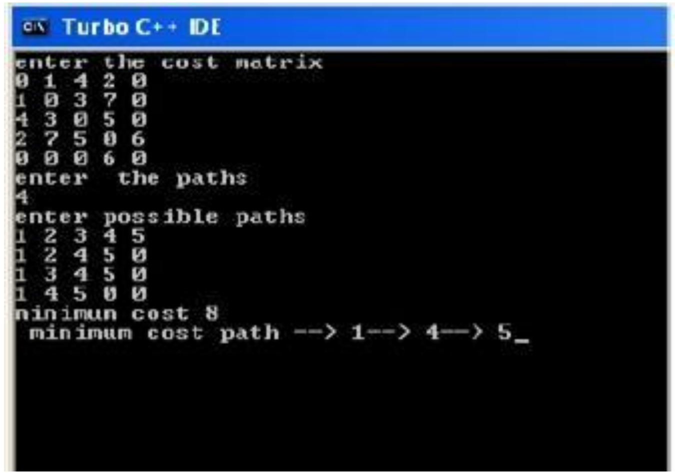
Program Logic: Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, and other set includes vertices not yet included in shortest path tree.
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:
//.PROGRAM FOR FINDING SHORTEST PATH FOR A GIVEN GRAPH//

```
#include<stdio.h>
#include<conio.h>
void main()

{

int path[5][5],i,j,min,a[5][5],p,st=1,ed=5,stp,edp,t[5],index;
clrscr();

printf("enter the cost
matrix\n");for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
```

```
scanf("%d",&a[i][j]);
printf("enter the paths\n");
scanf("%d",&p);

printf("enter possible
paths\n");for(i=1;i<=p;i++)
for(j=1;j<=5;j++)
scanf("%d",&path[i][j]);

for(i=1;i<=p;i++)

{
t[i]=0;

stp=st;
for(j=1;j<=5;j++)

{

edp=path[i][j+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)

break;
else
stp=edp
;

}

}min=t[st];
index=st;
for(i=1;i<=p;i++)

{

if(min>t[i])

{

min=t[i];
index=i;
}

}

printf("minimum cost %d",min);
printf("\n  minimum  cost  path
");for(i=1;i<=5;i++)

{

printf("--> %d",path[index][i]);
if(path[index][i]==ed)

break;

}

getch();

}
```

Output:


Turbo C++ IDE

```
enter the cost matrix
0 1 4 2 0
1 0 3 7 0
4 3 0 5 0
2 7 5 0 6
0 0 0 6 0
enter  the paths
4
enter possible paths
1 2 3 4 5
1 2 4 5 0
1 3 4 5 0
1 4 5 0 0
minimun cost 8
 minimum cost path --> 1--> 4--> 5_
```

Viva questions:

1. Define Dijkstra's algorithm?
2. What is the use of Dijkstra's algorithm?
3. What is path?
4. What is minimum cost path?
5. How to find shortest path using Dijkstra's algorithm?

EXPERIMENT NO: 5

5. Take an example subnet of hosts and obtain a broadcast tree for the subnet.

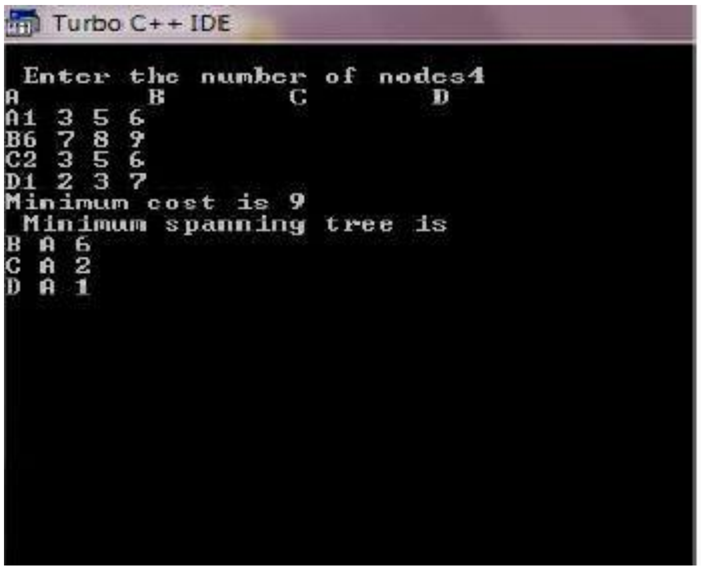
NAME OF THE EXPERIMENT: Broadcast Tree.

