

Guru Nanak Institutions Technical Campus(Autonomous)

School of Engineering & Technology
Department of Computer Science & Engineering(AIML&IOT)

Computer Networks Lab
Lab Manual [Subject Code: 18PC0AM05]
For the Academic year 2022-2023

III B Tech Semester I– CSE(AIML&IOT)



Guru Nanak Institutions Technical Campus (Autonomous)
Ibrahimpattanam, R R District – 501 506 (TS)

Department of Computer Science & Engineering

LAB MANUAL FOR THE ACADEMIC YEAR 2022-2023

SUB : Computer Networks Lab
SUB CODE : 18PC0AM05
SEMESTER : I
STREAM : CSE(AIML & IOT)
Document No : GNITC/CSE(AIML & IOT)/ CN/R-18
Date of Issue : August 24th 2022
Prepared : M.V. Anjana Devi/P.V.Pratima
PROGRAMMER : Laxman Kumar
VENUE : LAB-RD018
BLOCK : R&D

Authorized by

HOD-CSE(AIML & IOT)

INDEX:

S.No	Contents	Page.no
1	Lab Objective	4
2	Lab outcomes	4
3	Introduction About Lab	4-5
4	Guidelines to students a). Standard Operating Procedure (SOP) b). General guidelines	5-7
5	List of experiments as per the university curriculum	8
6	List of Additional Experiments	8
7	Text Books / Reference Books	9
8	Content of the experiment. Objective of the Experiment Hardware & Software Requirements Pre- requisite Flow chart/ algorithm with inputs and outputs Program Output Conclusion	10-28

COMPUTER NETWORKS LAB:

1.LAB OBJECTIVES

Upon successful completion of this Lab the student will be able to:

1. Intended to provide practical exposure of the concepts in computer networks.
2. Provide hands on experience of designing, modeling, and evaluation of computer networks

2.LAB OUTCOMES

Upon successful completion of this Lab the student will be able to:

1. Implement data link layer framing methods.
2. Implement error correction and detection techniques.
3. Implement data link layer protocols
4. Implement routing and congestion algorithms
5. Implement encryption algorithms
6. Able to create a scenario and study the performance of computer networks and protocols

3. INTRODUCTION ABOUT LAB

There are 60 systems (Compaq Presario) installed in this Lab. Their configurations are as follows:

Processor	:	Intel(R) Pentium(R) Dual CPU 2.0GHz
RAM	:	2 GB
Hard Disk	:	160 GB
Mouse	:	Optical Mouse

Software

- All systems are configured in DUAL BOOT mode i.e., Students can boot from Windows XP or Linux as per their lab requirement. This is very useful for students because they are familiar with different Operating Systems so that they can execute their programs in different programming environments.
- Each student has a separate login for database access Oracle 9i client version is installed in all systems. On the server, account for each student has been created. This is very useful because students can save their work (scenarios', PL / SQL programs, data related projects, etc) in their own accounts. Each student work is safe and secure from other students.
- ETL TOOL: Informatica 7.1.1 is installed in all the systems.
Students can execute ETL process using this tool. They can import the Meta data from different source definition into the data warehouse by applying the business logics.
- Reporting TOOL: COGONS is installed in all the systems.
Students can generate reports using this tool from different Business Objects in different formats.
- MASM (Macro Assembler) is installed in all the systems

- Students can execute their assembly language programs using MASM. MASM is very useful students because when they execute their programs they can see contents of Processor Registers and how each instruction is being executed in the CPU.
- Rational Rose Software is installed in some systems
- Using this software, students can depict UML diagrams of their projects.
- Software's installed: C, C++, JDK1.5, MASM, OFFICE-XP, J2EE and DOT NET, Rational Rose.
- Systems are provided for students in the 1:1 ratio.
- Systems are assigned numbers and same system is allotted for students when they do the lab.

4. GUIDELINES TO STUDENTS

4.A) STANDARD OPERATING PROCEDURE – SOP

- a) Explanation on today's experiment by the concerned faculty using OHP/PPT covering the following aspects: 25 min.
 - 1) Name of the experiment/Aim
 - 2) Software/Hardware required
 - 3) Commands with suitable Options
 - 4) Test Data
 - 5) Valid data sets
 - 6) Limiting value sets
 - 7) Invalid data sets
- b) Writing of c programs by the students 25 min.
- c) Compiling and execution of the program

100 mins.

Writing of the experiment in the Observation Book:

The students will write the today's experiment in the Observation book as per the following format:

- a) Name of the experiment/Aim
- b) Software/Hardware required
- c) Commands with suitable Options
- d) System call using C-Programs
- e) Test Data
 - a. Valid data sets
 - b. Limiting value sets
 - c. Invalid data sets
- f) _____ Results for different data sets
- g) _____ Viva-Voce Questions and Answers

- h) _____ Errors observed (if any) during compilation/execution
i) _____ Signature of the Faculty

4.B) GUIDE LINES TO STUDENTS IN LAB

Students are advised to maintain discipline and follow the guidelines given below:

- Keep all your bags in the racks and carry the observation book and record book.
- Mobile phones/pen drives/ CDs are not allowed in the labs.
- Maintain proper dress code along with ID Card
- Occupy the computers allotted to you and maintain the discipline.
- Student must submit the record with the last week experiment details and observation book with the brief of the present experiment.
- Read the write up of the experiment given in the manual.
- Students must use the equipment with care. Any damage is caused student is punishable
- After completion of every experiment, the observation notes to be shown to the lab in - charge and after correction the record must be updated and submit to the lab in charge for correction.
- Lab marks are given on Continuous Evaluation Basis as per JNTU guidelines
- If any student is absent for any lab, they need to be complete the same experiment in the free time before attending next lab session.

