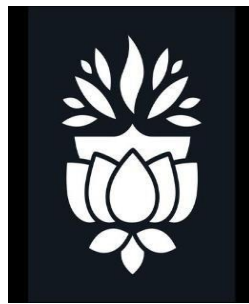


## **COMPUTER NETWORKS LAB**

# **LAB MANUAL**



**Department of Computer Science and Engineering**

**(AIML & DS)**

**R18-JNTUH**

**(2021-2022)**

**Faculty In charge:**

**Mr. MOHD FAISAL**

**Ms. ROQIA TABASSUM**

**(Asst. Professor in CSE)**

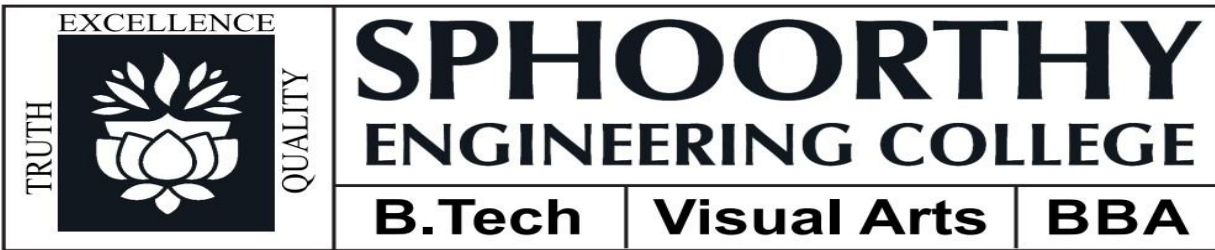
**Head of Dept.:**

**Mr. P RAMMOHAN RAO**

## Department of Computer Science and Engineering

### COMPUTER NETWORKS LAB ACADEMIC YEAR 2022-2023

<b>Lab Name /Course Name</b>	<b>COMPUTER NETWORKS LAB</b>
<b>Subject Code / Course Code</b>	
<b>Year &amp;Semester</b>	<b>III YEAR/I SEM</b>
<b>Branch</b>	<b>CSE – AIML &amp; DS</b>
<b>Name of the Faculty</b>	<b>Mr. MOHD FAISAL</b> <b>Ms. ROQIA TABASSUM</b>
<b>Designation</b>	<b>Assistant Professor</b>



## **VISION AND MISSION OF THE DEPARTMENT**

### **VISION:**

To create identity in Computer Science and Engineering with cutting edge technologies, catering ethical solutions for societal challenges.

### **MISSION:**

- To impart technical education with certified skill set to address societal challenges.
- To cultivate work ethics and communication skills in students for enriched work culture.
- To encourage community development programs for training personnel to fulfil the needs of society.

## LAB DESCRIPTION

Course Title	COMPUTER NETWORKS LAB			
Course Code				
Regulation	R18			
Course Structure	Lectures	Tutorials	Practical	Credits
	0	0	3	1.5
Course Faculty	<b>Mr. MOHD FAISAL</b> <b>Ms. ROQIA TABASSUM</b>			

### OVERVIEW:

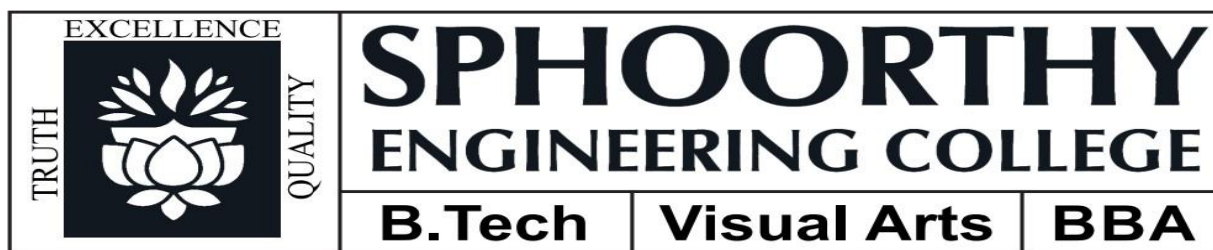
To understand the functionalities of various layers of OSI model, to understand the operating System functionalities

### PREREQUISITES:

1. Prior knowledge of C and C++ Programming.
2. Prior knowledge about NS2 Simulator.

### COURSE OBJECTIVES:

- To understand the working principle of various communication protocols.
- To understand the network simulator environment and visualize a network topology and observe its performance.
- To analyze the traffic flow and the contents of protocol frames.



