VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

TAUKSIK ANIL KUMAR (1BM20CS172)

in partial fulfilment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
October-2022 to Feb-2023

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**COMPUTER NETWORKS**" carried out by **TAUKSIK ANIL KUMAR (1BM20CS172),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **Computer Networks - (20CS5PCCON)** work prescribed for the said degree.

Dr. Rekha GS
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

Index

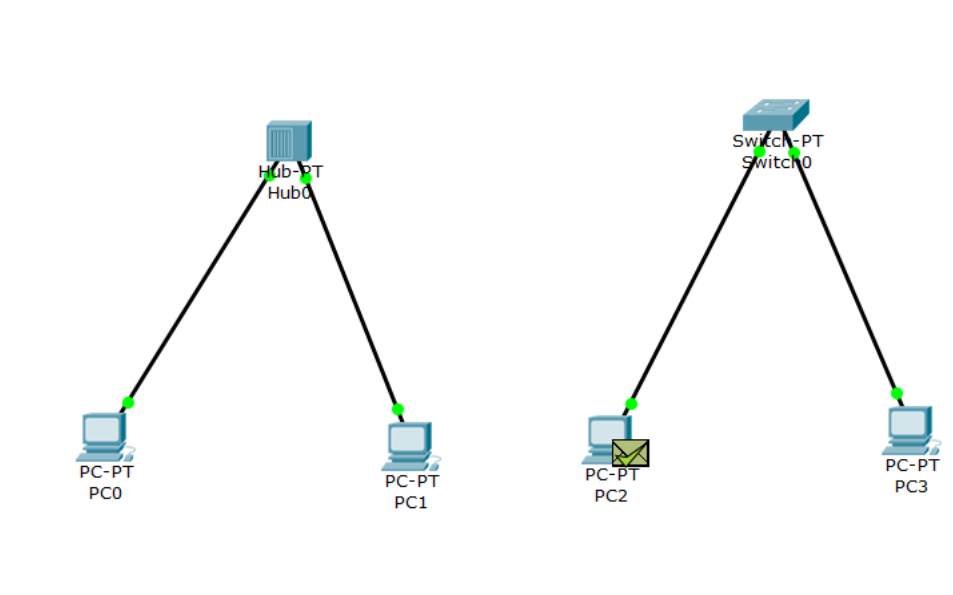
Cycle-1

Experiment 1

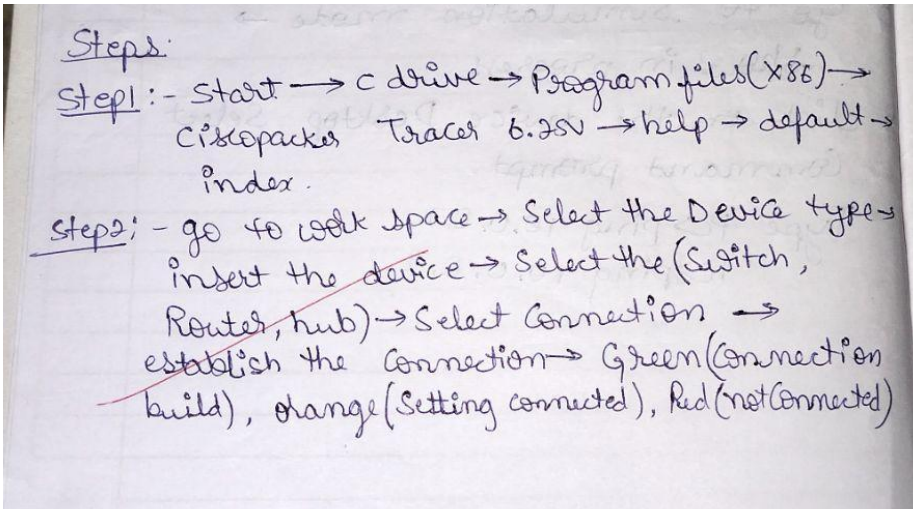
Aim of the program

Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices.

Topology



Procedure



Steps:

Step1 :- Start → c drive → Program files(x86) → Ciscopacket Tracer 6.2SV → help → default → index.

Step2; - go to work space → Select the Device type → insert the device → Select the (Switch, Router, hub) → Select Connection → establish the Connection → Green (Connection build), orange (Setting connected), Red (not Connected)

Step3:- Right click on device → Go to Fast ethernet() → Set IP address → Subnet mask address Generated
IP address like 192.168.0.104 and
192.168.0.103.

