COMPUTER NETWORKS LAB SYLLABUS

III Year B.Tech. CSE - I Sem

OBJECTIVE:

To understand the functionalities of various layers of OSI model To understand the operating System functionalities

SYSTEM/ SOFTWARE REQUIREMENT:

Intel based desktop PCs LAN CONNECTED with minimum of 166 MHZ or faster processor with atleast 64 MB RAM and 100 MB free disk space

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

2. Implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.

3. Implement Dijkstra‘s algorithm to compute the Shortest path thru a graph.

4. Take an example subnet graph with weights indicating delay between nodes.Now obtain Routing table art each node using distance vector routing algorithm

5. Take an example subnet of hosts. Obtain broadcast tree for it.

6. Take a 64 bit playing text and encrypt the same using DES algorithm.

7. Write a program to break the above DES coding

8. Using RSA algorithm encrypt a text data and Decrypt the same.

COMPUTER NETWORKS LAB OBJECTIVES

1. Analyze the different layers in networks.
2. Define, use, and differentiate such concepts as OSI-ISO,TCP/IP.
3. How to send bits from physical layer to data link layer
4. Sending frames from data link layer to Network layer
5. Different algorithms in Network layer
6. Analyze the presentation layer, application layer
7. They can understand how the data transferred from source to destination
8. They can come to know that how the routing algorithms worked out in network layer

Recommended System/Software Requirements

Intel based desktop PC with minimum of 2.6GHZ or faster processor with at least 1 GB RAM and 40 GB free disk space and LAN connected.

Operating system: Flavor of any WINDOWS or LINUX.

Software: Turbo C, C++.Geany

INTRODUCTION TO CN

The purpose of this is to acquaint the students with an overview of the Computer Networks from the perspective how the information is transferred from source to destination and different layers in networks. This course provides a basis for u. They can understand how the data transferred from source to destination. They can come to know that how the routing algorithms worked out in network layer understanding the networking techniques that can take place in computer. A computer network is made of two distinct subsets of components Distributed applications are programs running on interconnected computers; a web server, a remote login server, an e-mail exchanger are examples. This is the visible part of what people call “the Internet”. In this lecture we will study the simplest aspects of distributed applications. More sophisticated aspects are the object of lectures called “Distributed Systems” and “Information Systems”. The network infrastructure is the collection of systems which are required for the interconnection of computers running the distributed applications. It is the main focus of this lecture. The network infrastructure problem has itself two aspects: Distance: interconnect remote systems that are too far apart for a direct cable connection Meshing: interconnect systems together; even in the case of systems close to each other, it is not possible in non-trivial cases to put cables from all systems to all systems (combinatorial explosion, cable *salad* management problem s etc.).

LAB CODE

1. Students should report to the concerned lab as per the time table.

2. Students who turn up late to the labs will in no case be permitted to do the program schedule for the day.

3. After completion of the program, certification of the concerned staff in-charge in the observation book is necessary.

4. Student should bring a notebook of 100 pages and should enter the readings observations into the notebook while performing the experiment.

5. The record of observations along with the detailed experimental procedure of the experiment in the immediate last session should be submitted and certified staff member in-charge.
6. Not more than 3-students in a group are permitted to perform the experiment on the set.
7. The group-wise division made in the beginning should be adhered to and no mix up of students among different groups will be permitted.

8. The components required pertaining to the experiment should be collected from stores in-charge after duly filling in the requisition form.

9. When the experiment is completed, should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.

10. Any damage of the equipment or burn-out components will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year.
11. Students should be present in the labs for total scheduled duration.
12. Students are required to prepare thoroughly to perform the experiment before coming to laboratory.

INDEX

EXPERIMENT NO: 1. (a)

NAME OF THE EXPERIMENT: Bit Stuffing.

AIM: Implement the data link layer framing methods such as and bit stuffing.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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. When ever the senders data link layer encounters five consecutive one's in the data, it automatically stuffs a 0 bit into the out going bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the out going character stream before DLE in the data

ALGORITHM:

Begin

Step 1: Read frame length n

Step 2: Repeat step (3 to 4) until i<n(**: R**ead values in to the input frame (0's and 1's) i.e.