OBJECTIVE: Implement broadcast tree for a given subnet of hosts
RESOURCE: Turbo C

PROGRAM LOGIC:
This technique is widely used because it is simple and easy to understand. The idea of this algorithm is to build a graph of the subnet with each node of the graph representing a router and each arc of the graph representing a communication line. To choose a route between a given pair of routers the algorithm just finds the broadcast between them on the graph.
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:
```
// Write a ‘c’ program for Broadcast tree from subnet of host
#include<stdio.h>
#include<conio.h>
int p,q,u,v,n;
int min=99,mincost=0;
int t[50][2],i,j;
int parent[50],edge[50][50];
main()
{
clrscr();
printf("\n Enter the number of nodes");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("%c\t",65+i);
parent[i]=-1;
}
printf("\n");
for(i=0;i<n;i++)
{
printf("%c",65+i);
for(j=0;j<n;j++)
scanf("%d",&edge[i][j]);
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
if(edge[i][j]!=99)
if(min>edge[i][j])
{
min=edge[i][j];
u=i;
v=j;
}
p=find(u);
q=find(v);
if(p!=q)
{
t[i][0]=u;
t[i][1]=v;
mincost=mincost+edge[u][v];
sunion(p,q);
}

Else
{
t[i][0]=-1;t[i][1]=-1;
}
min=99;
}
printf("Minimum cost is %d\n Minimum spanning tree is\n" ,mincost);
for(i=0;i<n;i++)
if(t[i][0]!=-1 && t[i][1]!=-1)
{
printf("%c %c %d", 65+t[i][0],65+t[i][1],edge[t[i][0]][t[i][1]]);printf("\n");
}
getch();
}
sunion(int l,int m)
{
parent[l]=m;
}
find(int l)
{
if(parent[l]>0)

i=parent[i];

return i;

}
```

Output:



```
 Enter the number of nodes4
A       B       C       D
A1 3 5 6
B6 7 8 9
C2 3 5 6
D1 2 3 7
Minimum cost is 9
 Minimum spanning tree is
B A 6
C A 2
D A 1
```

Viva questions:

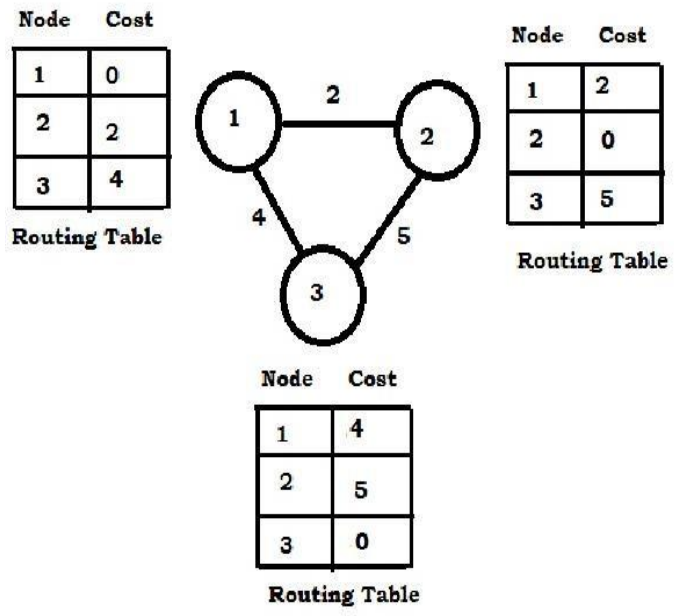
1. What is spanning tree?
2. What is broad cast tree?
3. What are the advantages of broadcast tree?
4. What is flooding?
5. What is subnet?

EXPERIMENT NO: 6

6. Implement distance vector routing algorithm for obtaining routing tables at each node.

NAME OF THE EXPERIMENT: Distance Vector routing.

OBJECTIVE: Obtain Routing table at each node using distance vector routing algorithm for agiven subnet.



RESOURCE: Turbo C

PROGRAM LOGIC:

Distance Vector Routing Algorithms calculate a best route to reach a destination based solely on distance. E.g. RIP. RIP calculates the reach ability based on hop count. It' s different from link state algorithms which consider some other factors like bandwidth and other metrics to reach a destination. Distance vector routing algorithms are not preferable for complex networks and take longer to converge.

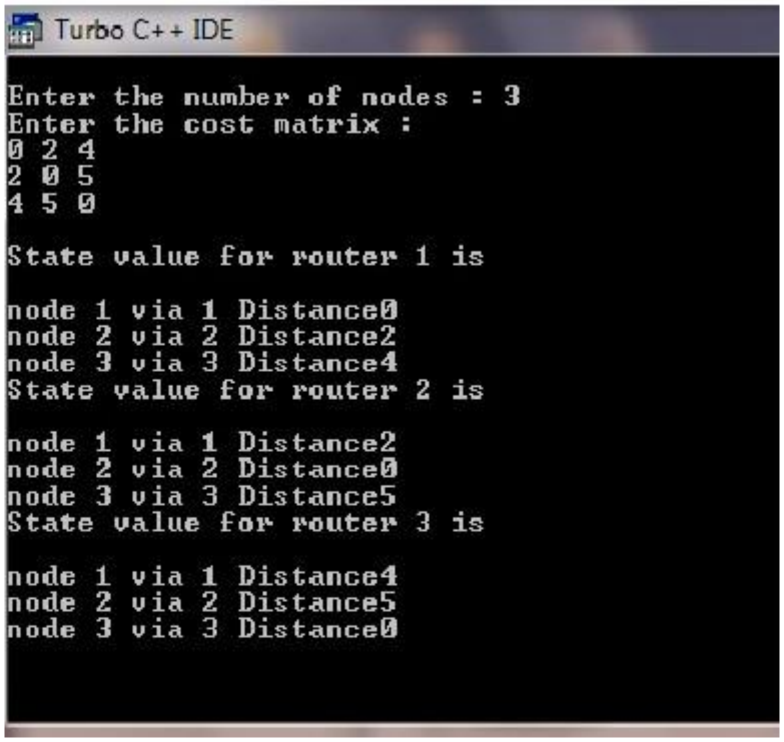
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
 #include<std