### COURSE OUTCOME (CO):

S.NO	Course Outcomes (CO)	Blooms Taxonomy Level
After completing this course the student must demonstrate the knowledge and ability to:		
CO1	Implement data link layer farming methods.	L2: Understand
CO2	Analyze error detection and error correction codes.	L3: Apply
CO3	Implement and analyze routing and congestion issues in network design.	L2: Understand
CO4	Implement Encoding and Decoding techniques used in presentation layer.	L2: Understand
CO5	To be able to work with different network tools.	L3: Apply

### PROGRAM OUTCOME (PO):

PO1:	<b>Engineering knowledge:</b> Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2:	<b>Problem analysis:</b> Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3:	<b>Design/development of solutions:</b> Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4:	<b>Conduct investigations of complex problems:</b> Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

<b>PO5:</b>	<b>Modern tool usage:</b> Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
<b>PO6:</b>	<b>The engineer and society:</b> Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
<b>PO7:</b>	<b>Environment and sustainability:</b> Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
<b>PO8:</b>	<b>Ethics:</b> Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
<b>PO9:</b>	<b>Individual and team work:</b> Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
<b>PO10:</b>	<b>Communication:</b> Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
<b>PO11:</b>	<b>Project management and finance:</b> Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
<b>PO12:</b>	<b>Life-long learning:</b> Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

#### PROGRAM SPECIFIC OUTCOMES (PSO):

<b>PSO1</b>	<b>Software Development and Research Ability:</b> Ability to understand the structure and development methodologies of software systems. Possess professional skills and
-------------	--

	knowledge of software design process. Familiarity and practical competence with a broad range of programming language and open source platforms. Use knowledge in various domains to identify research gaps and hence to provide solution to new ideas and innovations.
<b>PSO2</b>	<b>Foundation of mathematical concepts:</b> Ability to apply the acquired knowledge of basic skills, principles of computing, mathematical foundations, algorithmic principles, modeling and design of computer- based systems in solving real world engineering Problems.
<b>PSO3</b>	<b>Successful Career:</b> Ability to update knowledge continuously in the tools like Rational Rose, MATLAB, Argo UML, R Language and technologies like Storage, Computing, Communication to meet the industry requirements in creating innovative career paths for immediate employment and for higher studies.

### COURSE ARTICULATION MATRIX:

Course Outcomes	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
<b>CO1</b>	3	2	3	1	3	-	-	-	-	-	-	-	2	1	2
<b>CO2</b>	3	3	2	1	2	-	-	-	-	-	-	-	1	2	2
<b>CO3</b>	3	2	2	1	2	-	-	-	-	-	-	-	1	2	2
<b>CO4</b>	3	2	3	1	3	-	-	-	-	-	-	-	2	1	2
<b>CO5</b>	3	3	2	1	2	-	-	-	-	-	-	-	1	2	2
<b>1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)</b>															

### 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.



### LIST OF EXPERIMENTS

S.NO	EXPERIMENT NO.	NAME OF THE EXPERIMENT	PAGE NO.
1	1	Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.	10
2	2	Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP	18
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.	24
4	4	Implement Dijkstra's algorithm to compute the shortest path through a network	27
5	5	Take an example subnet of hosts and obtain a broadcast tree for the subnet.	32
6	6	Implement distance vector routing algorithm for obtaining routing tables at each node.	39
7	7	Implement data encryption and data decryption	44
8	8	Write a program for congestion control using Leaky bucket algorithm.	46
9	9	Write a program for frame sorting technique used in buffers.	49

### EXPERIMENT NO: 1. (a)

**Implement the data link layer framing methods such as character, character-stuffing and 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 of 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.

### PROGRAM ALGORITHM:

Begin

Step 1: Read frame length n

Step 2: Repeat step (3 to 4) until  $i < n$ : Read values into the input frame (0's and 1's) i.e.