Step 3: initialize I i=0;

Step 4: read a[i] and increment i

Step 5: Initialize i=0, j=0,count =0

Step 6: repeat step (7 to 22) until i<n

Step 7: If a[i] == 1 then

Step 8: b[j] = a[i]

Step 9: Repeat step (10 to 18) until (a[k] =1 and k<n and count <5)

Step 10: Initialize k=i+1;

Step 11: Increment j and b[j]= a[k];

Step 12: Increment count ;

Step 13: if count =5 then

Step 14: increment j,

Step 15: b[j] =0

Step 16: end if

Step 17: i=k;

Step 18: increment k

Step 19: else

Step 20: b[j] = a[i]

Step 21: end if

Step 22: increment I and j

Step 23: print the frame after bit stuffing

Step 24: repeat step (25 to 26) until i< j

Step 25: print b[i]

Step 26: increment i

End

SOURCE CODE:

```
// BIT Stuffing program
#include<stdio.h>
#include<string.h>
void main()
{
int a[20],b[30],i,j,k,count,n;
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]);
}
```

OUTPUT:

Enter frame length:5

Enter input frame (0's & 1's only):

1

1

1

1

1

After stuffing the frame is:111110

(program exited with code: 6)

Press return to continue

VIVA QUESTIONS:

1. What is bit stuffing?

2. What is the use of bit stuffing?

3. with bit stuffing the boundary b/w 2 frames can be unambiguously recognized by ----------

4. ---------- is analogous to character stuffing

5. Each frame begins and ends with a special bit pattern 01111110 called ---

---

6. The senders data link layer encounters ----------no of 1's consecutively

EXPERIMENT NO: 1. (b)

NAME OF THE EXPERIMENT: Character Stuffing.

AIM: Implement the data link layer framing methods such as character, character stuffing.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-**RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and 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 layer on the receiving end removes the DLE before the data are given to the network layer. This technique is called character stuffing

ALGORITHM:

Begin

Step 1: Initialize I and j as 0

Step 2: Declare n and pos as integer and a[20],b[50],ch as character

Step 3: read the string a

Step 4: find the length of the string n, i.e n-strlen(a)

Step 5: read the position, pos

Step 6: if pos > n then

Step 7: print invalid position and read again the position, pos

Step 8: end if

Step 9: read the character, ch

Step 10: Initialize the array b , b[0…5] as ’d’, ’l’, ’e’, ’s’ ,’t’ ,’x’
respectively

Step 11: j=6;

Step 12: Repeat step[(13to22) until i<n

Step 13: if i==pos-1 then

Step 14: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’, ‘e’ ,’ch, ’d’, ‘l’,‘e’
respectively

Step 15: increment j by 7, i.e j=j+7

Step 16: end if

Step 17: if a[i]=='d' and a[i+1]=='l' and a[i+2]=='e' then

Step 18: initialize array b, b[13…15]='d', 'l', 'e' respectively

Step 19: increment j by 3, i.e j=j+3

Step 20: end if

Step 21: b[j]=a[i]

Step 22: increment I and j;

Step 23: initialize b array,b[j],b[j+1]…b[j+6] as'd', 'l','e' ,'e','t', 'x','\0' respectively

Step 24: print frame after stiuffing

Step 25: print b

End

SOURCE CODE:

```
//PROGRAM FOR CHARACTER STUFFING
#include<stdio.h>
#include<string.h>
#include<process.h>
void main()
{
int i=0,j=0,n,pos;
char a[20],b[50],ch;
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);
}
```

OUTPUT:

```
enter string
MLRITM
enter position
2
enter the character
frame after stuffing:
dlestxMdldleLRITMdleetx
------------------
(program exited with code: 0)
Press return to continue
```

VIVA QUESTIONS:

1. What is character stuffing?
2. What is the use of character stuffing?
3. _____ is analogous to bit stuffing.
4. _____ are the delimiters for character stuffing
5. Expand DLE STX_____
6. Expand DLE ETX_____

EXPERIMENT NO: 2.

NAME OF THE EXPERIMENT: Cyclic Redundancy Check.

AIM: Implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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 some times known as Cyclic Redundancy Code

ALGORITHM/FLOWCHART:

Begin

Step 1: Declare I,j,fr[8],dupfr[11],recfr[11],tlen,flag,gen[4],genl,frl, rem[4] as integer

Step 2: initialize frl=8 and genl=4

Step 3: initialize i=0

Step 4: Repeat step(5to7) until i<frl

Step 5: read fr[i]

Step 6: dupfr[i]=fr[i]

Step 7: increment i

Step 8: initialize i=0

Step 9: repeat step(10to11) until i<genl

Step 10: read gen[i]

Step 11: increment i

Step 12: tlen=frl+genl-1

Step 13: initialize i=frl

Step 14: Repeat step(15to16) until i<tlen

Step 15: dupfr[i]=0

Step 16: increment i

Step 17: call the function remainder(dupfr)

Step 18: initialize i=0

Step 19: repeat step(20 to 21) until j<genl

Step 20: recfr[i]=rem[j]