io.h>

#include<con
```

```
io.h>    struct

node

{

unsigned

dist[20];

unsigned

from[20];

}rt[10];

int main()
{

int

dmat[20][20

]; int

n,i,j,k,count=

0;clrscr();

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

scanf("%d",&n);printf("Enter 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("\nState value for router %d is
\n",i+1);for(j=0;j<n;j++)
{
printf("\nnode  %d  via  %d  Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
printf("\n");
}
```

Output:



```
Enter the number of nodes : 3
Enter the cost matrix :
0 2 4
2 0 5
4 5 0

State value for router 1 is

node 1 via 1 Distance0
node 2 via 2 Distance2
node 3 via 3 Distance4
State value for router 2 is

node 1 via 1 Distance2
node 2 via 2 Distance0
node 3 via 3 Distance5
State value for router 3 is

node 1 via 1 Distance4
node 2 via 2 Distance5
node 3 via 3 Distance0
```

Viva Questions:

1. What is routing?
2. What is best algorithm among all routing algorithms?
3. What is static routing?
4. Differences between static and dynamic?
5. What is optimality principle?

EXPERIMENTNO: 7
7(a)

7. Implement data encryption and data decryption.

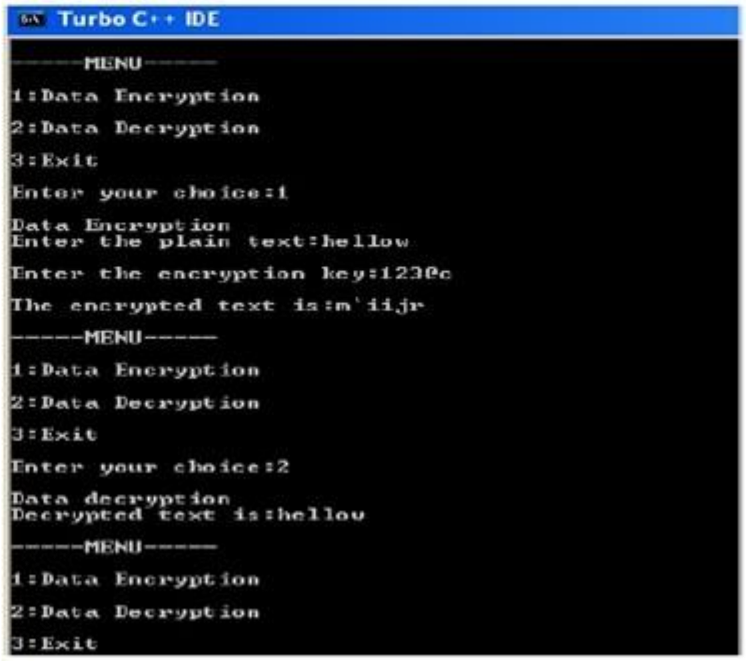
NAME OF THE EXPERIMENT: encrypting DES.

OBJECTIVE: Take a 64 bit playing text and encrypt the same using DES algorithm.
RESOURCE: Turbo C

PROGRAM LOGIC:
Data encryption standard was widely adopted by the industry in security products. Pla in text i s en crypt ed in bl ock s of 6 4 bits yiel din g 6 4 bits of ciph er text. T he algorithm which is parameterized by a 56 bit key has 19 distinct stages. The first stage is a key independent transposition and the last stage is exactly inverse of the t r a n s p o s i t i o n . T h e r e m a i n i n g s t a g e s a r e f u n c t i o n a l l y i d e n t i c a l b u t a r e parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption.
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:
```
#include<std
io.h>
#include<con
io.h>
#include<stdl
ib.h>
#include<stri
ng.h> void
main()
{
int i,ch,lp;char
cipher[50],plain[50];char
key[50];
clrsc
r();
while
(1)
{
printf("\n-----MENU ---- \n");
printf("\n1:Data Encryption\t\n\n2:Data Decryption\t\n\n3:Exit");
printf("\n\nEnter your choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("\nData Encryption");
printf("\nEnter the plain text:");
fflush(stdin);
gets(plain);
printf("\nEnter the encryption key:");
gets(key);
lp=strlen(key);
for(i=0;plain[i]!='\0';i++
) cipher[i]=plain[i]^lp;
cipher[i]='\0';
printf("\nThe encrypted text is:");
puts(cipher);
break;
case 2:
printf("\nData
decryption");
for(i=0;cipher[i]!='\0';i
++)
plain[i]=cipher[i]^lp;
printf("\nDecrypted text
is:");puts(plain);
break;
case3: exit(0);
}
}
getch();
}
```

Output:



```
------MENU------
1:Data Encryption
2:Data Decryption
3:Exit
Enter your choice:1
Data Encryption
Enter the plain text:hellow
Enter the encryption key:123@c
The encrypted text is:m`iijr
------MENU------
1:Data Encryption
2:Data Decryption
3:Exit
Enter your choice:2
Data decryption
Decrypted text is:hellow
------MENU------
1:Data Encryption
2:Data Decryption
3:Exit
```

Viva Questions:

1. Expand DES
2. What is cipher text?
3. What is plain text?
4. Define public key?
5. Define encryption?

EXPERIMENT NO: 7
7(b)

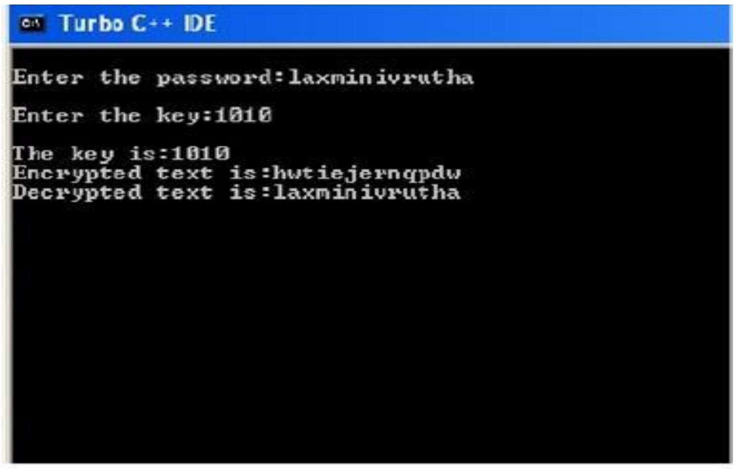
NAME OF THE EXPERIMENT: Decrypting DES.

OBJECTIVE: Write a program to break the above DES coding
RESOURCE: Turbo C

PROGRAM LOGIC:
Data encryption standard was widely adopted by the industry in security products. Pla in text i s en crypt ed in bl ock s of 6 4 bits yiel din g 6 4 bits of ciph er text. T he algorithm which is parameterized by a 56 bit key has 19 distinct stages. The first stage is a key independent transposition and the last stage is exactly inverse of the t r a n s p o s i t i o n . T h e r e m a i n i n g s t a g e s a r e f u n c t i o n a l l y i d e n t i c a l b u t a r e parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption.
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:
```
/*Write a program to break the above DES
coding*/#include<stdio.h>
#include<con
io.h>
#include<stri
ng.h>
#include<cty
pe.h>     void
main()
{
char pwd[20];
char
alpha[26]="abcdefghijklmnopqrstuvwxyz";
int num[20],i,n,key;
clrscr();
printf("\nEnter the
password:");
scanf("%s",&pwd);
n=strlen(pwd);
for(i=0;i<n;i++)
num[i]=toascii(tolower(pwd[i])
)-'a'; printf("\nEnter the
key:"); scanf("%d",&key);
for(i=0;i<n;i++)
num[i]=(num[i]+key)%26;
for(i=0;i<n;i++)
pwd[i]=alpha[num[i]];
printf("\nThe  key
is:%d",key);
printf("\nEncrypted text is:%s",pwd);
for(i=0;i<n;i++)
{
num[i]=(num[i]-
key)%26; if(num[i]<0)
num[i]=26+num[i];
pwd[i]=alpha[num[i]];
}
printf("\nDecrypted text
is:%s",pwd);getch();
}
```

Output:



```
Enter the password:laxminivrutha

Enter the key:1010

The key is:1010
Encrypted text is:hwtiejernqpdw
Decrypted text is:laxminivrutha
```

Viva Questions:

1. Define decryption
2. What is private key?
3. What is cipher feedback mode?
4. Define product cipher
5. What is DES chaining?

EXPERIMENT NO: 8

8. Write a program for congestion control using Leaky bucket algorithm.

Aim:

The congesting control algorithms are basically divided into two groups: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made. Open loop algorithms are further divided into ones that act at source versus ones that act at the destination.
In contrast, closed loop solutions are based on the concept of a feedback loop if there is any congestion. Closed loop algorithms are also divided into two sub categories: explicit feedback and implicit feedback. In explicit feedback algorithms, packets are sent back from the point of congestion to warn the source. In implicit algorithm, the source deduces the existence of congestion by making local observation, such as the time needed for acknowledgment to come back.
The presence of congestion means that the load is (temporarily) greater than the resources (in part of the system) can handle. For subnets that use virtual circuits internally, these methods can be used at the network layer.
Another open loop method to help manage congestion is forcing the packet to be transmitted at a more predictable rate. This approach to congestion management is widely used in ATM networks and is called traffic shaping.
The other method is the leaky bucket algorithm. Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more process are already queued, the new packet is unceremoniously discarded. This arrangement can be built into the hardware interface or simulate d by the host operating system. In fact it is nothing other than a single server queuing system with constant service time.
The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packet onto the network, smoothing out bursts and greatly reducing the chances of congestion.

Algorithm:

1. Start
2. Set the bucket size or the buffer size.
3. Set the output rate.
4. Transmit the packets such that there is no overflow.
5. Repeat the process of transmission until all packets are transmitted. (Reject packets where its size is greater than the bucket size)
6. Stop

Program:

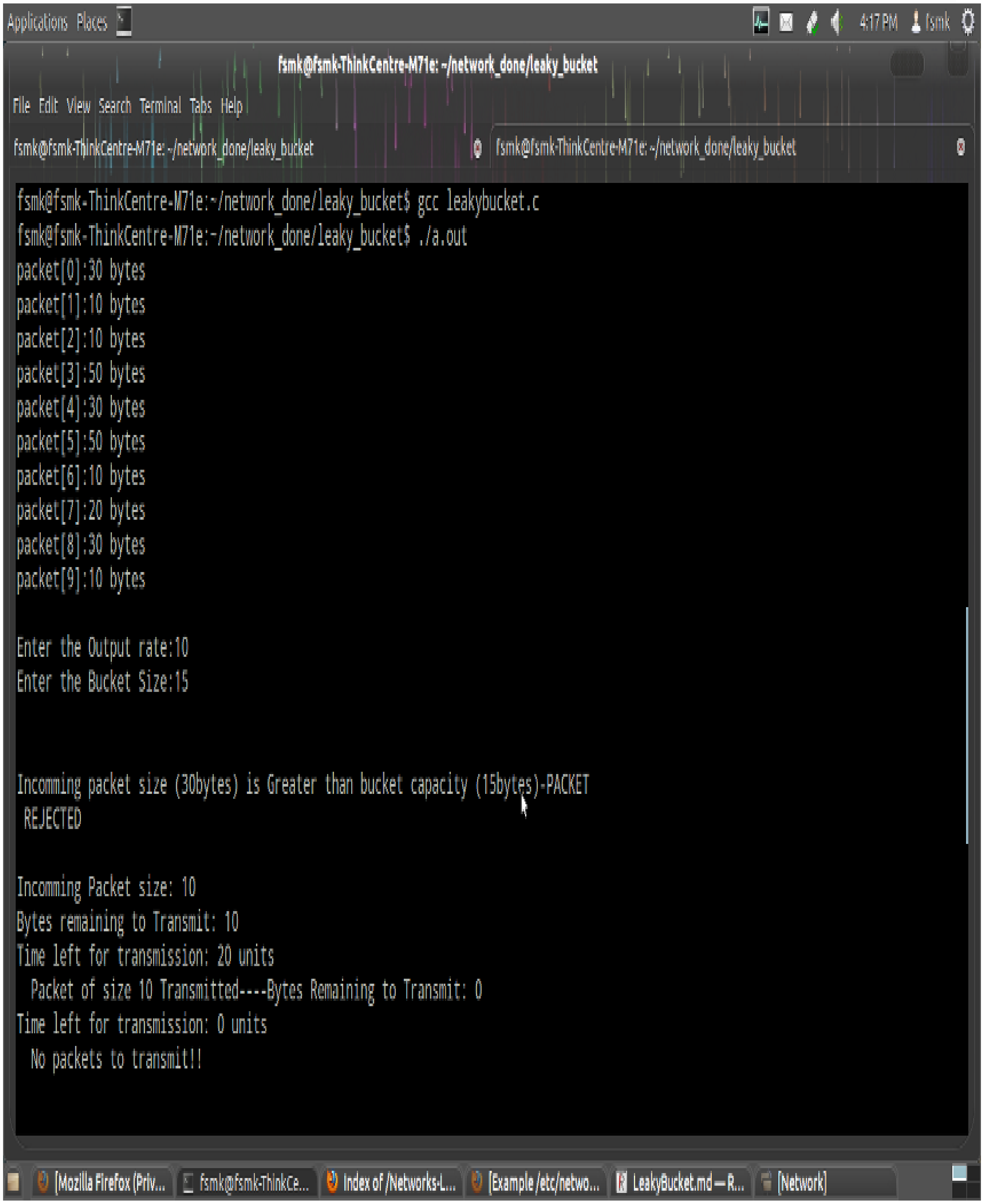
```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define NOF_PACKETS 10
int rand(int a)
{
    int rn = (random() % 10) % a;
    return  rn == 0 ? 1 : rn;
}
int main()
{
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0, p_sz, p_time, op;
    for(i = 0; i<NOF_PACKETS; ++i)
        packet_sz[i] = rand(6) * 10;
    for(i = 0; i<NOF_PACKETS; ++i)
        printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
    printf("\nEnter the Output rate:");
    scanf("%d", &o_rate);
    printf("Enter the Bucket Size:");
    scanf("%d", &b_size);
    for(i = 0; i<NOF_PACKETS; ++i)
    {
        if( (packet_sz[i] + p_sz_rm) > b_size)
        if(packet_sz[i] > b_size)/*compare the packet siz with bucket size*/
            printf("\n\nIncoming packet size (%dbytes) is Greater than bucket capacity
(%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