Steps to perform experiments in the lab by the student

Step1: Students have to write the Date, aim, Software and Hardware requirements for the scheduled experiment in the observation book.

Step2: Students have to listen and understand the experiment explained by the faculty and note down the important points in the observation book.

Step3: Students need to write procedure/algorithm in the observation book.

Step4: Analyze and Develop/implement the logic of the program by the student in respective platform

Step5: After approval of logic of the experiment by the faculty then the experiment has to be executed on the system.

Step6: After successful execution, the results have to be recorded in the observation book and shown to the lab in charge faculty..

Step7: Students need to attend the Viva-Voce on that experiment and write the same in the observation book.

Step8: Update the completed experiment in the record and submit to the concerned faculty in-charge.

Instructions to maintain the record

- Before starting of the first lab session students must buy the record book and bring the same to the lab.

- Regularly (Weekly) update the record after completion of the experiment and get it corrected with concerned lab in-charge for continuous evaluation.
- In case the record is lost, inform on the same day to the faculty in charge and submit the new record within 2 days for correction.
- If record is not submitted in time or record is not written properly, the record evaluation marks (5M) will be reduced accordingly.

Awarding the marks for day to day evaluation:

Total marks for day to day evaluation: 10 Marks (as per JNTUH).

Breakup for 10 Marks:

Record	5 Marks
Exp setup/program written and execution	3 Marks
Result and Viva-Voce	2 Marks

Allocation of Marks for Lab Internal Examinations:

Total marks for lab internal Examination : 30 Marks

Break up for 30 Marks :

Average of day to day evaluation marks: 10 Marks

Lab Internal Mid examination: 10 Marks

Viva Voice & Attendance(%) : 10 Marks

Allocation of Marks for Lab External Examintions:

Total marks for External lab Examinations: 70 Marks.

5. LIST OF LAB EXERCISES

S. No	Name of the experiment	Page No
1	Implement the data link layer framing method “character-stuffing”	10-12
2	Implement the data link layer framing method “bit stuffing”	12-14
3	Write a program to compute CRC code for the polynomial “CRC-12”	14-15
4	Write a program to compute CRC code for the polynomial “CRC-16”	16-20
5	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	20-21
6	Implement Dijkstra’s algorithm to compute the shortest path through a network	21-23
7	Write a program to implement congestion control in network layer using leaky bucket algorithm	23-25
8	Take an example subnet of hosts and obtain a broadcast tree for the subnet	25-27
9	Implement distance vector routing algorithm for obtaining routing tables at each node	27-29

6. LIST OF ADDITIONAL EXPERIMENTS FOR THE SEMESTER

S. No	Name of the experiment	Page No
1	To write a C program to develop a DNS client server to resolve the given hostname	29-30
2	Write a program for Hamming Code generation for error detection and correction	30-31
3	Implement the above program using as message queues or FIFO as IPC channels	31-33

7.TEXTBOOKS

1. Computer Networks -- Andrew S Tanenbaum, David. j. Wetherall, 5th Edition.
Pearson Education/PHI

REFERENCE BOOKS:

1. An Engineering Approach to Computer Networks-S.Keshav,2nd
Edition,Pearson Education
2. Data Communications and Networking – Behrouz A. Forouzan. Third Edition TMH.

Experiment-1

1. Implement the data link layer framing method “character-stuffing”

Theory:

Coming to the Character Stuffing, DLESTX and DLEETX are used to denote start and end of character data with some constraints imposed on repetition of characters as shown in the program below clearly.

SOURCE CODE: // CHARACTER STUFFING

PROGRAM:

```
#include<stdio.h>
#include<string.h>
void charc(void);
void main()
{
    int choice;
    while(1)
    {
        printf("\n\n1.character stuffing");
        printf("\n\n2.exit");
        printf("\n\nenter choice");
        scanf("%d",&choice);
        printf("%d",choice);
        if(choice>2)
            printf("\n\n invalid option....please reenter");
        switch(choice)
        {
            case 1:
                charc();
                break;
            case 2:
                exit(0);
        }
    }
}

void charc(void)
{
    char c[50],d[50],t[50];
    int i,m,j;
    clrscr();
    printf("enter the number of characters\n");
    scanf("%d",&m);
    printf("\n enter the characters\n");
    for(i=0;i<m+1;i++)
    {
        scanf("%c",&c[i]);
    }
}
```