Step 3: initialize  $i = 0$ ;

Step 4: read  $a[i]$  and increment  $i$

Step 5: Initialize  $i = 0, j = 0, \text{count} = 0$

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  $\text{count} < 5$ )

Step 10: Initialize  $k = i + 1$ ;

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

Step 12: Increment  $\text{count}$  ;

Step 13: if  $\text{count} = 5$  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

### PROGRAM 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]);
}

```

### PROGRAM 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

```

**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 loses 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.

**PROGRAM 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: endif

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: endif

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: endif

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

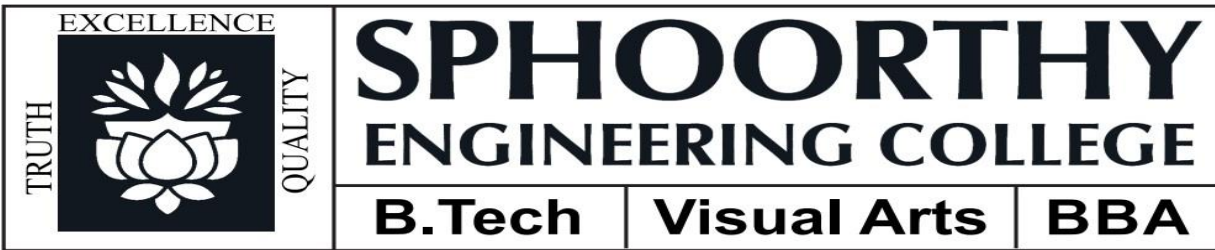
Step 24: print frame after stuffing

Step 25: print b

End

### **PROGRAM 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");
```



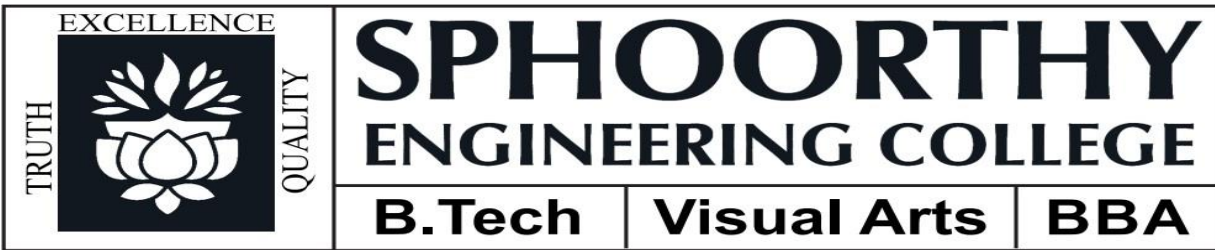
```
scanf("%d",&pos);
if(pos>n)
{
printf("invalid position, Enter again :");
scanf("%d",&pos);}
printf("Enter the character\n");
ch=getche();
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
if(i==pos-1)
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]=ch;
b[j+4]='d';
b[j+5]='l';
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);
}

```





**PROGRAM OUTPUT:**

Enter string

MLRITM

Enter position

2

Enter the character

frame after stuffing:

dlestxMdldleLRITMdleetx

-----

(program exited with code: 0)

Press return to continue

## EXPERIMENT NO: 2.

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

**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  $2^{-n}$ . This scheme sometimes known as Cyclic Redundancy Code.

### PROGRAM 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  $i = \text{frl}$

Step 14: Repeat step(15to16) until  $i < \text{tlen}$

Step 15:  $\text{dupfr}[i] = 0$

Step 16: increment  $i$

Step 17: call the function  $\text{remainder}(\text{dupfr})$

Step 18: initialize  $i = 0$

Step 19: repeat step(20 to 21) until  $j < \text{genl}$

Step 20:  $\text{recfr}[i] = \text{rem}[j]$

Step 21: increment  $i$  and  $j$

Step 22: call the function  $\text{remainder}(\text{dupfr})$

Step 23: initialize  $\text{flag} = 0$  and  $i = 0$

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

Step 25: if  $\text{rem}[i] \neq 0$  then

Step 26: increment  $\text{flag}$

Step 27: end if

Step 28: increment  $i$

Step 29: if  $\text{flag} = 0$  then

Step 25: print frame received correctly

Step 25: else

Step 25: print the received frame is wrong

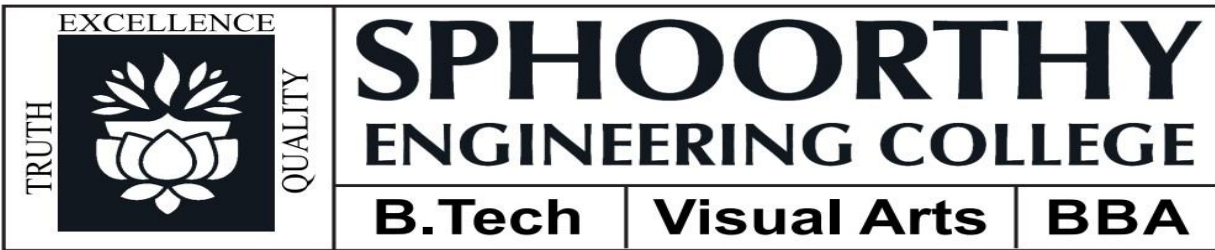
End

Function:  $\text{Remainder}(\text{int fr}[])$

Begin

Step 1: Declare  $k, k1, i, j$  as integer

Step 2: initialize  $k = 0$ ;



Step 3: repeat step(4 to 14) until  $k < \text{frl}$

Step 4: if  $((\text{fr}[k] == 1))$  then

Step 5:  $k1 = k$

Step 6: initialize  $i=0, j=k$

Step 7: repeat step(8 to 9) until  $i < \text{genl}$

Step 8:  $\text{rem}[i] = \text{fr}[j]$  exponential  $\text{gen}[i]$

Step 9: increment  $i$  and  $j$

Step 10: initialize  $I = 0$

Step 11: repeat step(12 to 13) until  $I < \text{genl}$

Step 12:  $\text{fr}[k1] = \text{rem}[i]$

Step 13: increment  $k1$  and  $i$

Step 14: end if

End

### **PROGRAM CODE: // PROGRAM FOR CYCLIC REDUNDENCY CHECK**