        else
            printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
    else
    {
        p_sz_rm += packet_sz[i];
        printf("\n\nIncoming Packet size: %d", packet_sz[i]);
        printf("\nBytes remaining to Transmit: %d", p_sz_rm);
        p_time = rand(4) * 10;
        printf("\nTime left for transmission: %d units", p_time);
        for(clk = 10; clk <= p_time; clk += 10)
        {
            sleep(1);
            if(p_sz_rm)
            {
                if(p_sz_rm <= o_rate)/*packet size remaining comparing with output rate*/
                    op = p_sz_rm, p_sz_rm = 0;
                else
                    op = o_rate, p_sz_rm -= o_rate;
                printf("\nPacket of size %d Transmitted", op);
                printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
            }
            else
            {
                printf("\nTime left for transmission: %d units", p_time-clk);
                printf("\nNo packets to transmit!!");
            }
        }
    }
}
}
```

Output:

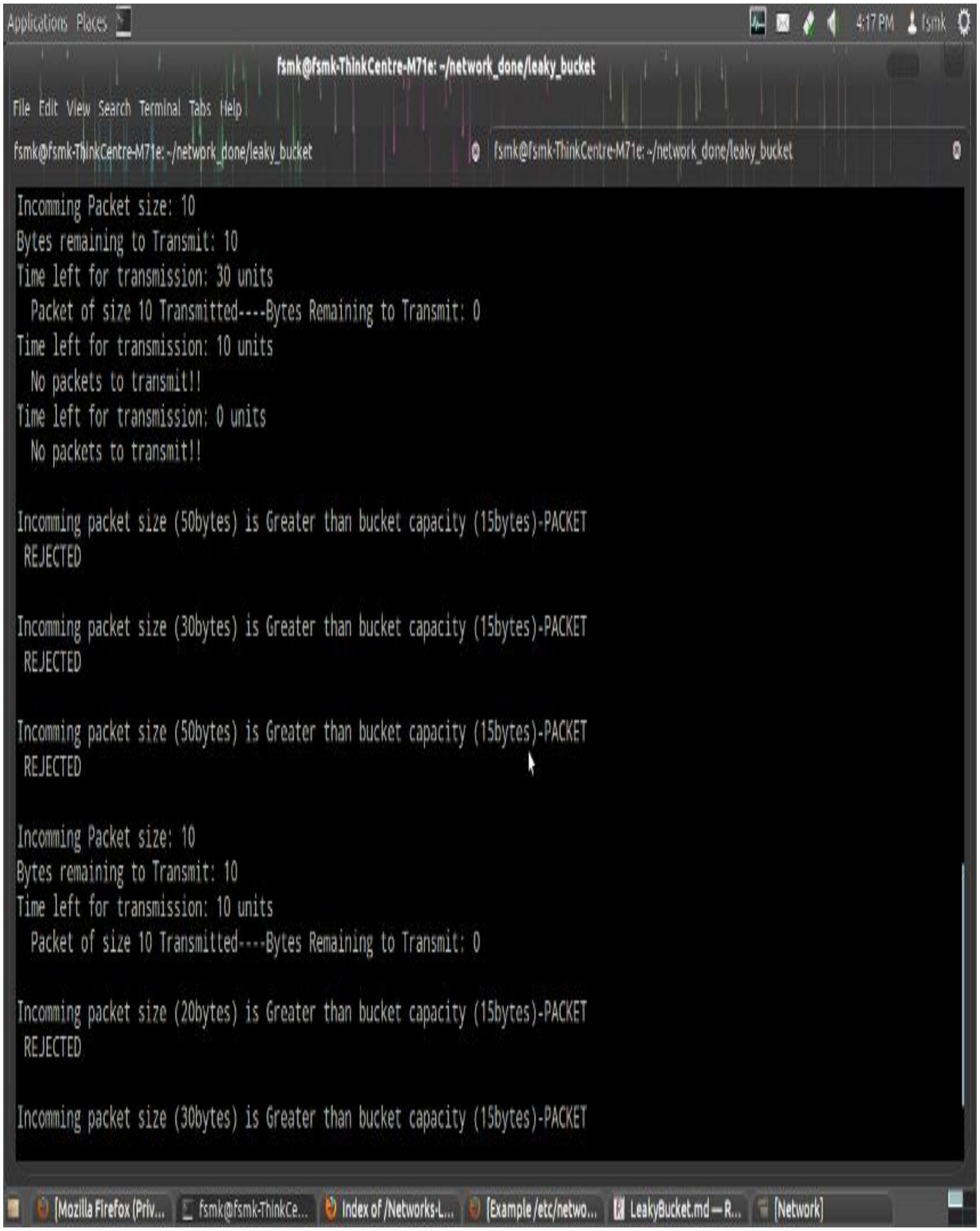
Commands for execution:-

- Open a terminal.
- Change directory to the file location.
- Run gcc *filename.c*
- If there are no errors, run ./a.out

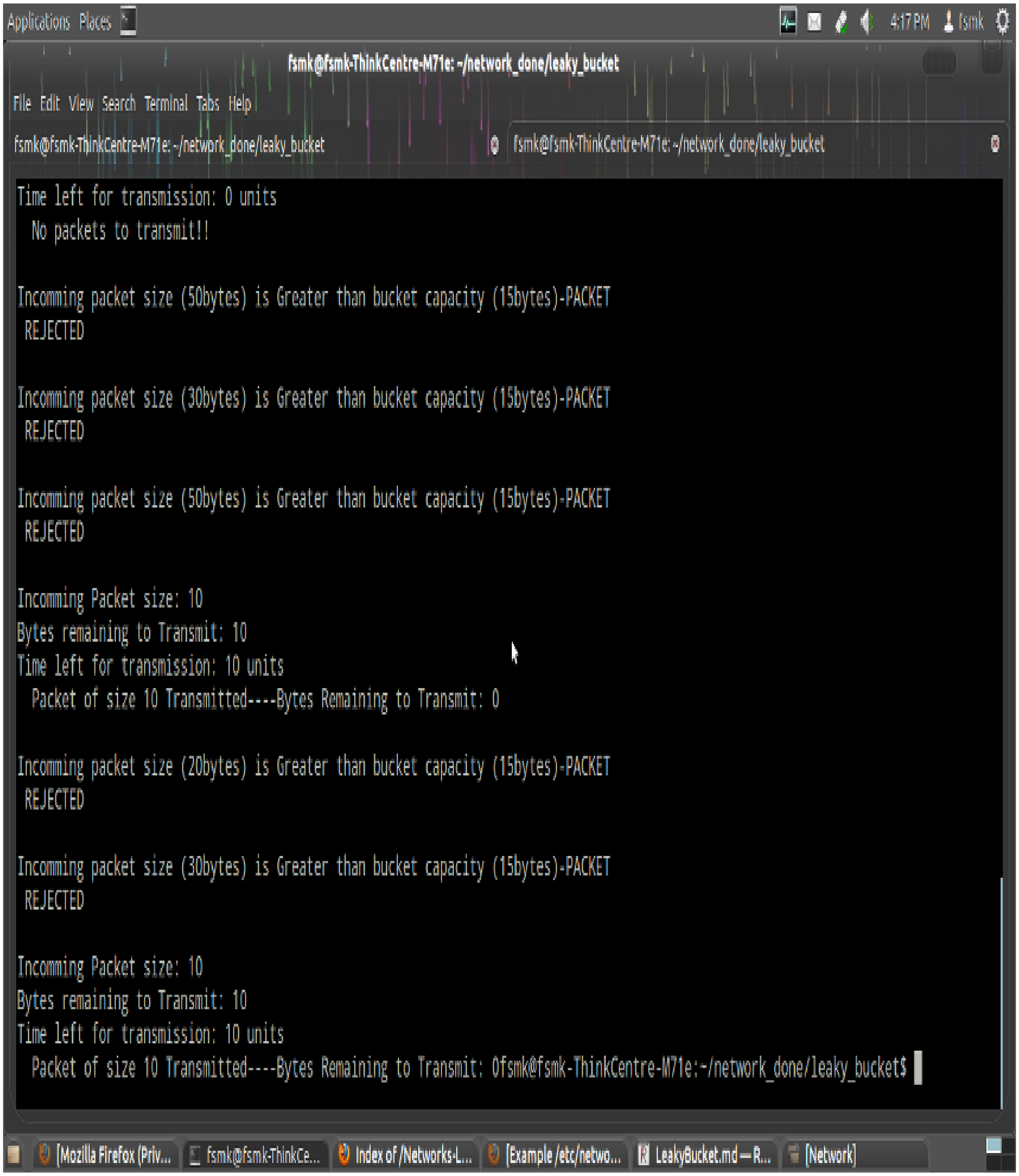
Screenshots:-


Applications Places
4:17 PM fsmk
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
File Edit View Search Terminal Tabs Help
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
[Mozilla Firefox (Priv...
fsmk@fsmk-ThinkCe...
Index of /Networks-L...
[Example /etc/netwo...
LeakyBucket.md — R...
[Network]