Step4:- Go to Common tool. Bar → Select packet which send to Source to destination

Step5:- place it on Source → and destination → Go to realtime → Simulation panel → click on auto Capture | play → See the Status → Successful (then Transaction is successful else failed)

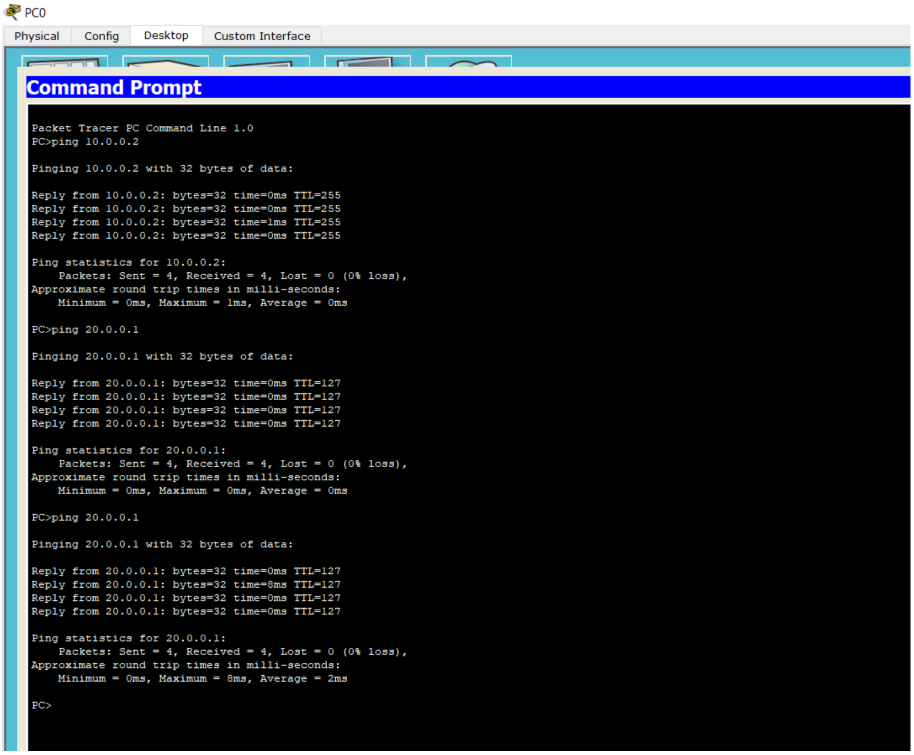
Step6:- Go to Simulation → Capture | Forward observe the moment of packets from one device (Source) to destination.

Step7:- End.

```
ping 10.0.0.10
Reply from 10.0.0.10: bytes = 32 time = 7ms TTL=128.
Reply from 10.0.0.10: bytes = 32 time = 4ms TTL=128
Reply from 10.0.0.10: bytes = 32 time = 1ms TTL=128
Reply from 10.0.0.10: bytes = 32 time = 1ms TTL=128
Packets! sent = 4; received = 4, Lost = 0 Minimum = 0ms,
maximum = 7ms, Average = 3ms.
```

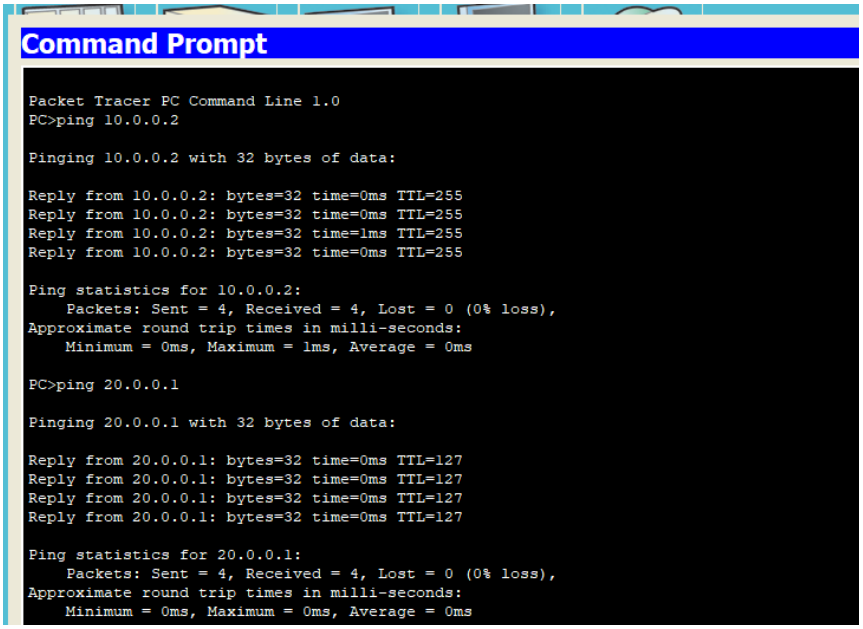
17|11|22

Output:



Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=1ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=8ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 8ms, Average = 2ms

PC>
```



Command Prompt

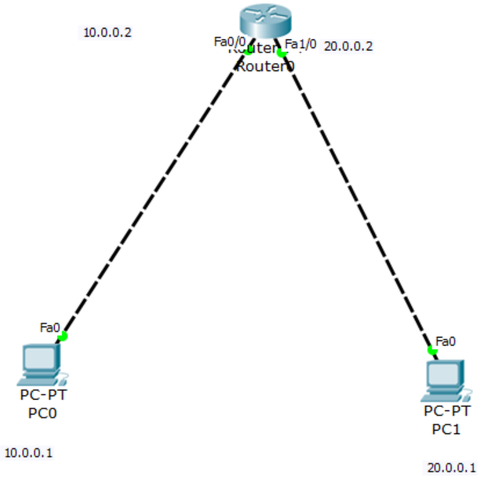
```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=1ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Experiment 2