```
#include<stdio.h>

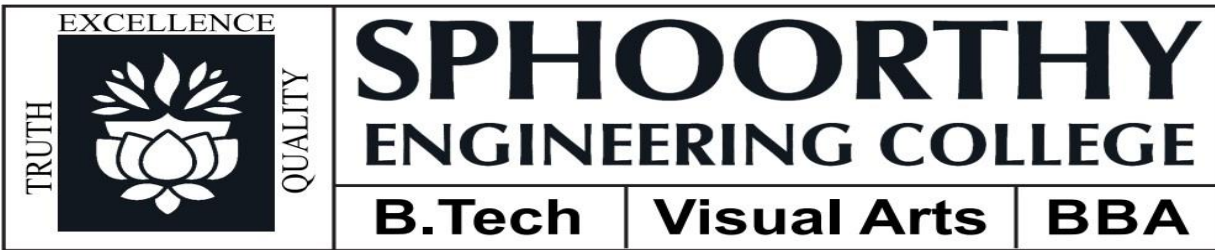
int gen[4],genl,frl,rem[4];

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++;
}

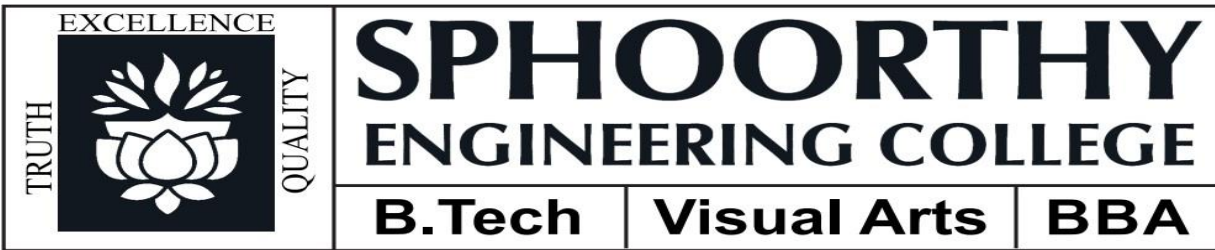
```



```

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++;

```



}  
}  
}  
}

### PROGRAM OUTPUT:

Enter frame: MLRITM

Enter generator: frame received correctly

-----

(program exited with code: 24)

Press return to continue.

### EXPERIMENT NO: 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:** 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.

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

**SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### THEORY:

Sliding window is a technique for controlling transmitted data frames between two network computers where reliable and sequential delivery of data frames is required, such as when using the Data Link Layer (OSI model) or Transmission Control Protocol (TCP). In the sliding window technique, each data frame (for most data link layers) and byte (in TCP) include a unique consecutive sequence number, which is used by the receiving computer to place data in the correct order. Data corruption occurred in transit. The objective of the sliding window technique is to use the sequence numbers to avoid duplicated at a and to request missing data. Sliding window is also known as windowing.

### Go-Back-N ARQ:

Stop and wait ARQ mechanism does not utilize the resources at their best. When the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N ARQ method, both sender and receiver maintain a window. The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame's sequence number. When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of



frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

## SOURCE CODE:

```
#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 CONSOLE:

```

C:\Users\SPHN\Documents\hello.exe
Enter window size: 3
Enter number of frames to transmit: 5
Enter 5 frames: 89 56 45 2 3
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
89 56 45
Acknowledgement of above frames sent is received by sender
2 3
Acknowledgement of above frames sent is received by sender
-----
Process exited after 17.66 seconds with return value 0
Press any key to continue . . .

```

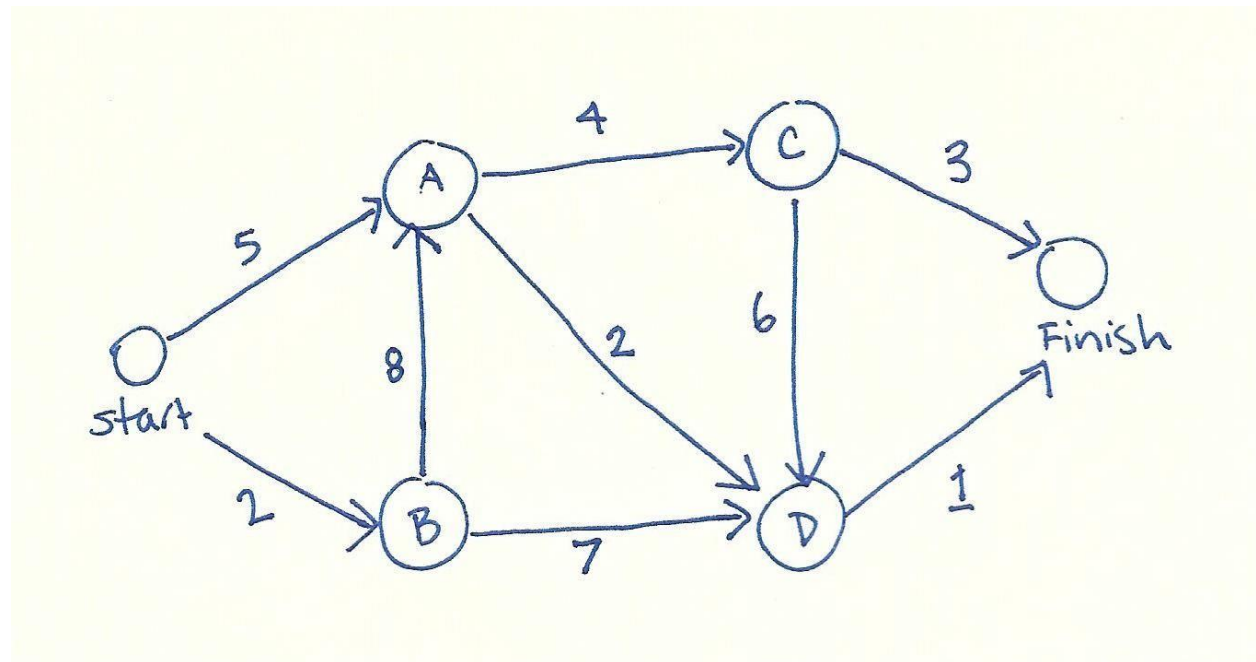
#### EXPERIMENT NO: 4

**Implement Dijkstra's algorithm to compute the shortest path through a network**

**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.



#### PROGRAM 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 ( $i \leq 5$ )

Step 7:  $j=1$

Step 8: repeat step (9 to 10) until ( $j \leq 5$ )

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 ( $i \leq p$ )

Step 17:  $j=1$

Step 18: repeat step(19 to 20) until ( $i \leq 5$ )

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( $i \leq p$ )

Step 24:  $t[i]=0$

Step 25:  $stp=st$

Step 26:  $j=1$

Step 27: repeat step(26 to 34) until( $j \leq 5$ )