Step 21: increment I and j

Step 22: call the function remainder(dupfr)

Step 23: initialize flag=0 and i=0

Step 24: Repeat step(25to28) until i<4

Step 25: if rem[i]!=0 then

Step 26: increment flag

Step 27: end if

Step 28: increment i

Step 29: if flag=0 then

Step 25: print frame received correctly

Step 25: else

Step 25: print the received frame is wrong

End

Function: Remainder(int fr[])

Begin

Step 1: Declare k,k1,I,j as integer

Step 2: initialize k=0;

Step 3: repeat step(4 to 14) until k< frl

Step 4: if ((fr[k] == 1) then

Step 5: k1=k

Step 6: initialize i=0, j=k

Step 7: repeat step(8 to 9) until i< genl

Step 8: rem[i] =fr[j] exponential gen[i]

Step 9: increment I and j

Step 10: initialize I = 0

Step 11: repeat step(12to13) until I <genl

Step 12: fr[k1] = rem[i]

Step 13: increment k1 and i

Step 14: end if

End

SOURCE CODE:

```
//PROGRAM FOR CYCLIC REDUNDENCY CHECK
#include<stdio.h>
int gen[4],genl,frl,rem[4];
void main()
{
int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;
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");
}
}
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:MLRITM
enter generator:frame received correctly
------------------
(program exited with code: 24)
Press return to continue
```

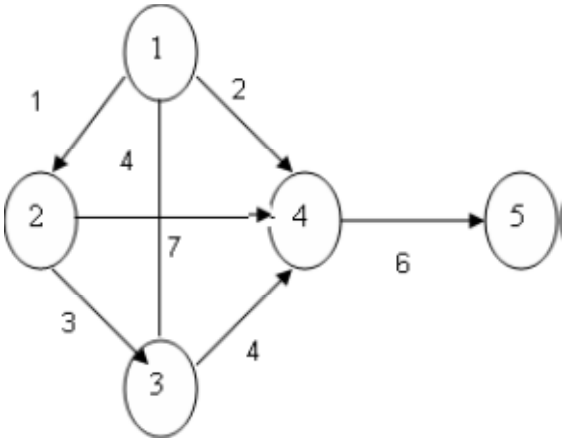
VIVA QUESTIONS:

1. What is CRC?
2. What is the use of CRC?
3. Name the CRC standards
4. Define checksum?
5. Define generator polynomial?
6. Polynomial arithmetic is done by_____

XPERIMENT NO: 3

NAME OF THE EXPERIMENT: Shortest Path.

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



HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

ALGORITHM/FLOWCHART:

Begin

Step1: Declare array path [5] [5], min, a [5][5], index, t[5];

Step2: Declare and initialize st=1,ed=5

Step 3: Declare variables i, j, stp, p, edp

Step 4: print “enter the cost “

Step 5: i=1

Step 6: Repeat step (7 to 11) until (i<=5)

Step 7: j=1

Step 8: repeat step (9 to 10) until (j<=5)

Step 9: Read a[i] [j]

Step 10: increment j

Step 11: increment i

Step 12: print “Enter the path”

Step 13: read p

Step 14: print “Enter possible paths”

Step 15: i=1

Step 16: repeat step(17 to 21) until (i<=p)

Step 17: j=1

Step 18: repeat step(19 to 20) until (i<=5)

Step 19: read path[i][j]

Step 20: increment j

Step 21: increment i

Step 22: j=1

Step 23: repeat step(24 to 34) until(i<=p)

Step 24: t[i]=0

Step 25: stp=st

Step 26: j=1

Step 27: repeat step(26 to 34) until(j<=5)

Step 28: edp=path[i][j+1]

Step 29: t[i]= [ti]+a[stp][edp]

Step 30: if (edp==ed) then

Step 31: break;

Step 32: else

Step 33: stp=edp

Step 34: end if

Step 35: min=t[st]

Step 36: index=st

Step 37: repeat step(38 to 41) until (i<=p)

Step 38: min>t[i]

Step 39: min=t[i]

Step 40: index=i

Step 41: end if

Step 42: print” minimum cost” min

Step 43: print" minimum cost pth"

Step 44: repeat step(45 to 48) until (i<=5)

Step 45: print path[index][i]

Step 46: if(path[idex][i]==ed) then

Step 47: break

Step 48: end if

End

SOURCE CODE:

```
//*********************************
//5 .PROGRAM FOR FINDING SHORTEST //PATH FOR A GIVEN GRAPH
//*********************************
#include<stdio.h>
void main()
{
int path[5][5],i,j,min,a[5][5],p,st=1,ed=5,stp,edp,t[5],index;
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;
}
}
```