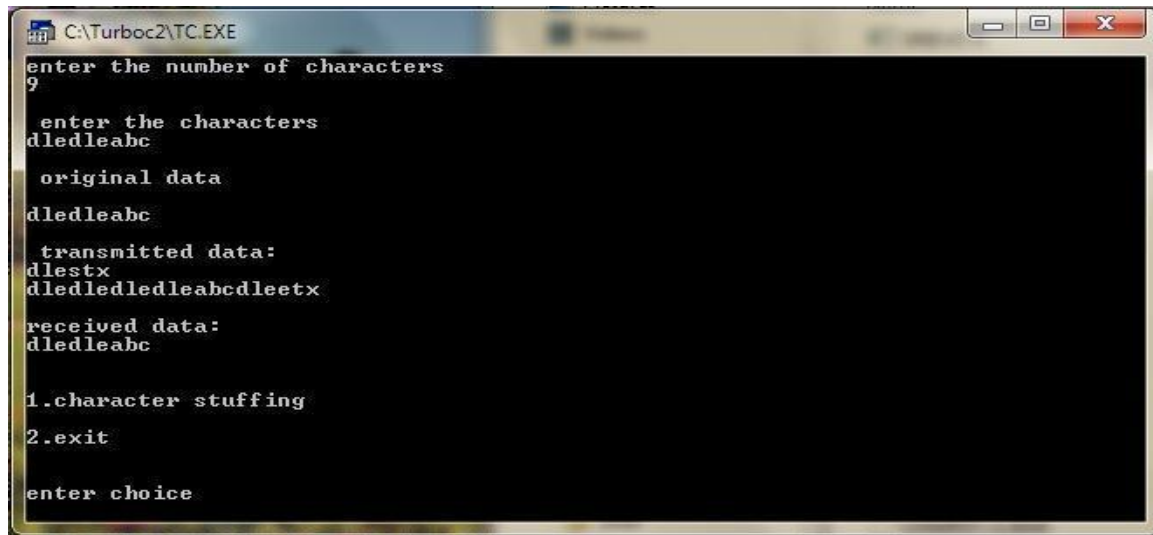
```

printf("\n original data\n");
for(i=0;i<m+1;i++)
    printf("%c",c[i]);
d[0]='d';
d[1]='l';
d[2]='e';
d[3]='s';
d[4]='t';
d[5]='x';
for(i=0,j=6;i<m+1;i++,j++)
{
    if((c[i]=='d'&& c[i+1]=='l'&& c[i+2]=='e'))
    {
        d[j]='d';
        j++;
        d[j]='l';
        j++;
        d[j]='e';
        j++;
        m=m+3;
    }
    d[j]=c[i];
}
m=m+6;
m++;
d[m]='d';
m++;
d[m]='l';
m++;
d[m]='e';
m++;
d[m]='e';
m++;
d[m]='t';
m++;
d[m]='x';
m++;
printf("\n\n transmitted data: \n");
for(i=0;i<m;i++)
{
    printf("%c",d[i]);
}
for(i=6,j=0;i<m-6;i++,j++)
{
    if(d[i]=='d'&& d[i+1]=='l'&& d[i+2]=='e'&& d[i+3]=='d'&& d[i+4]=='l'&& d[i+5]=='e')
        i=i+3;
    t[j]=d[i];
}
printf("\n\n received data:");
for(i=0;i<j;i++)
    {printf("%c",t[i]);

```

```
}  
}
```

OUTPUT:



A screenshot of a TurboC++ console window titled "C:\Turboc2\TC.EXE". The window has a black background with white text. The output of the program is as follows:

```
enter the number of characters  
9  
  
enter the characters  
dledleabc  
  
original data  
dledleabc  
  
transmitted data:  
dlestx  
dledledleabcdleetx  
  
received data:  
dledleabc  
  
1.character stuffing  
2.exit  
  
enter choice
```

Exprimment: 2

2. Implement the data link layer framing method "bit stuffing"

Theory:

Security and Error detection are the most prominent features that are to be provided by any application which transfers data from one end to the other end. One of such a mechanism in tracking errors which may add up to the original data during transfer is known as Stuffing. It is of two types namely Bit Stuffing and the other Character Stuffing. Coming to the Bit Stuffing, 01111110 is appended within the original data while transfer of it. The following program describes how it is stuffed at the sender end and de-stuffed at the reciever end.

SOURCE CODE: // BIT STUFFING

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
    int a[15];
    int i,j,k,n,c=0,pos=0;
    clrscr();
    printf("\n Enter the number of bits");
    scanf("%d",&n);
    printf("\n Enter the bits");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=0;i<n;i++)
    {
        if(a[i]==1)
        {
            c++;
            if(c==5)
            {
                pos=i+1;
                c=0;
                for(j=n;j>=pos;j--)
                {
                    k=j+1;
                    a[k]=a[j];
                }
                a[pos]=0;
                n=n+1;
            }
        }
        else
            c=0;
    }
}
```

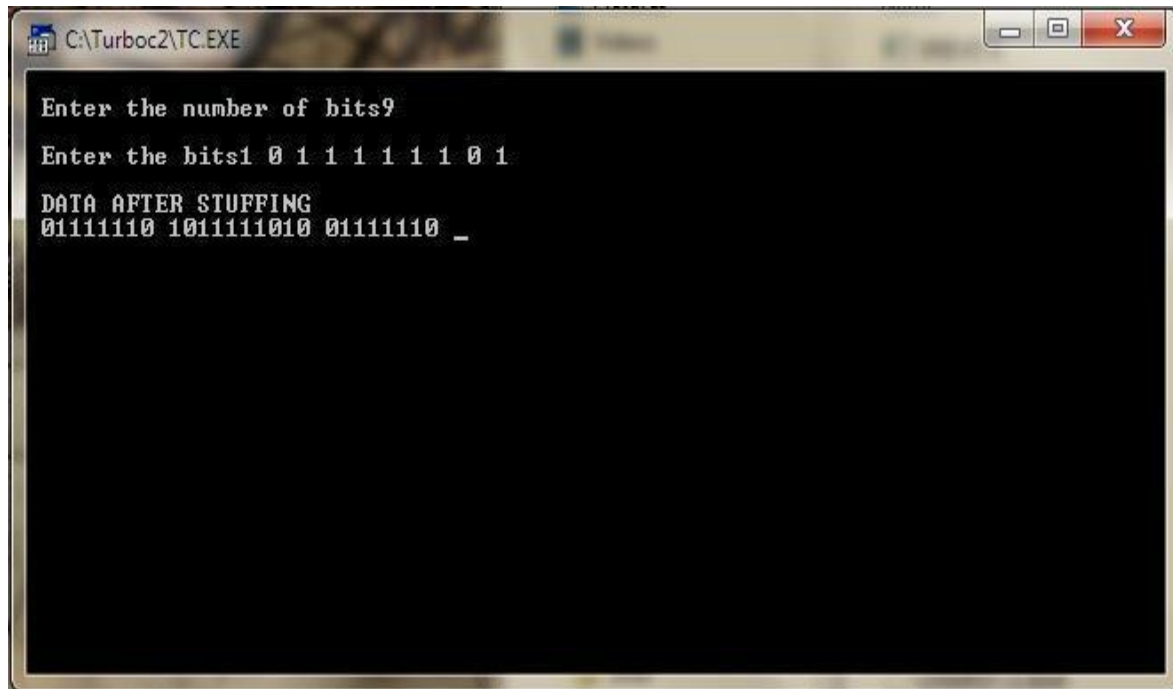
```