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

### PROGRAM CODE: 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;
}
}

```

### PROGRAM 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: 5

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

**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: write "\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: initialize min = edge[i][j]

step31: initialize u = i

step32: initialize v = j

step33: calling function  $p = \text{find}(u)$ ;

step34: calling function  $q = \text{find}(v)$ ;

step35: if  $P \neq q$  repeat steps(35-39)

step36: initialize  $t[i][0] = u$

step37: initialize  $t[i][1] = v$

step38: initialize  $\text{mincost} = \text{mincost} + \text{edge}[u][v]$

step39: call function  $\text{union}(p, q)$

step40: else repeat steps(40-42)

step41: Initialize  $t[i][0] = -1$ ;

step42: Initialize  $t[i][1] = -1$ ;

step43: initialize  $\text{min} = 99$ ;

step44; write "Minimum cost is %d\n Minimum spanning tree is",  $\text{mincost}$

step45: Initialize  $i = 0$

step46: repeat until  $i < n$  step47: increment  $i$  step48: if  $t[i][0] \neq -1 \ \&\& \ t[i][1] \neq -1$  repeat step(48-50)

step49: write "%c %c %d",  $65 + t[i][0]$ ,  $65 + t[i][1]$ ,  $\text{edge}[t[i][0]][t[i][1]]$

step50: write "\n" step51: end step52: called function  $\text{union}(\text{int } l, \text{int } m)$  repeat step(51-52)

step53: initialize  $\text{parent}[l] = m$

step54: called function  $\text{find}(\text{int } l)$  repeat step(53-56)

step55: if  $\text{parent}[l] > 0$

step56: initialize  $l = \text{parent}$

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++)
```

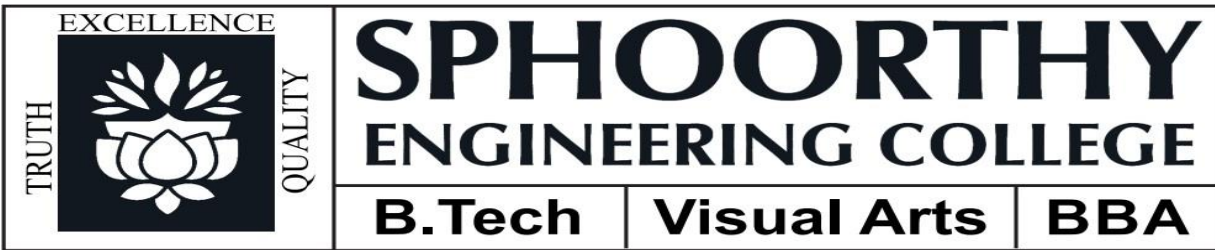
```
scanf("%d",&edge[i][j]);
```

```
}
```

```

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];
union(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 I,int m)
```

```
{
```

```
parent[]=m;
```

```
}
```

```
find(int I)
```

```
{
```

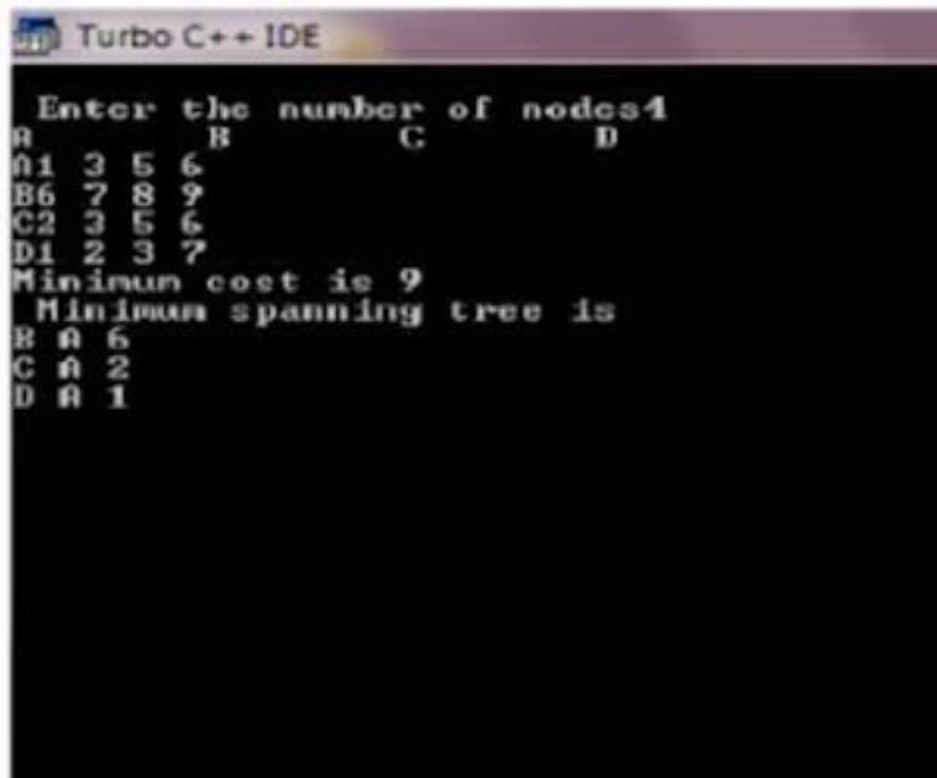
```
if(parent[I]>0)
```

```
I=parent[I];
```

```
return I;
```

```
}
```