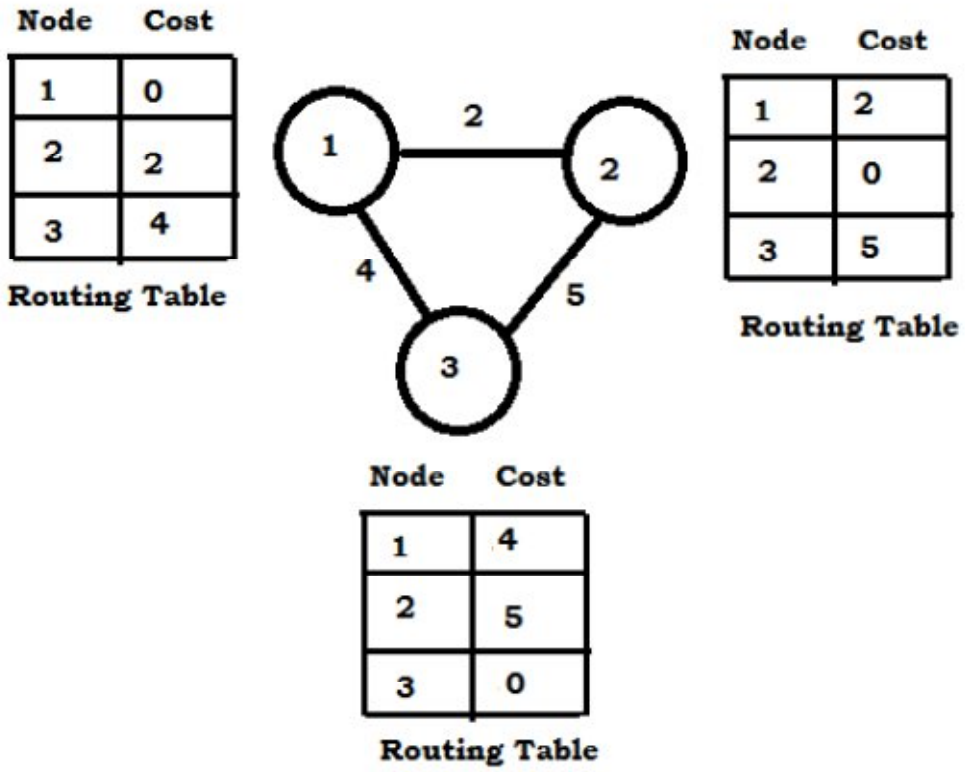
OUTPUT:

```
enter the cost matrix
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
enter the paths
2
enter possible paths
1 2 3 4 5
1 2 3 4 5
minimum cost 14
```

EXPERIMENT NO: 4

NAME OF THE EXPERIMENT: Distance Vector routing.

AIM: Obtain Routing table at each node using distance vector routing algorithm for a given subnet.



HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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.

ALGORITHM:

Begin

Step1: Create struct node unsigned dist[20],unsigned from[20],rt[10]

Step2: initialize int dmat[20][20], n,i,j,k,count=0,

Step3: write "the number of nodes "

Step4: read the number of nodes "n"

Step5: write" the cost matrix :"

Step6: intialize i=0

Step7: repeat until i<n

Step8: increment i

Step9: initialize j=0

Step10: repeat Step(10-16)until j<n

Step11: increment j

Step12:read dmat[i][j]

Step13:intialize dmat[i][j]=0

Step14:intialize rt[i].dist[j]=dmat[i][j]

Step15:intialize rt[i].from[j]=j

Step16:end

Step17:start do loop Step (17-33)until

Step18:intilialize count =0

Step19:initialize i=0

Step20:repeat until i<n

Step21:increment i

Step22:initialize j=0

Step23:repeat until j<n

Step24:increment j

Step25:initialize k=0

Step26:repeat until k<n

Step27:increment k

Step28:if repeat Step(28-32) until rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]

Step29:intialize rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]

Step30:intialize rt[i].from[j]=k;

Step31:increment count

Step32:end if

Step33:end do stmt

Step34:while (count!=0)

Step35:initialize i=0

Step36:repeat Steps(36-44)until i<n

Step37:increment i

Step38:write ' state values for router',i+1

Step39:initialize j=0

Step40:repeat Steps (40-43)until j<n

Step41:increment j

Step42:write 'node %d via %d distance % ',j+1,rt[i].from[j]+1,rt[i].dist[j]

Step43:end

Step44:end

end

SOURCE CODE:

```
#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("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 : 2
Enter the cost matrix :
1 2
1 2
State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance2
State value for router 2 is
node 1 via 1 Distance1
node 2 via 2 Distance0
```

VIVA QUESTIONS:

1. What is routing
2. What is best algorithm among all routing algorithms?
3. What is static routing?
4. Difference between static and dynamic
5. How distance vector routing works
6. What is optimality principle?