printf("\n DATA AFTER STUFFING \n");
printf(" 01111110 ");

for(i=0;i<n;i++)
{
    printf("%d",a[i]);
}
printf(" 01111110 ");
getch();
}

```

OUTPUT:



Experiment: 3

3. Write a program to compute CRC code for the polynomial “CRC-12”

Theory:

CRC means Cyclic Redundancy Check. It is the most famous and traditionally successful mechanism used in error detection through the parity bits installed within the data and obtaining checksum which acts as the verifier to check whether the data retrieved at the receiver end is genuine or not. Various operations are involved in implementing CRC on a data set through CRC generating polynomials.

PROGRAM:

```

#include<stdio.h>
#include<conio.h>

```

```

#include<string.h>
#define N strlen(g)
    char t[28],cs[28],g[28];
    int a,e,c,b;
void xor()
{
    for(c=1;c<N;c++)
        cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc()
{
    for(e=0;e<N;e++)
        cs[e]=t[e];
    do
    {
        if(cs[0]=='1')
            xor();
        for(c=0;c<N-1;c++)
            cs[c]=cs[c+1];
        cs[c]=t[e++];
    }while(e<=a+N-1);
}
void main()
{
    int flag=0;
    clrscr();
    do
    {
        printf("\n1.crc12\n2.exit\n\nEnter your option.");
        scanf("%d",&b);
        switch(b)
        {
            case 1:
                strcpy(g,"10011");
                break;
            case 2:
                exit(0);
        }
        printf("\n enter data:");
        scanf("%s",t);
        printf("\n-----\n");
        printf("\n generating polynomial:%s",g);
        a=strlen(t);
        for(e=a;e<a+N-1;e++)
            t[e]='0';
        printf("\n-----\n");
        printf("mod-ified data is:%s",t);
        printf("\n-----\n");
        crc();
        printf("checksum is:%s",cs);
        for(e=a;e<a+N-1;e++)

```

```

t[e]=cs[e-a];
printf("\n-----\n");
printf("\n final codeword is : %s",t);
printf("\n-----\n");
printf("\ntest error detection 0(yes) 1(no)?:"");
scanf("%d",&e);
if(e==0)
{
    do
    {
        printf("\ntenter the position where error is to be inserted:");
        scanf("%d",&e);
    } while(e==0||e>a+N-1);
    t[e-1]=(t[e-1]=='0')?'1':'0';
    printf("\n-----\n");
    printf("\n\terroneous data:%s\n",t);
}
crc();
for(e=0;(e<N-1)&&(cs[e]!='1');e++);
if(e<N-1)
    printf("error detected\n\n");
else
    printf("\n no error detected \n\n");
printf("\n-----");
} while(flag!=1);
}

```

OUTPUT 1:


```

1.crc12
2.exit
Enter your option.1

enter data:1110001101

-----

generating polynomial:10011
-----
mod-ified data is:11100011010000
-----
checksum is:1000
-----

final codeword is : 11100011011000
-----

test error detection 0(yes) 1(no)?:1

no error detected
-----
1.crc12
2.exit
Enter your option._

```

Activate Windows
Go to Settings to
activate Windows.

OUTPUT 2:

```

1.crc12
2.exit
Enter your option.1

enter data:110001101

-----

generating polynomial:10011
-----
mod-ified data is:1100011010000
-----
checksum is:0101
-----

final codeword is : 1100011010101
-----

test error detection 0(yes) 1(no)?:0

enter the position where error is to be inserted:3_

```

Activate Windows
Go to Settings to
activate Windows.

```

erroneous data:1110011010101
error detected
-----
1.crc12
2.exit
Enter your option._

```

Experiment: 4

4. Write a program to compute CRC code for the polynomial “CRC-16”

Program for CRC-16:-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<dos.h>
#include<stdlib.h>
int copy();
int check();
int i, j, k, t, count = 0, num = 0;
int gen[10], frame[30], rem[30], temp[30];
void main()
{
    char c, plym[50];
    char ch[]={'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
    clrscr();
    printf("\t\t\t***CYCLIC REDUNDANCY CHECK-16***\n\n\n");
    for(i = 0; i < 50; i++)
        plym[i] = '\0';
    for(i = 0; i < 30; i++)
        temp[i] = rem[i] = frame[i] = '\0';
    for(i = 0; i < 10; i++)
        gen[i] = '\0';
    printf("enter the polynomial: ");
    gets(plym);
    plym[strlen(plym)] = ' ';
    for(i = 0; i < strlen(plym); i++)
    {
        if(plym[i] == 'x')
        {
            i++;
            for(;plym[i] != ' '; count, i++)
            {
            }
            if(count == 3)
            {

