Output :-

```
enter data : 1011101
- - - - - - - - - - -
generating polynomial : 10001000000010000 1
- - - - - - - - - - - - -
mod-ified data is : 10111010000000000000000
- - - - - - - - - - - - - -
checksom is : 1000101101011000
- - - - - - - - - - - - - - - -
final codeword is : 1011011000101101011000
- - - - - - - - - - - - - - -
test error detection 0 (yes) 1 (no) ? : 0
enter the position where error is to be inserted : 3
- - - - - - - - - - - - - -
erroneous data : 1001101 1000 1011 01011000
error detected
```

Date 20-12-21

Expt. No. 7 Page No. 11

Experiment - 7

AIM :- Write a program for error detecting code using CRC - CCITT (16-bits)

```
#include <stdio.h>
#include <string.h>
#define N strlen(gen)
char modif[28], checksum[28], gen[28];
int a,c,e,b;
void xor()
{
for(c=1;c<N;c++)
checksum[c] = ((checksum[c]==gen[c])?'0':'1');
}
void crc()
{
for(e=0;e<N;e++)
checksum[e]=modif[e];
do
{
if(checksum[0]=='1')
xor();
for(c=0;c<N-1;c++)
checksum[c]=checksum[c+1];
checksum[c]=modif[e++];
}while(e<=a+N-1);
}
```

Teacher's Signature : _____

```
int main()
{
int flag=0;
strcpy(gen,"10001000000010000 1");
printf("\n enter data");
scanf("%s",modif);
printf("\n ----------\n");
printf("\n generating polynomials : %s",gen);
a=strlen(modif);
for(e=a;e<a+N-1;e++)
modif[e]='0';
printf("\n ----------\n");
printf(" mod-ified data is : %s",modif);
printf("\n----------\n");
crc();
printf(" checksum is : %s",checksum);
for(e=a;e<a+N-1;e++)
modif[e]=checksum[e-a];
printf("\n ----------\n");
printf("\n final codeword is : %s",modif);
printf("\n ----------\n");
printf("\n test error detected 0(yes) 1(no)?: ");
scanf("%d",&e);
if(e==0)
{
do{
printf("\n enter the position where error is to be inserted:");
scanf("%d",&e);
```

Date _____

Expt. No. 7

Page No. 13

```
}
while(e==0 || e>a+N-1);
modif[e-1] = (modif[e-1]=='0')? '1' : '0';
printf("\n-----------\n");
printf("\n erroneous data: %s \n", modif);
}

crc();
for(e=0; (e<N-1) && (checksum[e]!='1'); e++);
if(e<N-1)
printf("error detected \n\n");
else
printf("\n error detected \n\n");
printf("\n ----------");
}
```

Teacher's Signature : _____

OUTPUT :-

Enter the no of nodes required = 3

Enter the dists from the nodes 1 to other node...
Pls enter 999 if there is no direct

Enter dist to node 2 = 100
Enter dist to node 3 = 999

Enter the dists from the nodes 2 to other node..
Pls enter 999 if there is no direct

Enter dist to node 1 = 999
Enter dist to node 3 = 15

Enter the dists from the nodes 3 to other node...
Pls enter 999 if there is no direct

Enter dist to node 1 = 20
Enter dist to node 2 = 25

The configuration of the nodes after initialization is as follows:
The routing table for node no 1 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	0	1
2	10	2
3	NO LINK	NO HOP

The routing table for node no 2 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	NO LINK	NO HOP
2	0	2
3	15	3

The routing table for node no 3 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	20	1
2	25	2
3	0	3

The config of the nodes after the comp of the paths is as follows:
The routing table for node no. 1 is as follows,

DESTINATION	DISTANCE	NEXT_HOP
1	0	1
2	10	2
3	25	2

The routing table for node no 2 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	35	3
2	0	2
3	15	3

The routing table for node no 3 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	20	1
2	25	2
3	0	3