EXPERIMENT NO: 5

NAME OF THE EXPERIMENT: Broadcast Tree.

AIM: Implement broadcast tree for a given subnet of hosts

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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.

ALGORITHM/FLOWCHART:

step 1: declare variable as int p,q,u,v,n;

step 2: Initialize min=99,mincost=0;

step 3: declare variable as int t[50][2],i,j;

step 4: declare variable as int parent[50],edge[50][50];

step 5: Begin

step 6: write "Enter the number of nodes"

step 7: read "n"

step 8: Initialize i=0

step 9: repeat step(10-12) until i<n

step10: increment i

step11: write"65+i"

step12: Initialize parent[i]=-1

step13:wite "\n"

step14: Initialize i=0

step15: repeat step(15-21) until i<n

step16: increment i

step17: write"65+i"

step18: Initialize j=0

step19: repeat until j<n

step20: increment j

step21: read edge[i][j]

step22: Initialize i=0

step23: repeat step(23-43) until i<n

step24: increment i

step25: Initialize j=0

step26: repeat until j<n

step27: increment j

step28: if'edge[i]j]!=99

step29: if'min>edge[i][j] repeat step (29-32)

step30: intialize min=edge[i][j]

step31: intialize u=i

step32: intialize v=j

step33: calling function p=find(u);

step34: calling function q=find(v);

step35: if'P!=q repeat steps(35-39)

step36: intialize t[i][0]=u

step37: intialize t[i][1]=v

step38: initialize mincost=mincost+edge[u][v]

step39: call function sunion(p,q)

step40: else repeat steps(40-42)

step41: Intialize t[i][0]=-1;

step42: Intialize t[i][1]=-1;

step43: intialize min=99;

step44; write"Minimum cost is %d\n Minimum spanning tree is",mincost

step45: Initialize i=0

step46: repeat until i<n

step47: increment i

step48: if't[i][0]!=-1 && t[i][1]!=-1'repeat step(48-50)

step49: write "%c %c %d", 65+t[i][0], 65+t[i][1], edge[t[i][0]][t[i][1]]

step50: write"\n"

step51: end

step52: called function sunion(int l,int m) repeat step(51-52)

step53: intialize parent[l]=m

step54: called function find(int l) repeat step(53-56)

step55: if parent([l]>0)

step56: initialize l=parent

step57: return l

SOURCE CODE:

```
// Write a ‘c’ program for Broadcast tree from subnet of host
#include<stdio.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");
}
}
sunion(int l,int m)
{
parent[l]=m;
}
find(int l)
{
if(parent[l]>0)
l=parent[l];
return l;
}
```

OUTPUT:

```
Enter the number of nodes3
A       B       C
A1 2 3 4
B1 2 3 4
C4 5 6 7
Minimum cost is 3
Minimum spanning tree is
C A 3
```

VIVA QUESTIONS:

1. What is spanning tree
2. What is broad cast tree?
3. What are the advantages of broad cast tree?
4. Where we should use the broad cast tree
5. What is flooding?
6. What is the subnet?

EXPERIMENT NO: 6

NAME OF THE EXPERIMENT: encrypting DES.

AIM: Take a 64 bit playing text and encrypt the same using DES algorithm.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

Data encryption standard was widely adopted by the industry in security products. Plain text is encrypted in blocks of 64 bits yielding 64 bits of cipher text. The 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 transposition. The remaining stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption

ALGORITHM/FLOWCHART:

Begin

Step1: Initialize as int i,ch,lp;

Step2: Initialize as char cipher[50],plain[50];

Step3: Initialize as char key[50];

Step4: while(1) repeat steps(4-36)

Step5: write "\n-----MENU-----\n"

Step6: write "\n1:Data Encryption\t\n\n2:Data Decryption\t\n\n3:Exit"

Step7: write ("\n\nEnter your choice:"