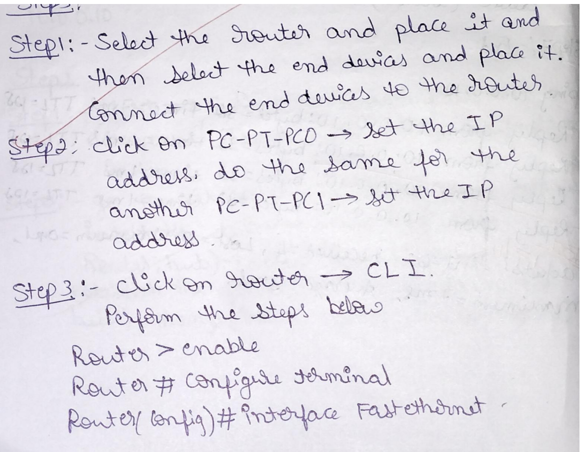
Aim of the program

Configuring IP address to Routers in Packet Tracer. Exploring the following messages: Ping Responses, Destination unreachable, Request timed out, Reply.

Topology



Procedure

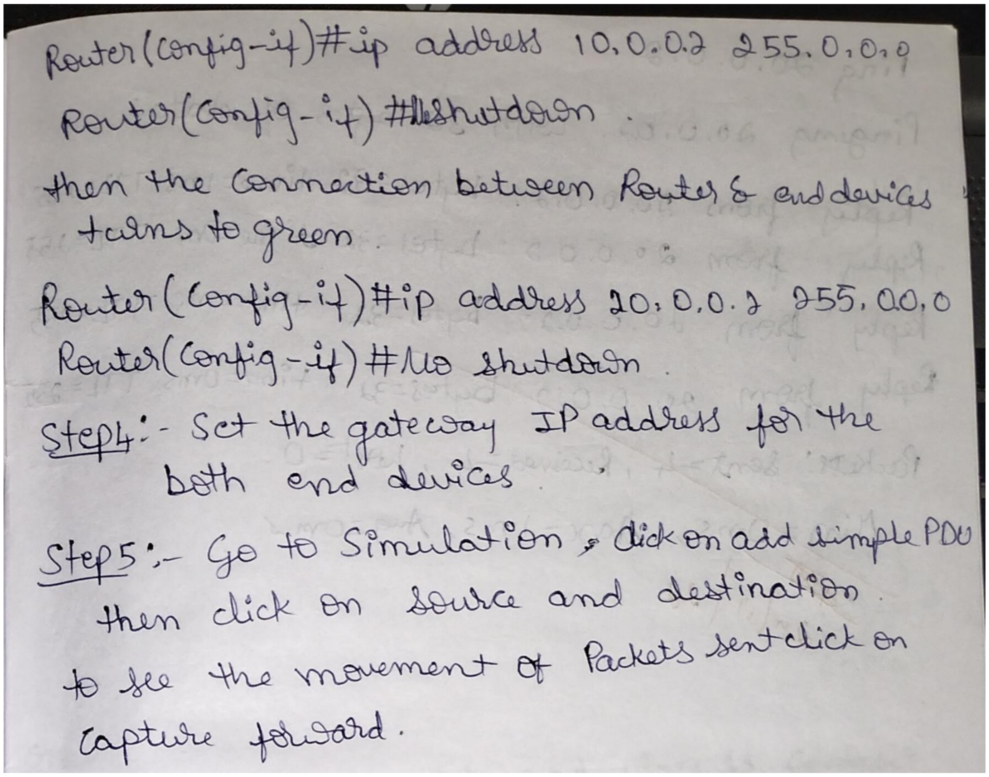


Step1:- Select the router and place it and then select the end devices and place it. Connect the end devices to the router.

Step2: click on PC-PT-PC0 → set the IP address. do the same for the another PC-PT-PC1 → set the IP address.

Step 3:- click on router → CLI.
Perform the steps below

```
Router > enable
Router # configure terminal
Router(config)# interface Fastethernet
```



Router(config-if)#ip address 10.0.0.2 255.0.0.0

Router(Config-if)#Noshutdown.

then the Connection between Router & end devices turns to green.