```

```

        for(i = i - 1, j = 3; j <= 9; j++)
        if(plym[i] == ch[j])
        {
            printf("\nEnter the polynomial's");
            printf("degree is high");
            getch(); exit(0);
        }
        for(j = 0, num = 10; j <= 2; j++)
        if(plym[i] == ch[j])
        {
            num=num + j;
            frame[num] = 1;
        }
    }
    if(count == 2)
    {
        for(i = i - 1, j = 1, num = 0; j <= 9; j++)
        if(plym[i] == ch[j])
        num = j;
        frame[num] = 1;
    }
    if(count == 0)
    frame[1] = 1;
    count = 0;
}
else if(plym[i] == '1')
frame[0] = 1;
}
printf("FRAME is: ");
for(i = 12, j = 0; i >= 0; i--, j++)
{
    temp[j] = frame[i];
    printf("%d", frame[i]);
}
printf("\n\n\n>>>>both high & low orders");
printf("bits of GENERATOR must be 1<<<<");
printf("\n enter the generator: ");
for(num = i = 0; (c = getchar()) != '\n'; i++, num++)
{
    if(c == '1')

```

```

    gen[i] = 1;
    else if(c == '0')
    gen[i] = 0;
    else
    {
        printf("\n Enter the GENERATOR");
        printf("is other then 0 or 1");
        getch(); exit(0);
    }
}
for(j = 13; i = i - 1; i > 0; i--, j++)
temp[j] = 0;
printf("\n\n FRAME after appending 0's: ");
copy(); check();
printf("\n The REMAINDER is: ");
for(i = 13; i < j; i++)
{
    temp[i] = rem[i];
    printf("%d", rem[i]);
}
printf("\n\n\n Transmitting FRAME.....");
delay(10000);
printf("\n\n\n Transmitted FRAME is: ");
copy(); check();
printf("\n frame recieved");
printf("\n\n\n checking for errors.....");
delay(10000);
printf("\n\n\n recieved frame is: ");
copy(); check();
printf("\n the remainder is: ");
for(i = 13; i < j; i++)
printf("%d", rem[i]);
printf("\n DATA SENT SUCCESSFULLY");
getch();
}
check()
{
    for(i = 0; i <= 12; i++)
    {
        if(rem[i] == 0)

```

```

        continue;
    else
    {
        for(k = 0, t = i; k < num; k++, t++)
        {
            if(rem[t] == 1 && gen[k] == 1)
                rem[t] = 0;
            else if(rem[t] == 0 && gen[k] == 0)
                rem[t] = 0;
            else if(rem[t] == 1 && gen[k] == 0)
                rem[t] = 1;
            else if(rem[t] == 0 && gen[k] == 1)
                rem[t] = 1;
        }
    }
}
return 0;
}
copy()
{
    for(i = 0; i < j; i++)
    {
        printf("%d", temp[i]);
        rem[i] = temp[i];
    }
    return 0;
}

```

Output:-

CYCLIC REDUNDANCY CHECK-16

Enter the polynomial: $x^{15} x^8 x^7 x^5 x^3 x^2 1$

FRAME is: 01000000110101101

>>>>both high & low ordersbits of GENERATOR must be 1<<<<

Enter the generator: 10011

FRAME after appending 0's: 010000001101011010000

The REMAINDER is: 1100

Transmitting frame.....

TRANSMITTED FRAME is: 010000001101011011100

frame RECEIVED

Checking for ERRORS.....

Recieved Frame is: 010000001101011011100

The REMAINDER is: 0000

DATA SENT SUCCESSFULLY

5. 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

Theory:

Go-Back-N ARQ is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver.

PROGRAM:

```
#include<stdio.h>
int main()
{
    int window size,sent=0,ack,i;
    printf("enter window size\n");
    scanf("%d",&window size);
    while(1)
    {
        for( i = 0; i < window size; i++)
        {
            printf("Frame %d has been transmitted.\n",sent);
            sent++;
            if(sent == window size)
                break;
        }
        printf("\nPlease enter the last Acknowledgement received.\n");
        scanf("%d",&ack);
        if(ack == window size)
            break;
        else
            sent = ack;
    }
    return 0;
}
```

OUTPUT:

enter window size:8

Frame 0 has been transmitted.

Frame 1 has been transmitted.

Frame 2 has been transmitted.
 Frame 3 has been transmitted.
 Frame 4 has been transmitted.
 Frame 5 has been transmitted.
 Frame 6 has been transmitted.
 Frame 7 has been transmitted.
 Please enter the last Acknowledgement received: 2
 Frame 2 has been transmitted.
 Frame 3 has been transmitted.
 Frame 4 has been transmitted.
 Frame 5 has been transmitted.
 Frame 6 has been transmitted.
 Frame 7 has been transmitted.

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

Theory:

Dijkstra's algorithm is a non-adaptive routing algorithm which is very widely used to route packets from source to destination through various routers available during the transmission. It is implemented at the network layer of the architecture where data packets are sent through routers which maintain routing tables that help to denote the exact location to where the destined packets need to be delivered. Major advantage in using Dijkstra's algorithm is that it forwards the data packets from source to destination through the most optimized path in terms of both the distance and cost observed. It prompts the user to enter the number of nodes and the source and destination nodes among them. In addition, the algorithm written below also asks for the neighbors to each node with the distances to reach to them from each node is also prompted. All this data is stored and used further to calculate and estimate the best path possible for data packets to reach their destination from source.