```
fsmk@fsmk-ThinkCentre-M71e:~/network_done/leaky_bucket$ gcc leakybucket.c
fsmk@fsmk-ThinkCentre-M71e:~/network_done/leaky_bucket$ ./a.out
packet[0]:30 bytes
packet[1]:10 bytes
packet[2]:10 bytes
packet[3]:50 bytes
packet[4]:30 bytes
packet[5]:50 bytes
packet[6]:10 bytes
packet[7]:20 bytes
packet[8]:30 bytes
packet[9]:10 bytes

Enter the Output rate:10
Enter the Bucket Size:15



Incomming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED


Incomming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 20 units
  Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
  No packets to transmit!!
```


Applications Places
4:17 PM fsmk
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
File Edit View Search Terminal Tabs Help
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
[Mozilla Firefox (Priv...
fsmk@fsmk-ThinkCe...
Index of /Networks-L...
[Example /etc/netwo...
LeakyBucket.md — R...
[Network]

```
Incomming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 30 units
  Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
  No packets to transmit!!
Time left for transmission: 0 units
  No packets to transmit!!

Incomming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
  Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0

Incomming packet size (20bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET
```



Applications Places
4:17 PM fsmk
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
File Edit View Search Terminal Tabs Help
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
[Mozilla Firefox (Priv...
fsmk@fsmk-ThinkCe...
Index of /Networks-L...
[Example /etc/netwo...
LeakyBucket.md — R...
[Network]

```
Time left for transmission: 0 units
  No packets to transmit!!

Incomming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
  Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0

Incomming packet size (20bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET
 REJECTED

Incomming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
  Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0fsmk@fsmk-ThinkCentre-M71e:~/network_done/leaky_bucket$
```

EXPERIMENT NO: 9

9. Write a program for frame sorting technique used in buffers.

```
#include<stdio.h>
#include<string.h>
#define FRAM_TXT_SIZ 3
#define MAX_NOF_FRAM 127
char str[FRAM_TXT_SIZ*MAX_NOF_FRAM];
struct frame // structure maintained to hold frames
{ char text[FRAM_TXT_SIZ];
int seq_no;
}fr[MAX_NOF_FRAM], shuf_ary[MAX_NOF_FRAM];
int assign_seq_no() //function which splits message
{ int k=0,i,j; //into frames and assigns sequence no
for(i=0; i < strlen(str); k++)
{ fr[k].seq_no = k;
for(j=0; j < FRAM_TXT_SIZ && str[i]!='\0'; j++)
fr[k].text[j] = str[i++];
}
printf("\nAfter assigning sequence numbers:\n");
for(i=0; i < k; i++)
printf("%d:%s ",i,fr[i].text);
return k; //k gives no of frames
}
void generate(int *random_ary, const int limit) //generate array of random nos
{ int r, i=0, j;
while(i < limit)
{ r = random() % limit;
for(j=0; j < i; j++)
if( random_ary[j] == r )
break;
if( i==j ) random_ary[i++] = r;
} }
void shuffle( const int no_frames ) // function shuffles the frames
```

```c
{
int i, k=0, random_ary[no_frames];
generate(random_ary, no_frames);
for(i=0; i < no_frames; i++)
shuf_ary[i] = fr[random_ary[i]];
printf("\n\nAFTER SHUFFLING:\n");
for(i=0; i < no_frames; i++)
printf("%d:%s ",shuf_ary[i].seq_no,shuf_ary[i].text);
}
void sort(const int no_frames) // sorts the frames
{
int i,j,flag=1;
struct frame hold;
for(i=0; i < no_frames-1 && flag==1; i++) // search for frames in sequence
{
flag=0;
for(j=0; j < no_frames-1-i; j++) //(based on seq no.) and display
if(shuf_ary[j].seq_no > shuf_ary[j+1].seq_no)
{
hold = shuf_ary[j];
shuf_ary[j] = shuf_ary[j+1];
shuf_ary[j+1] = hold;
flag=1;
}
}
}
int main()
{
int no_frames,i;
printf("Enter the message: ");
gets(str);
no_frames = assign_seq_no();
shuffle(no_frames);
sort(no_frames);
printf("\n\nAFTER SORTING\n");
for(i=0;i<no_frames;i++)
printf("%s",shuf_ary[i].text);
printf("\n\n");
}
```

OUTPUT C Program to Frame sorting technique used in buffers.:

```
[root@localhost nwcn]# ./a.out

Enter the message: Welcome To Mahatma Institute Technology

After assigning sequence numbers:

0:Wel 1:com 2:e T 3:o M 4:aha 5:tma 6: In 7:sti 8:tut 9:e o 10:f T 11:ech 12:nol 13:ogy

AFTER SHUFFLING:

1:com 4:cha 9:e o 5:rya 3:o A 10:f T 2:e T 6: In 11:ech 13:ogy 0:Wel 8:tut 12:nol 7:sti

AFTER SORTING
```