OUTPUT:



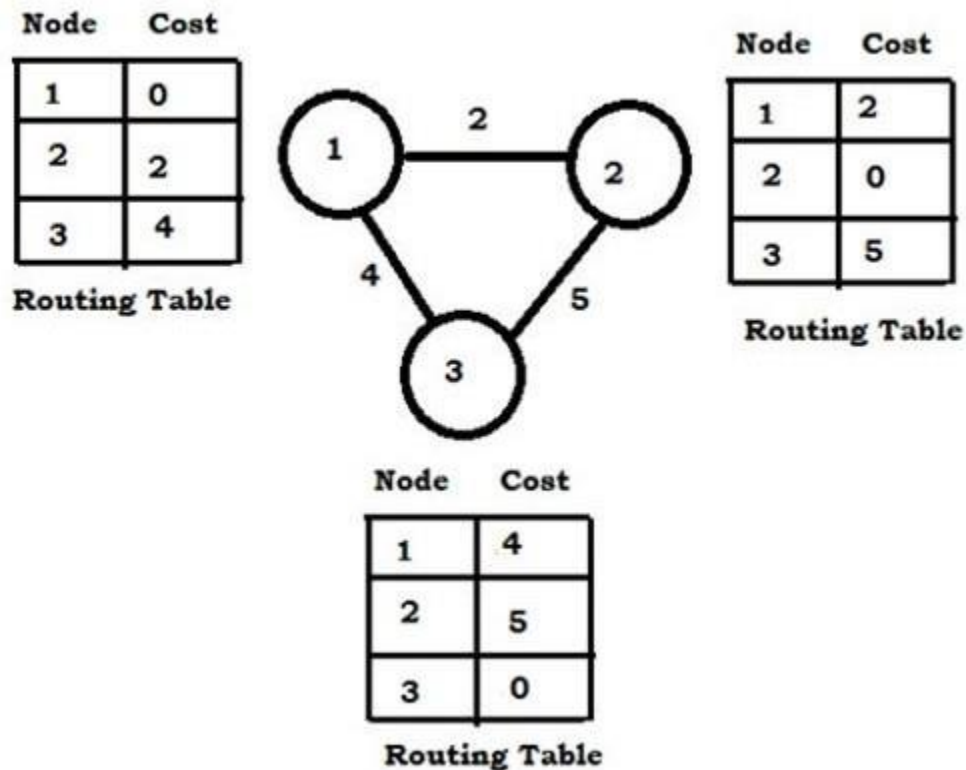
```

Turbo C++ IDE
Enter the number of nodes: 4
A B C D
A 1 3 5 6
B 6 7 8 9
C 2 3 5 6
D 1 2 3 7
Minimum cost is 9
Minimum spanning tree is
A 6
C 2
D 1
  
```

### EXPERIMENT NO: 6

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

**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:** initialize 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:**initialize dmat[i][j]=0