Enter 0 to exit or any other key to find the shortest path : 1

Enter the nodes btn which path is to be found : 1 3

The most suitable route from node 1 to 3 is as follows

1 --> 2 --> 3

The length of the shortest path between node 1 & 3 is 25

Date 20-12-21

Expt. No. 8 Page No. 14

Experiment - 8

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

```
#include<stdio.h>
#include<stdlib.h>
#define INFINITY 1000
#define NODES 10
struct node
{
int dist[NODES][3];
};
struct node n[NODES];
typedef struct node NOD;
int main()
{
void init(int, int);
void inp(int, int);
void callee(int, int);
void bf1(int, int, int);
void find(int, int);
int i,j,x,y,ne;
do{
printf("\n Enter the no. of nodes required:");
scanf("%d", &ne);
}while(ne>10||ne<0);
```

Teacher's Signature :

```
for(i=0; i<no; i++)
{
 init(no,i);
 inp(no,i);
}
printf("\n The configuration of the nodes after initialization is as follows:");
for(i=0; i<no; i++)
op1(no,i,0);
for(j=0; j<no; j++)
{
for(i=0; i<no; j++)
caller(no,i);
}
printf("\n The config of the nodes after the comp the paths is as follows:");
for(i=0; i<no; i++)
op1(no,i,1);
while(1)
{
printf("\n Enter 0 to exit or any other key to find the shortest path:");
scanf("%d",&j);
if(!j)
break;
do{
printf("\n Enter the nodes btn which path is to be found:");
scanf("%d%d",&x,&y);
} while((x<0||x>no) && (y<0||y>no));
if(flag) printf("\n The most suitable route from node %d to %d is as follows\n
                                                    ",x,y);
```

Teacher's Signature : _____

```
find(x,y);
printf("%d",y);
printf("\n The length of the shortest path between node %d & %d is %d",
                                        x,y,n[x-1].t[y-1][2]);
}
}
void init(int no, int x)
{
int i;
for(i=0;i<no;i++){
n[x].t[i][1]=i;
n[x].t[i][2]=999;
n[x].t[i][3]=NUL;
}
n[x].t[x][2]=0;
 n[x].t[x][3]=x;
}
void inp(int no, int x)
{
int i;
printf("\n Enter the dists from nodes %d to other node...",x+1);
printf("\n Pls enter 999 if there is no direct\n");
for(i=0;i<no;i++)
{
if(i!=x)
{
do
{
```

Date

Expt. No. 8 Page No. 17

```
printf("\n Enter dist to node %d = ", i+1);
scanf("%d", &n[x].t[i][2]);
} while(n[x].t[i][2]<0 || n[x].t[i][2]>999);
if(n[x].t[i][2]!=999)
n[x].t[i][3]=1
}}
}
void caller(int no, int x)
{
void compar(int,int,int);
int i;
for(i=0; i<no; i++)
{
 if(n[x].t[i][2]!=999 && n[x].t[i][2]!=0)
{
 compar(x, i, no);
 }}}
void compar(int z, int y, int no)
{
int i, z;
for(i=0; i<no; i++)
{
z=n[x].t[y][2]+n[y].t[i][2];
if(n[x].t[i][2]>z)
{
n[x].t[i][2]=z;
n[x].t[i][3]=y;
}}}
```

Teacher's Signature :

```
void sht(int no, int x, int z)
{
int i, j;
printf("\n The routing table for node no %d is as follows", x+1);
printf("\n\n\t\t\t DESTINATION \t DISTANCE \t NEXT_HOP");
for(i=0; i<no; i++){
if((z == 999 && n[x].t[i][2]>=999) || (n[x].t[i][2]>=(999*no)))
printf("\n\t\t\t %d \t NO LINK \t NO HOP", n[x].t[i][1]+1);
else
if(n[x].t[i][3]==NULL)
printf("\n\t\t\t %d \t\t %d \t\t NO HOP", n[x].t[i][1]+1, n[x].t[i][2]);
else
printf("\n\t\t\t %d \t\t %d \t\t %d", n[x].t[i][1]+1, n[x].t[i][2],
                                        n[x].t[i][3]+1);

}}
void find(int x, int y)
{
int i,j;
i=x-1;
j=y-1;
printf("%d --> ", x);
if(n[i].t[j][3]!=j)
{
find(n[i].t[j][3]+1, y);
return;
}}
```