Step8: read"%d",&ch

Step9: stament switch(ch) repeat steps(9-35)

case 1:

step10: read "\nData Encryption"

step11:read ("\nEnter the plain text:"

step12: fflush(stdin)

step13 : gets(plain)

step14: write "\nEnter the encryption key:"

step15: gets(key)

step16: lp=strlen(key)

step17: Initialize i=0

step18: repeat until plain[i]!='\0'

step19: increment i

step20: initialize cipher[i]=plain[i]^lp

step21: initialize cipher[i]='\0';

step22: write "\nThe encrypted text is:"

step23: puts(cipher)

step24: break

case 2:

step25: write"\nData decryption"

step26: Initialize i=0

step27: repeat until plain[i]!='\0'

step28: increment i

step29: initialize plain[i]=cipher[i]^lp

step30:write"\nDecrypted text is:"

step32: puts(plain)

step33:break

case 3:

step34:exit(0);

step35: end switch stmt

step36: end while(1)stmt

End

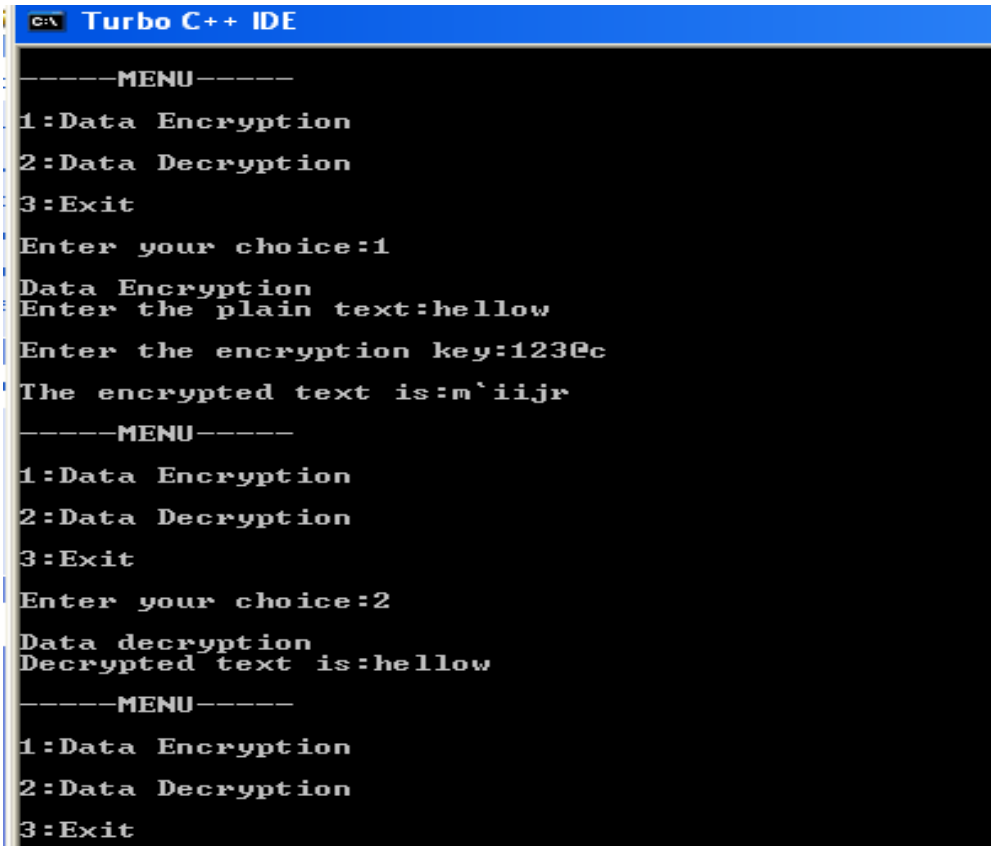
SOURCE CODE:

/*Take a 64 bit playing text and encrypt the same using DES algorithm */

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
int i,ch,lp;
char cipher[50],plain[50];
char key[50];
clrscr();
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;
case 3: exit(0);
}
}
getch();
}
```

OUTPUT:



VIVA QUESTIONS:

1. Expand DES_____
2. What is cipher text?
3. What is plain text?
4. Define public key?
5. Define encryption?
6. Substitutions are performed by_____boxes

EXPERIMENT NO: 7.

NAME OF THE EXPERIMENT: Decrypting DES.

AIM: Write a program to break the above DES coding

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

Data encryption standard was widely adopted by the industry in security products. Plain text is encrypted in blocks of 64 bits yielding 64 bits of cipher text. The 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 transposition. The remaining stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption

ALGORITHM/FLOWCHART:

Begin

Step1: Initialize char pwd[20];

Step2: Initialize char alpha[26]="abcdefghijklmnopqrstuvwxyz";

Step3: Initialize int num[20],i,n,key;

Step4: Write "Enter the password:"

Step5: Read pwd

Step6: Initialize n=strlen(pwd

Step7: Initialize i=0

Step8: Repeat until i<n

Step9: Increment i

Step10: Intialize num[i]=toascii(tolower(pwd[i]))-'a'

Step11: Write "Enter the key:"

Step12: Read key

Step13: Initialize i=0

Step14: Repeat until i<n

Step15: Increment i

Step16: Initialize num[i]=(num[i]+key)%26

Step17: Initialize i=0

Step18: Repeat until i<n

Step19: Increment i

Step20: Intialize pwd[i]=alpha[num[i]]

Step21: Write “The key is:%d",key

Step22: Write “Encrypted text is:%s",pwd

Step23: Initialize i=0

Step24: Repeat steps(24-29)until i<n

Step25: Increment i

Step26: Intialize num[i]=(num[i]-key)%26

Step27: if ‘ num[i]<0 ’

Step28: Initialize num[i]=26+num[i]

Step29: Intialize pwd[i]=alpha[num[i]]

Step30: Write “Decrypted text is:%s",pwd

End

SOURCE CODE:

/*Write a program to break the above DES coding*/

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
void main()
{
char pwd[20];
char alpha[26]="abcdefghijklmnopqrstuvwxyz";
int num[20],i,n,key;
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);
}
```