**Step14:**initialize rt[i].dist[j]=dmat[i][j]

**Step15:**initialize 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

### 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

## EXPERIMENT NO: 7

### Implement data encryption and data decryption

**AIM:** Implement data encryption and data decryption

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

**SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### SOURCE CODE:

```
//Simple C program to encrypt and decrypt a string

#include <stdio.h>
int main()
{
    int i, x;
    char str[100];

    printf("\nPlease enter a string:\t");
    gets(str);

    printf("\nPlease choose following options:\n");
    printf("1 = Encrypt the string.\n");
    printf("2 = Decrypt the string.\n");
    scanf("%d", &x);

    //using switch case statements
    switch(x)
    {
    case 1:
        for(i = 0; (i < 100 && str[i] != '\0'); i++)
            str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value

        printf("\nEncrypted string: %s\n", str);
        break;

    case 2:
        for(i = 0; (i < 100 && str[i] != '\0'); i++)
            str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value

        printf("\nDecrypted string: %s\n", str);
```

```

break;

default:
    printf("\nError\n");
}
return 0;
}

```

## OUTPUT:

### #Encryption

```

"C:\Users\ICT\Google Drive\c project\code\encrypt-decrypt.exe"

Please enter a string:  hello

Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
1

Encrypted string: khood

Process returned 0 (0x0)   execution time : 8.564 s
Press any key to continue.

```

### #Decryption

```

"C:\Users\ICT\Google Drive\c project\code\encrypt-decrypt.exe"

Please enter a string:  khood

Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
2

Decrypted string: hello

Process returned 0 (0x0)   execution time : 4.288 s
Press any key to continue.

```

## EXPERIMENT NO: 8

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

**AIM:** Implement a program for congestion control using Leaky bucket algorithm.

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

**SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### SOURCE CODE:

```
#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)-\n\nPACKET 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:

```
fsmk@fsmk-ThinkCentre-M71e: ~/network_done/leaky_bucket
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!!

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
```



## EXPERIMENT NO: 9

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

**AIM:** Implement a program for frame sorting technique used in buffers.

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

**SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### THEORY:

A frame is a digital data transmission unit in computer networking and telecommunication. A frame typically includes frame synchronization features consisting of a sequence of bits or symbols that indicate to the receiver the beginning and end of the payload data within the stream of symbols or bits it receives. If a receiver is connected to the system in the middle of a frame transmission, it ignores the data until it detects a new frame synchronization sequence.

### SOURCE CODE:

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct frame{
int fslno;
char finfo[20];
};

struct frame arr[10];

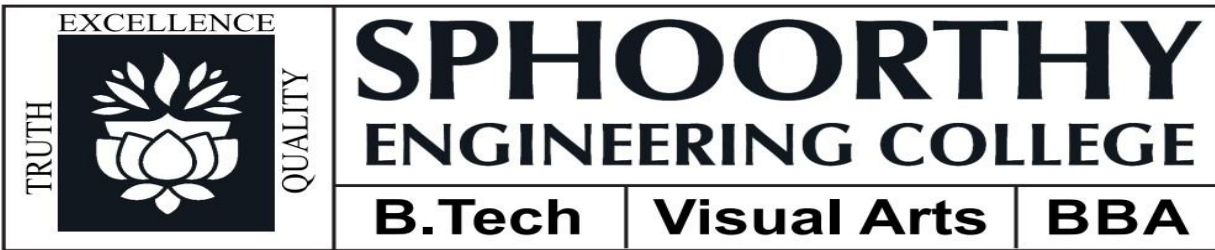
int n;

void sort()
{
int i,j,ex;
struct frame temp;
for(i=0;i<n;i++)
{
```

```

ex=0;
for(j=0;j<n-i-1;j++)
if(arr[j].fslno>arr[j+1].fslno)
{
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
ex++;
}
if(ex==0) break;
}
}
void main()
{
int i;
clrscr();
printf("\n Enter the number of frames \n");
scanf("%d",&n);
for(i=0;i<n;i++)
{ arr[i].fslno=random(50);
printf("\n Enter the frame contents for sequence number
%d\n",arr[i].fslno);
scanf("%s",arr[i].finfo);
}
sort();

```



```
printf("\n The frames in sequence \n");
for(i=0;i<n;i++)
printf("\n %d\t%s \n",arr[i].fslno,arr[i].finfo);
getch();
}
```

### OUTPUT:

Enter the number of frames:3

Enter the frame contents for sequence number 23

Wrote

Enter the frame contents for sequence number 45

Program

Enter the frame contents for sequence number 9

Vikas

The frames in sequence

9 Vikas

23 Wrote

45 Program.