Teacher's Signature : _____

OUTPUT:-

Distance Matrix (4x4 , max distance /infinity is 99):
12 43 54 67
32 45 67 43
23 23 23 43
1 2 34 4
Enter the source vertex : (0-3)
0

vertex	Distance	Path
0 → 1	43	0 1
0 → 2	54	0 2
0 → 3	67	0 3

Date _____

Expt. No. 9 Page No. 19

Experiment-9:-

AIM: Implement Dijkstra's algorithm to compute the shortest path for a given topology.

```
#include <bits/stdc++.h>
using namespace std;
#define V 4
int minDistance(int dist[], bool sptSet[]){
    int min = 9999, min_index;
    for(int v = 0; v < V; v++)
        if(sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
    return min_index;
}
void printPath(int parent[], int j){
    if (parent[j] == -1)
        return;
    printPath(parent, parent[j]);
    cout << j << " ";
}
void printSolution(int dist[], int n, int parent[]){
    int src = 0;
    cout << "Vertex \t Distance \t Path" << endl;
    for(int i = 1; i < V; i++){
     cout << "\n"
        << src << " -> " << i << " \t \t " << dist[i] << " \t \t"
```

Teacher's Signature : _____

```
        << src << " ";
    printPath(parent, j);
    }
}
void dijkstra( int graph[V][V], int src) {
    int dist[V];
    bool sptSet[V];
    int parent[V];
    for (int i = 0; i< V; i++){
      parent[i] = -1;
      dist[i]= 9999;
      sptSet[i] = false;
    }
  dist[src] = 0;
  for (int count = 0; count < V - 1; count++){
      int u = minDistance(dist, sptSet);
      sptSet[u] = true;
      for (int v = 0; v < V; v++)
          if (!sptSet[v] && graph[u][v] && dist[u] + graph[u][v]
            < dist[v]){
            parent[v] = u;
            dist[v] = dist[u] + graph[u][v];
          }
  }
  printSolution(dist, V, parent);
}
int main(){
  int graph[V][V];
```

Teacher's Signature : _____

```
cout<< "Distance Matrix (" << V << "x" << V << ", max distance (infinity is
    99): " <<endl;
for (int i = 0 ; i< V ; i++) {
    for ( int j = 0; j<V ; j++)
        cin >> graph[i][j];
}
cout << "Enter the source vertex : (0-" << V-1 << ")," << endl;
int src;
cin >> src;

dijkstra(graph, src);
cout << endl;
return 0;
}
```

Teacher's Signature : _____

OUTPUT :-

packet[0] : 83 bytes
packet [1] : 86 bytes
packet [2] : 77 bytes
packet [3] : 15 bytes
packet [4] : 93 bytes
Enter the Outgoing rate : 30
Enter the Bucket size : 85

Incoming Packet size : 83
Bytes remaining to Transmit : 83
Packet of size 30 Transmitted --- Bytes Remaining to Transmit : 53
Packet of size 30 Transmitted --- Bytes Remaining to Transmit : 23
Packet of size 23 Transmitted --- Bytes Remaining to Transmit : 0

Incoming packet size (86 bytes) is Greater than bucket capacity (85 bytes) - PACKET REJECTED

Incoming Packet size :- 77
Bytes remaining to Transmit : 77
Packet of size 30 Transmitted --- Bytes Remaining to Transmit : 47
Packet of size 30 Transmitted --- Bytes Remaining to Transmit : 17
Packet of size 17 Transmitted --- Bytes Remaining to Transmit : 0

Incoming Packet size : 15
Bytes remaing to Transmit : 15
Packet of size 15 Transmitted --- Bytes Remaining to Transmit : 0

Incoming packet size (93 bytes) is Greater than bucket capacity (85 bytes) - PACKET REJECTED