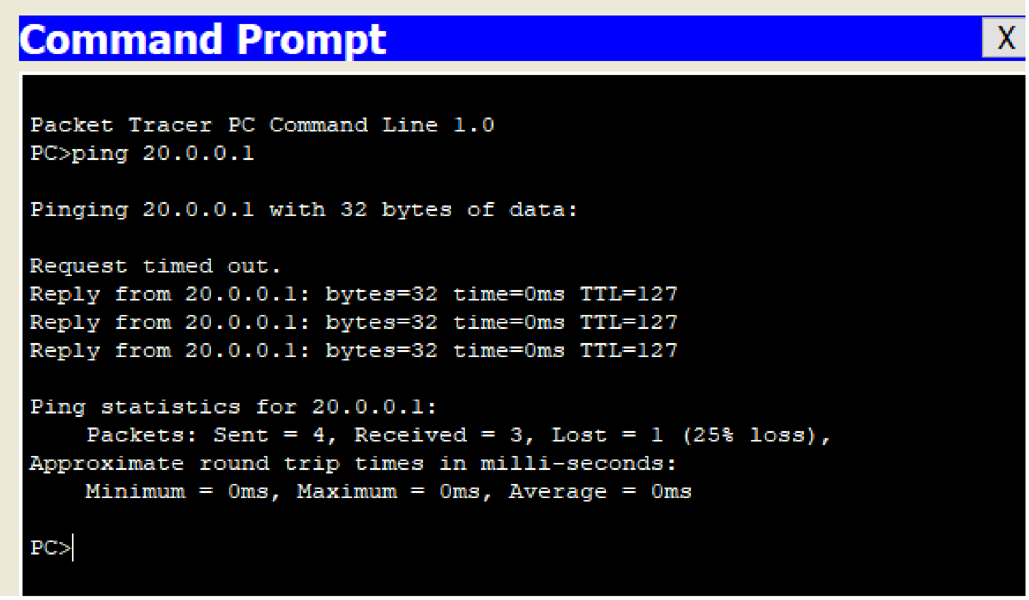
Router(Config-if)#ip address 20.0.0.2 255.0.0.0

Router(Config-if)#No shutdown.

Step4:- Set the gateway IP address for the both end devices.

Step5:- Go to Simulation, click on add simple PDU then click on source and destination. to see the movement of Packets sent click on capture forward.

Output:



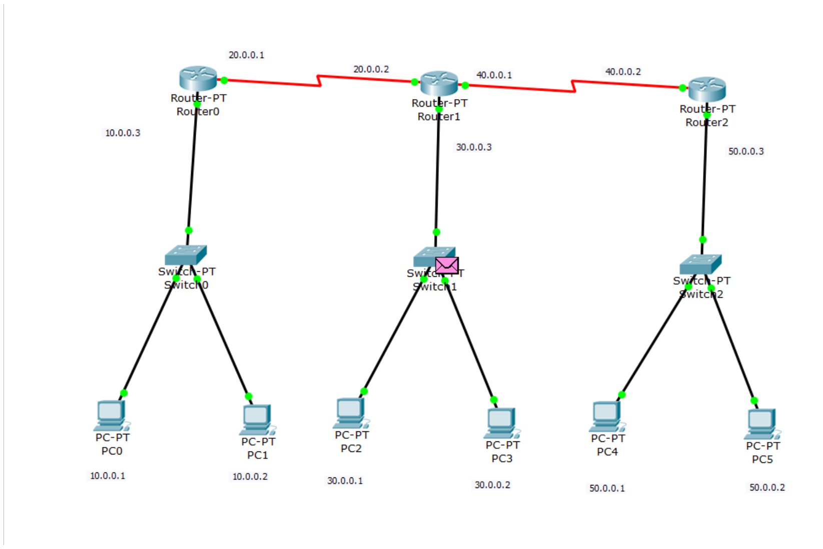
```
Packet Tracer PC Command Line 1.0
PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>
```