OUTPUT:

Enter the password:appu

Enter the key:1202

The key is:1202

Encrypted text is:gvva

Decrypted text is:appu

VIVA QUESTIONS:

1. Define decryption?
2. What is private key?
3. Transpositions are performed in _____box
4. What is cipher feedback mode?
5. Define product cipher?
6. What is DES chaining?

EXPERIMENT NO: 8

NAME OF THE EXPERIMENT: RSA.

AIM: Using RSA algorithm encrypt a text data and Decrypt the same.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-**RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

RSA method is based on some principles from number theory. In encryption process divide the plain text into blocks, so that each plain text message p falls in the interval 0<p<n this can be done by grouping the plain text into blocks of k bits. Where k is the largest integer for which 2 power k <n is true. The security of this method is based on the difficulty of factoring large numbers. The encryption and decryption functions are inverses

ALGORITHM/FLOWCHART:

Step1: Start

Step2: Initialize variables as int a,b,i,j,t,x,n,k=0,flag=0,prime[100]

Step3: Initialize variables as char m[20],pp[20]

Step4: Initialize variables as float p[20],c[20]

Step5: Initialize variables as double e,d;

Step6: Initialize i=0

Step7: Repeat step(7-16) until i<50

Step8: Increment i

Step9: Initialize flag=0

Step10: Initialize j=2

Step11: Repeat until j<i/2

Step12: if ‘ i%j == 0 ’ repeat until(12-14)

Step13: Initialize flag=1

Step14: break

Step15: if ’ (flag==0) ‘

Step16: Initialize prime[k++]=i

Step17: Initialize a=prime[k-1]

Step18: Initialize b=prime[k-2]

Step19: Initialize n=a*b

Step20: Initialize t=(a-1)*(b-1)

Step21: Initialize e=(double)prime[2]

Step22: Initialize d=1/(float)e

Step23: write "\nKey of encryption is:%lf\n",d

Step24: write"\nEnter plain the text:"

Step25: read m

Step26: Intialize x=strlen(m)

Step27: write"\nDecryption status From Source to Destination:\n"

Step28: write"\nSource\t->---------------------------------<-destination\n"

Step29: write"\nChar\tnumeric\tcipher\t\tnumeric\t\tchar \n"

Step30: write"\n**\n"

Step31: write"\n"

Step32: Intialize i=0

Step33: repeat steps(33-46) until i<x

Step34: Increment i

Step35: write "%c",m[i]

Step36: write"\t%d",m[i]-97

Step37; Intialize c[i]=pow(m[i]-97,(float)e)

Step38: Initialize c[i]=fmod(c[i],(float)n)

Step39: write "\t%f",c[i]

Step40: Intialize p[i]=pow(c[i],(float)d);

Step41: Intialize p[i]=fmod(p[i],(float)n);

Step42: write "\t%f",p[i]

Step43: Intialize pp[i]=p[i]+97

Step44: write "\t%c\n",pp[i]

Step45: write "\n**\n"

Step46: write "\n"