Date _____

Expt. No. 10 Page No. 22

Experiment - 10 :-

AIM :- Write program for congestion control using Leaky_bucket algorithm.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define NOF_PACKETS 5
/*
int rand (int a)
{
int rn = (random() % 10) % a;
return rn == 0 ? 1 : rn;
}
*/
/*
#include<stdlib.h>
long int random(void);
The random() function uses a nonlinear additive feedback random number
generator employing a default table of size 31 long integers to return successive
pseudo-random numbers in the range from 0 to RAND_MAX. The period
of this random number generator is very large, approximately 16 * ((2^31)-1).
*/
int main() {
  int packet_sz [~~NOF_PACKETS~~][NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0,
  p_sz, p_time, op;
  for (i=0; i<NOF_PACKETS; ++i)
```

Teacher's Signature : _____

Date

Expt. No. 10 Page No. 23

```
    packet_sz[i] = random() % 100;
for(i=0; i<NOF_PACKETS; ++i)
    printf("\n packet[%d]: %d bytes \t", i, packet_sz[i]);
printf("\n Enter 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\n Incoming packet size (%d bytes) is Greater than bucket
      capacity (%d bytes)-PACKET REJECTED", packet_sz[i], b_size);
     else
       printf("\n\n Bucket capacity exceeded-PACKETS REJECTED!!");
     else
     {
      p_sz_rm += packet_sz[i];
      printf("\n\n Incoming Packet size: %d", packet_sz[i]);
      printf("\n Bytes remaining to Transmit: %d", p_sz_rm);
     //p_time = random() * 10;
     //printf("\n Time left for transmission: %d units", p_time);
     //for(clk=10; clk<=p_time; clk+=10)
      while(p_sz_rm>0)
      {
        sleep(1);
        if(p_sz_rm)
          {
```

Teacher's Signature : _____

```
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("\n Packet of size %d Transmitted", op);
    printf("---- Bytes Remaining to Transmit : %d", p_sz_rm);
}
else
{
    printf("\n No packets to transmit!!");
}
}
}
}
}
```

OUTPUT :-

The server is ready to ~~receive~~ receive

Enter file name : ServerTCP.py

From Server :

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serversocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file = open(sentence, "r")
    l = file.read(1024)

    connectionSocket.send(l.encode())
    print('\n Sent contents of ' + sentence)
    file.close()
    connectionsocket.close()
```

The server is ready to receive

Sent contents of ServerTCP.py

The server is ready to receive.

Date .....

Expt. No. 11 Page No. 25

Experiment : 11

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

Client TCP.py

```
from Socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\n Enter file name :")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print('\n From Server: \n')
print(filecontents)
clientSocket.close()
```

ServerTCP.py

```
from socket import*
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
```

Teacher's Signature : _____

Date _____

Expt. No. 11 Page No. 26

```
while 1:
    print ("The server is ready to receive")
    connection Socket, addr = server Socket. accept()
    sentence = connection Socket. recv (1024). decode()

    file = open (sentence, "r")
    l = file. read (1024)
    connection Socket. send (l. encode())
    print ("In Sent contents of ' + sentence)
    file. close()
    connection Socket. close()
```

Teacher's Signature : _____

OUTPUT :-

The Server is ready to receive
Sent contents of Server UDP.py
The Server is ready to receive

Experiment 12 :-

AIM :- Using UDP, sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Client UDP.py

```
from socket import*
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\n Enter file name :")
clientSocket.sendto(bytes(sentence, "utf-8"), (serverName, serverPort))
fileContents, serverAddress = clientSocket.recvfrom(2048)
print("\n Reply from server :\n")
print(" fileContents.decode("utf-8"))
# for i in file contents:
# print(str(i), end="")
clientSocket.close()
client.Socket.close()
```

ServerUDP.py

```
from socket import*
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
```

Teacher's Signature :

```
print("The server is ready receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    l = file.read(2048)
    serverSocket.sendto(bytes(l, "utf-8"), clientAddress)
    print("\n Sent contents of ", end = ' ')
    print(sentence)
    # for i in sentence:
        # print(str(i), end = '')
    file.close()
```

Teacher's Signature : _____