Experiment 3

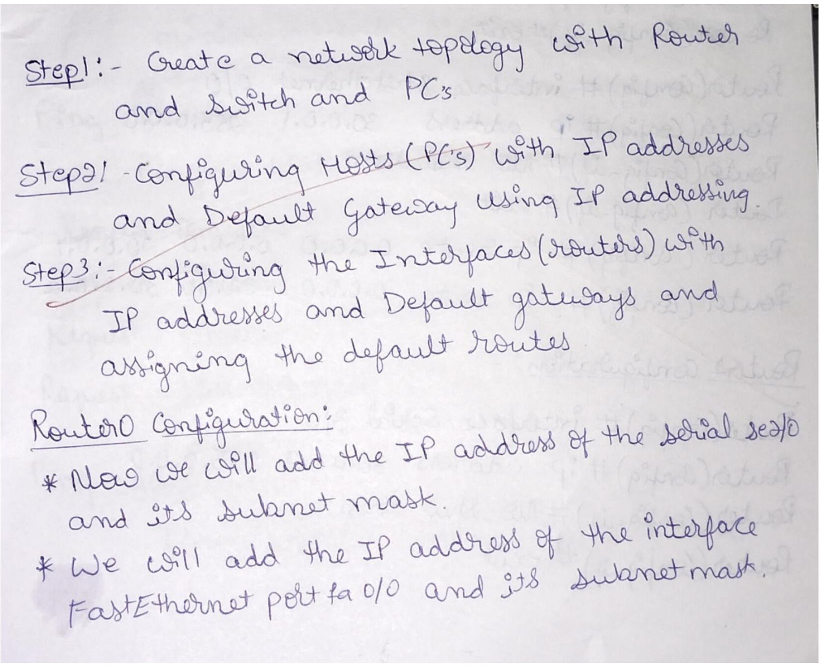
Aim of the program

Configuring default route to the Router

Topology



Procedure



Step1:- Create a network topology with Router and Switch and PC's

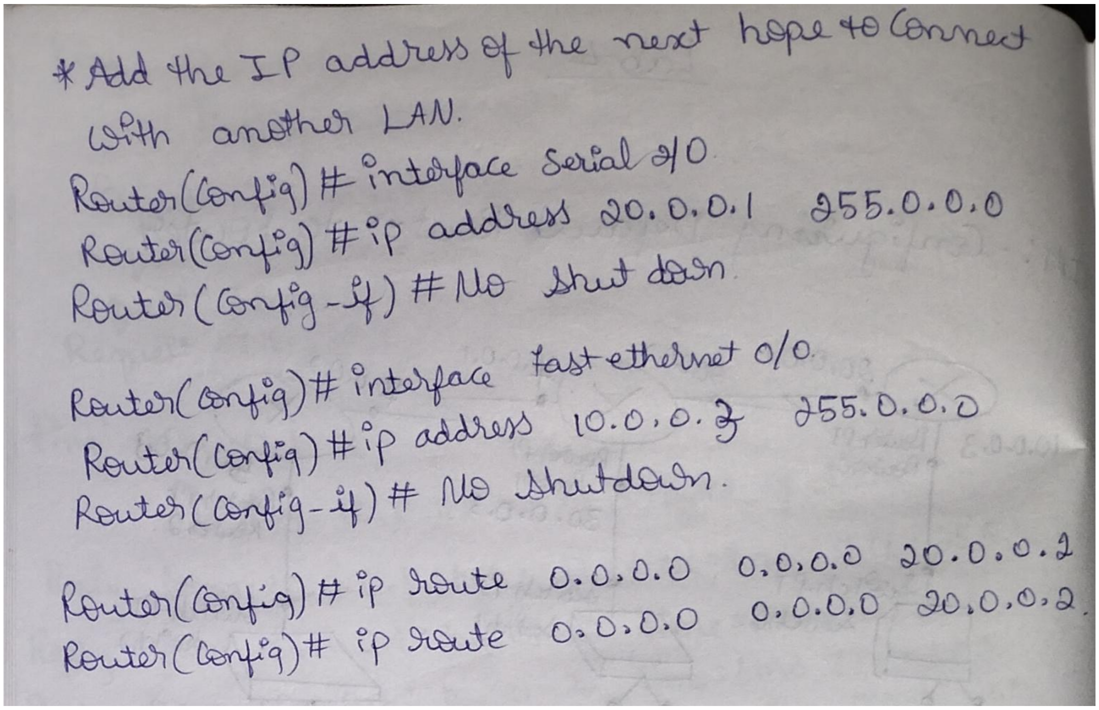
Step2:- Configuring Hosts (PC's) with IP addresses and Default Gateway using IP addressing.

Step3:- Configuring the Interfaces (routers) with IP addresses and Default gateways and assigning the default routes

Router0 Configuration:

* Now we will add the IP address of the serial se2/0 and its subnet mask.

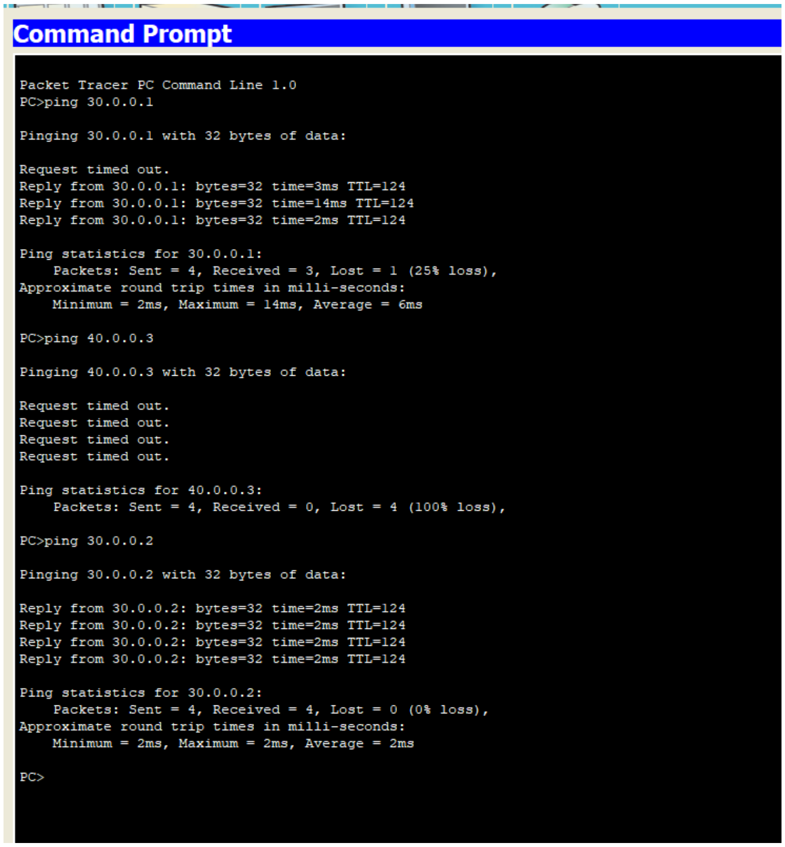
* We will add the IP address of the interface FastEthernet port fa 0/0 and its subnet mask.