PROGRAM:

```

/* Dijkstra's algorithm*/

#include<stdio.h>
#include<conio.h>
void dij(int n,int v,int cost[10][10],int dist[]);
void main()
{
    int n,v,i,j,cost[10][10],dist[10];
    clrscr();
    printf("*** Dijkstra Algorithm ***\n");
    printf("Enter the total number of Nodes.\t");
    scanf("%d",&n);
    printf("Enter the cost matrix.\n");
  
```

```

for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
    scanf("%d",&cost[i][j]);
    if(cost[i][j]==0)
        cost[i][j]=999;
}
printf("Enter the source Vertex.\t");
scanf("%d",&v);
dij(n,v,cost,dist);
printf("\nShortest path:\n");
for(i=1;i<=n;i++)
if(i!=v)
printf("V%d->V%d,cost=%d\n",v,i,dist[i]);
getch();
}
void dij(int n,int v,int cost[10][10],int dist[])
{
    int i,u,count,w,flag[10],min;
    for(i=1;i<=n;i++)
    flag[i]=0,dist[i]=cost[v][i];
    count=2;
    while(count<=n)
    {
        min=99;
        for(w=1;w<=n;w++)
        if(dist[w]<min && !flag[w])
        min=dist[w],u=w;
        flag[u]=1; count++;
        for(w=1;w<=n;w++)
        if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
        dist[w]=dist[u]+cost[u][w];
    }
}
}

```

OUTPUT:


```

*** Dijkstra Algorithm ***
Enter the total number of Nodes.      5
Enter the cost matrix.
0 10 3 0 0
0 0 1 2 0
0 4 0 8 2
0 0 0 0 7
0 0 0 9 0
Enter the source Vertex.      1

Shortest path:
U1->U2,cost=7
U1->U3,cost=3
U1->U4,cost=9
U1->U5,cost=5

```

7. Write a program to implement congestion control in network layer using leaky bucket algorithm

Theory:

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 simulated 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.

Program for Leaky bucket:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int incoming, outgoing, buck_size, n, store = 0;
    printf("Enter bucket size, outgoing rate and no of inputs: ");
    scanf("%d %d %d", &buck_size, &outgoing, &n);
    while (n != 0)
    {
        printf("Enter the incoming packet size : ");
        scanf("%d", &incoming);
        printf("Incoming packet size %d\n", incoming);
        if (incoming <= (buck_size - store))
        {
            store += incoming;
            printf("Bucket buffer size %d out of %d\n", store, buck_size);
        }
        else
        {

```

```

        printf("Dropped %d no of packets\n", incoming - (buck_size - store));
        printf("Bucket buffer size %d out of %d\n", store, buck_size);
        store = buck_size;
    }
    store = store - outgoing;
    printf("After outgoing %d packets left out of %d in buffer\n", store, buck_size);
    n--;
}
getch();
}

```

Output:-

```

Enter bucket size, outgoing rate and no of inputs: 10 8 2
Enter the incoming packet size : 9
Incoming packet size 9
Bucket buffer size 9 out of 10
After outgoing 1 packets left out of 10 in buffer
Enter the incoming packet size : 7
Incoming packet size 7
Bucket buffer size 8 out of 10
After outgoing 0 packets left out of 10 in buffer

```

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

Theory

IP addressing is the allocation of unique ID to each and every system connected in a network to maintain communication among them through out the affixed network. There are 5 classes of IP Addresses namely A through E with the range varying from one class to the other class.

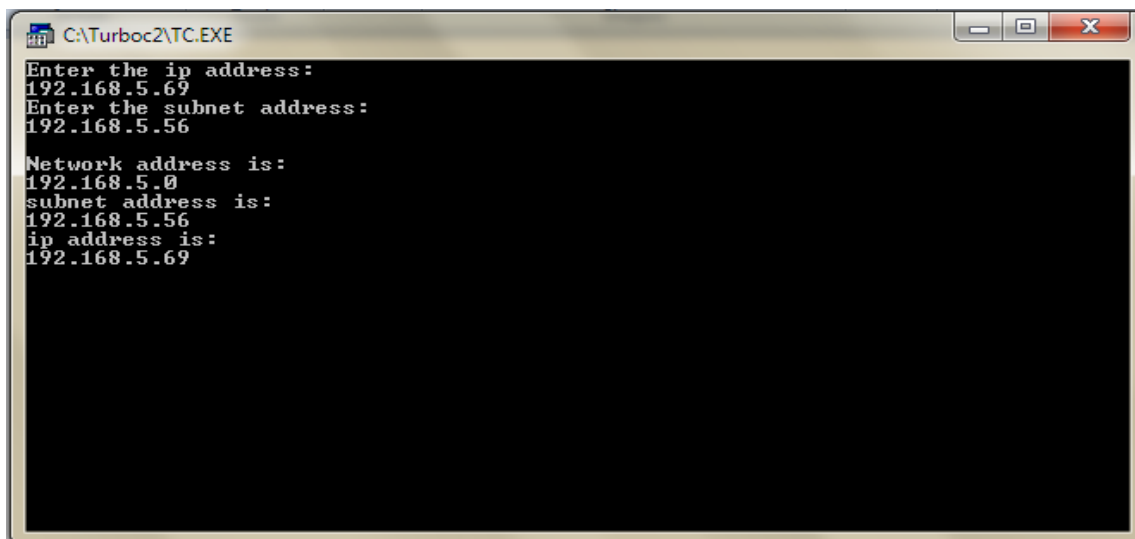
A subnet is a network allocation to similar systems or same hierarchial systems present in a allocated network like an organisation. Each and every system can be reachd through a client-server computing environment where the server acts as the Master and the clients acts as the Slaves to form a Master-Slave computing environment.

PROGRAM:

A) NETWORK ADDRESS:

```
#include<stdio.h>
void main()
{
    unsigned int compad[4];
    unsigned int mask[4];
    unsigned int netadr[4];
    int i;
    clrscr();
    printf("Enter the ip address:\n");
    scanf("%u%*c%u%*c%u%*c%u%*c",&compad[3],&compad[2],&compad[1],&compad[0]);
    printf("Enter the subnet address:\n");
    scanf("%u%*c%u%*c%u%*c%u%*c",&mask[3],&mask[2],&mask[1],&mask[0]);
    for(i=0;i<4;i++)
    {
        netadr[i]= compad[i]&mask[i];
    }
    printf("\nNetwork address is:\n");
    printf("%u.%u.%u.%u",netadr[3],netadr[2],netadr[1],netadr[0]);
    printf("\nsubnet address is:\n");
    printf("%u.%u.%u.%u",mask[3],mask[2],mask[1],mask[0]);
    printf("\nip address is:\n");
    printf("%u.%u.%u.%u",compad[3],compad[2],compad[1],compad[0]);
    getch();
}
```

OUTPUT:

A screenshot of a TurboC2 console window titled "C:\Turboc2\TC.EXE". The window has a black background with white text. The text shows the program's execution: it prompts for an IP address (192.168.5.69) and a subnet address (192.168.5.56), then displays the calculated network address (192.168.5.0), subnet address (192.168.5.56), and the original IP address (192.168.5.69).

```
C:\Turboc2\TC.EXE
Enter the ip address:
192.168.5.69
Enter the subnet address:
192.168.5.56

Network address is:
192.168.5.0
subnet address is:
192.168.5.56
ip address is:
192.168.5.69
```

B) NETWORK ADDRESS WITH AUTOMATIC SUBNET ADDRESS GENERATION:


```
#include <studio.h>
#include<coinio.h>
```

```

void main()
{
    unsigned int compad[4];
    unsigned int mask[4];
    unsigned int netadr[4];
    unsigned long int ma=0;
    int i,pre;
    clrscr();
    printf("Enter the ip address:\n");
    scanf("%u%*c%u%*c%u%*c%u%*c",&compad[3],&compad[2],&compad[1],&compad[0]);
    printf("Enter the prefix:\n");
    scanf("%ul",&pre);
    for(i=(32-pre);i<32;i++)
    ma=ma|(1<<i);
    for(i=0;i<4;i++)
    {
        mask[i]=ma%256;
        ma=ma/256;
    } for(i=0;i<4;i++)
    {
        netadr[i]= compad[i]&mask[i]; }
    printf("\nNetwork address is:\n");
    printf("%u.%u.%u.%u",netadr[3],netadr[2],netadr[1],netadr[0]);
    printf("\nsubnet address is:\n");
    printf("%u.%u.%u.%u",mask[3],mask[2],mask[1],mask[0]);
    printf("\nip address is:\n");
    printf("%u.%u.%u.%u",compad[3],compad[2],compad[1],compad[0]);
    getch();
}

```

OUTPUT:

A screenshot of a TurboC++ console window titled "C:\Turboc2\TC.EXE". The window has a black background with white text. The text shows a sequence of prompts and user inputs for a subnetting calculation. The prompts are "Enter the ip address:", "Enter the prefix:", "Network address is:", "subnet address is:", and "ip address is:". The user inputs are "192.168.5.69", "24", "192.168.5.0", "255.255.255.0", and "192.168.5.69_".

```
C:\Turboc2\TC.EXE
Enter the ip address:
192.168.5.69
Enter the prefix:
24

Network address is:
192.168.5.0
subnet address is:
255.255.255.0
ip address is:
192.168.5.69_
```

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

Theory:

Distance Vector routing (DVR) algorithm is unlike Dijkstra's algorithm which is a non-adaptive routing algorithm and means that it is purely static, that is pre-destined and fixed, not flexible in networks where congestions are more prone to occur. DVR is an adaptive routing algorithm in which the information from neighbours is maintained well by each and every node and this helps us to determine the simplest path possible in a changing network. Though, one of the node may fail, still, the destined node is reachable through other possible intermediate nodes that are found out by the DVR algorithm.

PROGRAM:

```
#include<stdio.h>
struct node
{
    unsigned dist[20];
    unsigned from[20];
}rt[10];
int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes);//Enter the nodes
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
```

```

{
scanf("%d",&costmat[i][j]);
costmat[i][i]=0;
rt[i].dist[j]=costmat[i][j]; //initialise the distance equal to cost matrix
rt[i].from[j]=j;
}
}
do
{
count=0;
for(i=0;i<nodes;i++) //We choose arbitrary vertex k and we calculate the direct distance from the node i to
k using the cost matrix
//and add the distance from k to node j
for(j=0;j<nodes;j++)
for(k=0;k<nodes;k++)
if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
{//We calculate the minimum distance
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
rt[i].from[j]=k;
Count++;
}}
while(count!=0);
for(i=0;i<nodes;i++)
{
printf("\n\n For router %d\n",i+1);
for(j=0;j<nodes;j++)
{
printf("\tnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}}
printf("\n\n");
getch();
}

```

OUTPUT:

Enter the number of nodes :

3

Enter the cost matrix :

0 2 7

2 0 1

7 1 0

For router 1

node 1 via 1 Distance 0

node 2 via 2 Distance 2

node 3 via 3 Distance 3

For router 2
node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1
For router 3
node 1 via 1 Distance 3
node 2 via 2 Distance 1
node 3 via 3 Distance 0

ADDITIONAL EXPERIMENTS:

10. To write a C program to develop a DNS client server to resolve the given hostname.

PROGRAM:

```
#include <stdio.h>
#include <coinio.h>
int main(int argc, char *argv[1])
{
    struct hostent *hen; if(argc!=2)
    {
        fprintf(stderr, "Enter the hostname \n"); exit(1);
    }
    hen=gethostbyname(argv[1]);
    if(hen==NULL){ fprintf(stderr, "Host not found \n");
    }
    printf("Hostname is %s \n", hen->h_name);
    printf("IP address is %s \n", inet_ntoa(((struct in_addr *)hen->h_addr)));
}
```

OUTPUT:

Hostname is www.google.com
IP Address is 173.194.73.99.

11. Write a program for Hamming Code generation for error detection and correction

Theory:

Hamming codes are used for detecting and correcting single bit errors in transmitted data. This requires that 3 parity bits (check bits) be transmitted with every 4 data bits. The algorithm is called A(7, 4) code, because it requires seven bits to encode 4 bits of data.

PROGRAM


```

#include<stdio.h>
#include<conio.h>
char data[5];
int encoded[8], edata[7], syndrome[3];
int hmatrix[3][7]= { 1,0,0,0,1,1,1,0,1,0,1,0,1,1,0,0,1,1,1,0,1};
char gmatrix[4][8]={ "0111000", "1010100", "1100010", "1110001"};
void main()
{
int i,j;
clrscr();
cout<<"Hamming Code --- Encoding\n";
cout<<"Enter 4 bit data : ";
cin>>data;
cout<<"Generator Matrix\n";
for(i=0;i<4;i++) cout<<"\t"<<gmatrix[i]<<"\n";
cout<<"Encoded Data : ";
for(i=0;i<7;i++) {
for(j=0;j<4;j++)
encoded[i]+=((data[j]- '0')*(gmatrix[j][i]- '0'));
encoded[i]=encoded[i]%2;
cout<<encoded[i]<<" ";
}
cout<<"\nHamming code --- Decoding\n";
cout<<"Enter Encoded bits as received : ";
for(i=0;i<7;i++) cin>>edata[i];
for(i=0;i<3;i++) {
for(j=0;j<7;j++)
syndrome[i]=syndrome[i]+(edata[j]*hmatrix[i][j]);
syndrome[i]=syndrome[i]%2;
}
for(j=0;j<7;j++)
if ((syndrome[0]==hmatrix[0][j])&&(syndrome[1]==hmatrix[1][j])&&
(syndrome[2]==hmatrix[2][j]))
break;
if(j==7)
cout<<"Data is error free!!\n";
else {
cout<<"Error received at bit number "<<j+1<<" of the data\n";
edata[j]=!edata[j];
cout<<"The Correct data Should be : ";
for(i=0;i<7;i++) cout<<edata[i]<<" ";
}
}
}
}

```

OUTPUT:

Hamming Code --- Encoding

Enter 4 bit data : 1 0 1 0

Generator Matrix

0111000

1010100

1100010

1110001

Encoded Data : 1 0 1 1 0 1 0

Hamming code --- Decoding

Enter Encoded bits as received : 1 0 1 1 0 1 1

Error received at bit number 7 of the data

The Correct data Should be : 1 0 1 1 0 1 0

Hamming Code --- Encoding

Enter 4 bit data : 1 0 1 0

Generator Matrix

0111000

1010100

1100010

1110001

Encoded Data : 1 0 1 1 0 1 0

Hamming code --- Decoding

Enter Encoded bits as received : 1 0 1 1 0 1 0

Data is error free!!

12. Implement the above program using as message queues or FIFO as IPC channels

PROGRAM:

```
/*Server*/
```

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666
char fname[256];
int main()
{
    int readfd, writefd, fd;
    ssize_t n;
    char buff[512];
```

```

if (mkfifo(FIFO1, PERMS)<0)
printf("Cant Create FIFO Files\n");
if (mkfifo(FIFO2, PERMS)<0)
printf("Cant Create FIFO Files\n");
printf("Waiting for connection Request..\n");
readfd =open(FIFO1, O_RDONLY, 0);
writefd=open(FIFO2, O_WRONLY, 0);
printf("Connection Established..\n");
read(readfd, fname, 255);
printf("Client has requested file %s\n", fname);
if ((fd=open(fname,O_RDWR))<0) {
strcpy(buff,"File does not exist..\n");
write(writefd, buff, strlen(buff));
}
else
{
while((n=read(fd, buff,512))>0)
write(writefd, buff, n);
}
close(readfd);
unlink(FIFO1);
close(writefd);
unlink(FIFO2);
}

/*Client*/

#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666
char fname[256];
int main()
{
ssize_t n;
char buff[512];
int readfd,writefd;
printf("Trying to Connect to Server..\n");
writefd = open(FIFO1, O_WRONLY, 0);
readfd = open(FIFO2, O_RDONLY, 0);
printf("Connected..\n");
printf("Enter the filename to request from server: ");
scanf("%s",fname);

```

```
write(writefd, fname, strlen(fname));
printf("Waiting for Server to reply..\n");
while((n=read(readfd,buff,512))>0)
write(1,buff,n);
close(readfd);
close(writefd);
return 0;
}
```

OUTPUT (SERVER):

```
[root@localhost CN Lab] ./s.o
Waiting for connection Request..
Connection Established..
Client has requested file alpha
[root@localhost CN Lab]
```

OUTPUT (CLIENT):

```
[root@localhost CN Lab] ./c.o
Trying to Connect to Server..
Connected..
Enter the filename to request from server: alpha
Waiting for Server to reply..
This a demo of client server using Sockets
Just for trial.
Now End of file
[root@localhost CN Lab]
```