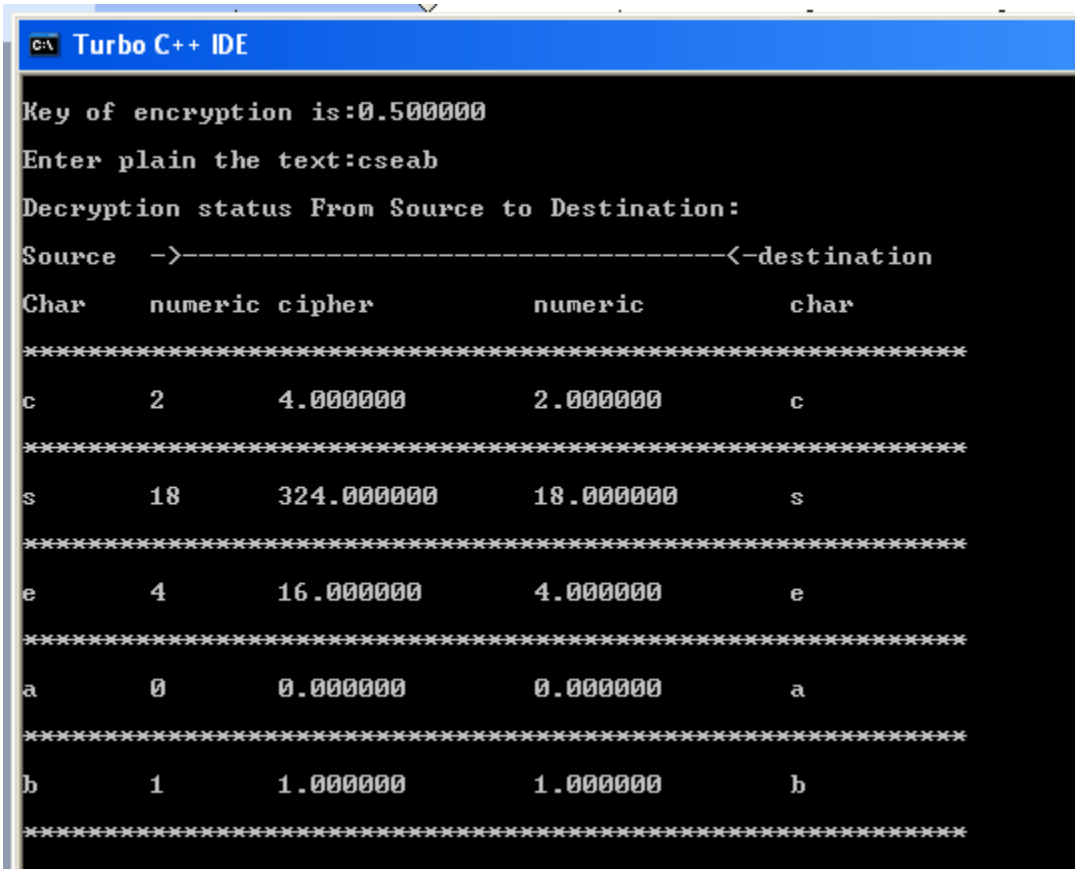
Step 47 end

SOURCE CODE:

```
/*Using RSA algorithm encrypt a text data and Decrypt the same*/
#include<stdio.h>
#include<ctype.h>
#include<math.h>
#include<string.h>
void main()
{
int a,b,i,j,t,x,n,k=0,flag=0,prime[100];
char m[20],pp[20];
float p[20],c[20];
double e,d;
for(i=0;i<50;i++)
{
flag=0;
for(j=2;j<i/2;j++)
if(i%j==0)
{
flag=1;
break;
}
if(flag==0)
prime[k++]=i;
}
a=prime[k-1];
b=prime[k-2];
n=a*b;
t=(a-1)*(b-1);
e=(double)prime[2];
d=1/(float)e;
```

```
printf("\nKey of encryption is:%lf\n",d);
printf("\nEnter plain the text:");
scanf("%s",&m);
x=strlen(m);
printf("\nDecryption status From Source to Destination:\n");
printf("\nSource\t->---------------------------------<-destination\n");
printf("\nChar\tnumeric\tcipher\t\tnumeric\t\tchar \n");
printf("\n************************************************************\n");
printf("\n");
for(i=0;i<x;i++)
{
printf("%c",m[i]);
printf("\t%d",m[i]-97);
c[i]=pow(m[i]-97,(float)e);
c[i]=fmod(c[i],(float)n);
printf("\t%f",c[i]);
p[i]=pow(c[i],(float)d);
p[i]=fmod(p[i],(float)n);
printf("\t%f",p[i]);
pp[i]=p[i]+97;
printf("\t%c\n",pp[i]);
printf("\n************************************************************\n");
printf("\n");
}
}
```

OUTPUT:



VIVA QUESTIONS:

1. Expand RSA_____
2. What is encryption & decryption in RSA
3. To encrypt a message P, Compute C=_____
4. To decrypt C, Compute P=_____
5. Define cryptography?
6. _____ systems use public key cryptography