* Add the IP address of the next hope to Connect with another LAN.

```
Router(Config)# interface Serial 2/0
Router(Config)# ip address 20.0.0.1  255.0.0.0
Router(Config-if)# No shut down

Router(config)# interface fastethernet 0/0.
Router(Config)# ip address 10.0.0.3  255.0.0.0
Router(Config-if)# No Shutdown.

Router(Config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2
Router(Config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2.
```

Output:



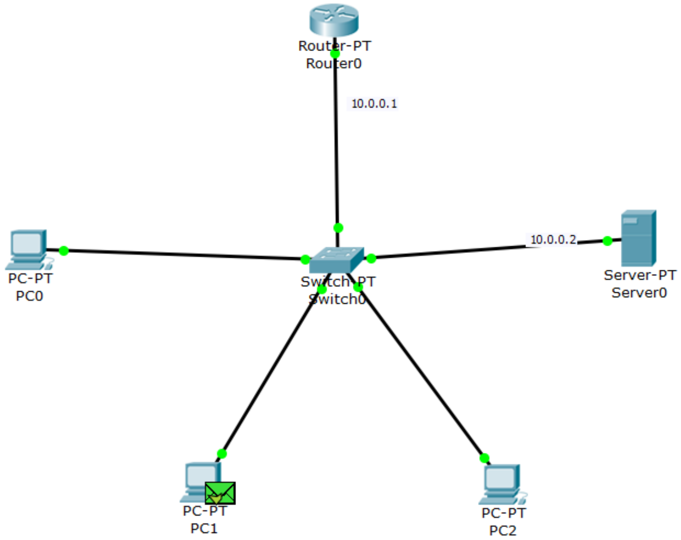
```
Packet Tracer PC Command Line 1.0
PC>ping 30.0.0.1

Pinging 30.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 30.0.0.1: bytes=32 time=3ms TTL=124
Reply from 30.0.0.1: bytes=32 time=14ms TTL=124
Reply from 30.0.0.1: bytes=32 time=2ms TTL=124

Ping statistics for 30.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 14ms, Average = 6ms

PC>ping 40.0.0.3

Pinging 40.0.0.3 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 40.0.0.3:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

PC>ping 30.0.0.2

Pinging 30.0.0.2 with 32 bytes of data:

Reply from 30.0.0.2: bytes=32 time=2ms TTL=124
Reply from 30.0.0.2: bytes=32 time=2ms TTL=124
Reply from 30.0.0.2: bytes=32 time=2ms TTL=124
Reply from 30.0.0.2: bytes=32 time=2ms TTL=124

Ping statistics for 30.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 2ms, Average = 2ms

PC>
```

Experiment 4

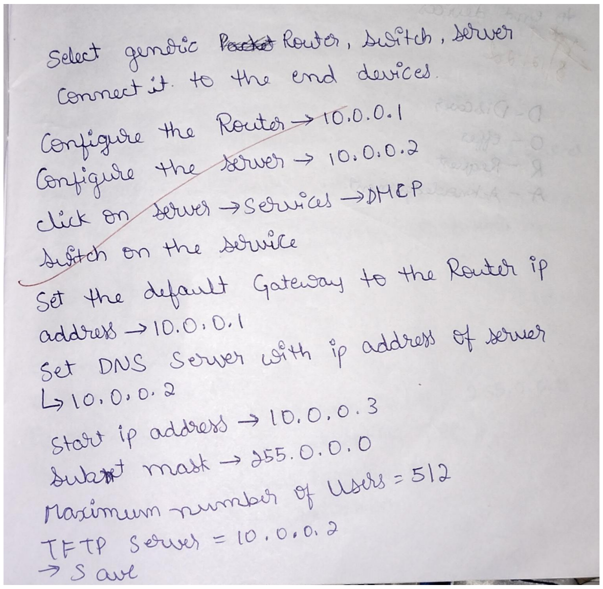
Aim of the program

Configuring DHCP within a LAN in a packet Tracer

Topology



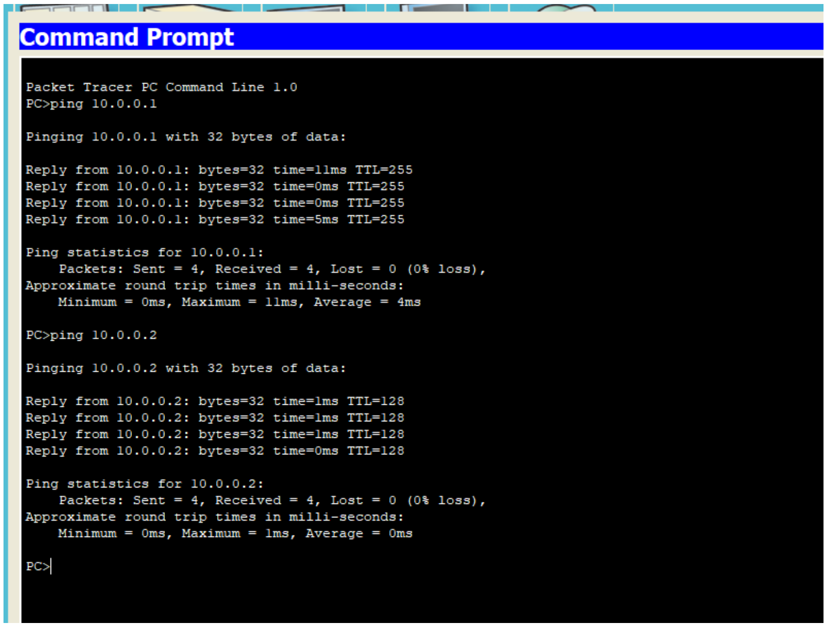
Procedure



Select generic ~~Packet~~ Router, Switch, server
Connect it. to the end devices.

Configure the Router → 10.0.0.1
Configure the server → 10.0.0.2
click on server → Services → DHCP
Switch on the service
Set the default Gateway to the Router ip address → 10.0.0.1
Set DNS Server with ip address of server
↳ 10.0.0.2
Start ip address → 10.0.0.3
Subnet mask → 255.0.0.0
Maximum number of Users = 512
TFTP Server = 10.0.0.2
→ Save

Output



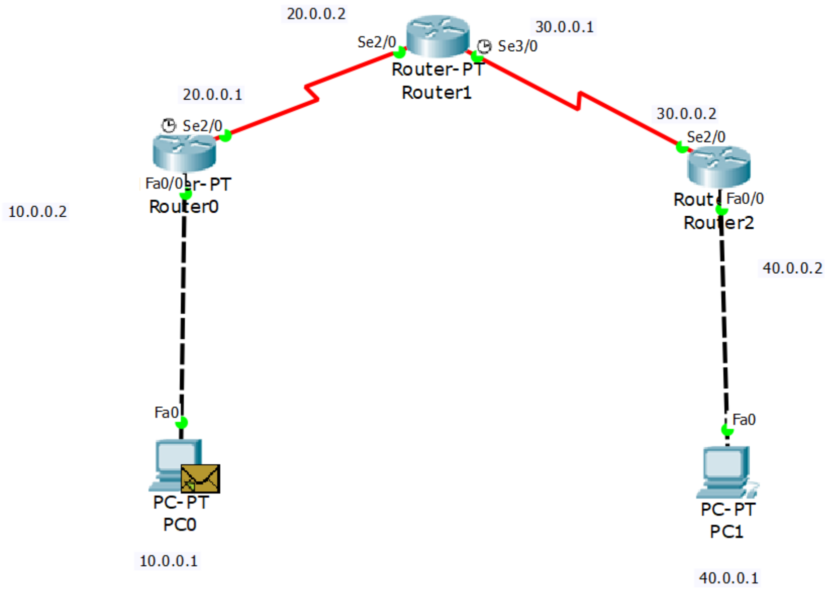
```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=11ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=5ms TTL=255

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 11ms, Average = 4ms

PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>
```

Experiment 5

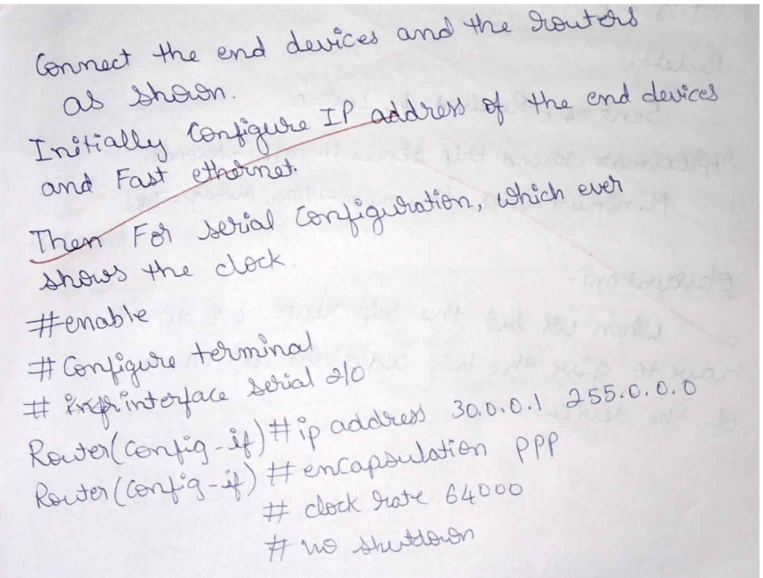
Aim of the program

Configuring RIP Routing Protocol in Routers

Topology



Procedure

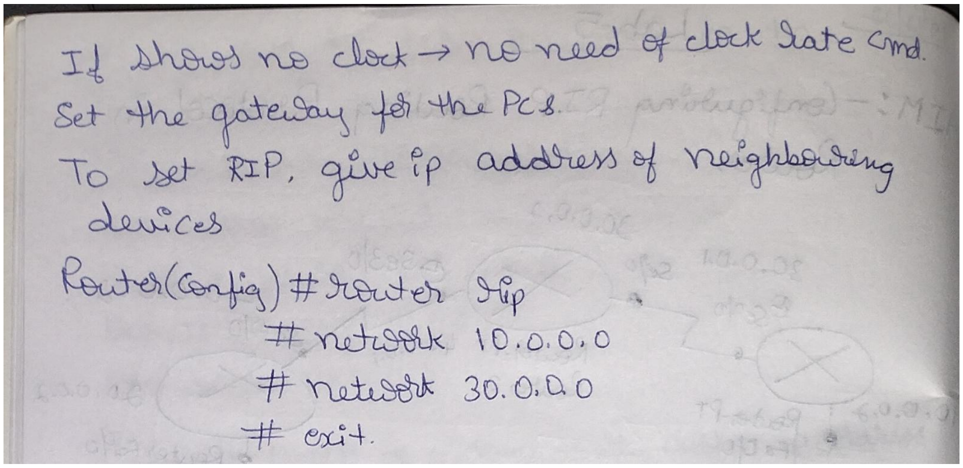


Connect the end devices and the routers as shown.

Initially configure IP address of the end devices and Fast ethernet.

Then for serial configuration, which ever shows the clock

```
#enable
# Configure terminal
# interface serial 2/0
Router(config-if)# ip address 30.0.0.1 255.0.0.0
Router(config-if) # encapsulation PPP
                  # clock rate 64000
                  # no shutdown
```



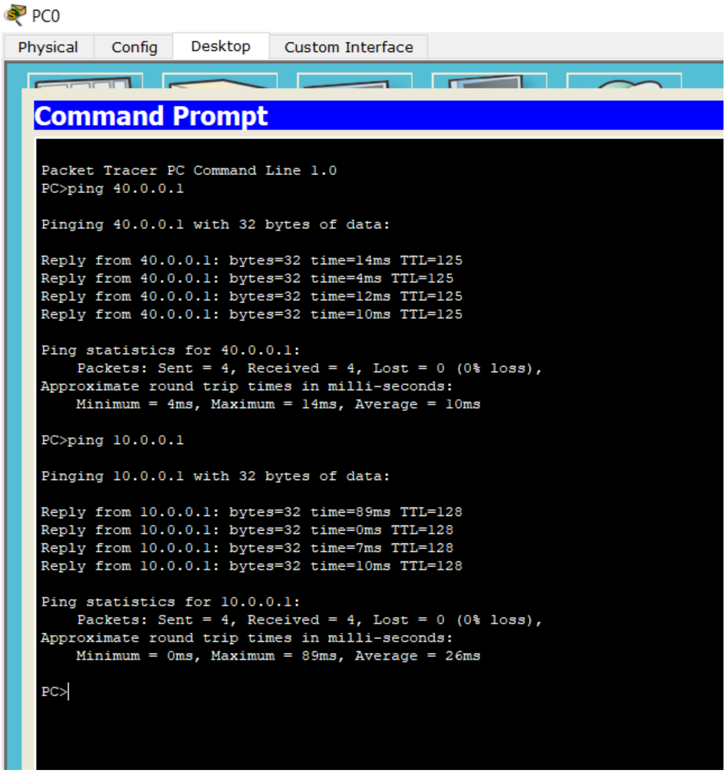
If shows no clock → no need of clock rate Cmd.

Set the gateway for the PCs.

To set RIP, give ip address of neighbouring devices

```
Router(Config)#router rip
              #network 10.0.0.0
              #network 30.0.0.0
              # exit.
```

Output



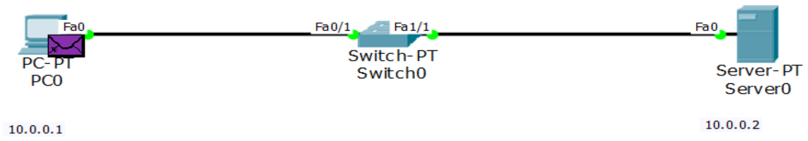
```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 time=14ms TTL=125
Reply from 40.0.0.1: bytes=32 time=4ms TTL=125
Reply from 40.0.0.1: bytes=32 time=12ms TTL=125
Reply from 40.0.0.1: bytes=32 time=10ms TTL=125

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 14ms, Average = 10ms

PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=89ms TTL=128
Reply from 10.0.0.1: bytes=32 time=0ms TTL=128
Reply from 10.0.0.1: bytes=32 time=7ms TTL=128
Reply from 10.0.0.1: bytes=32 time=10ms TTL=128

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 89ms, Average = 26ms

PC>
```

Experiment 6

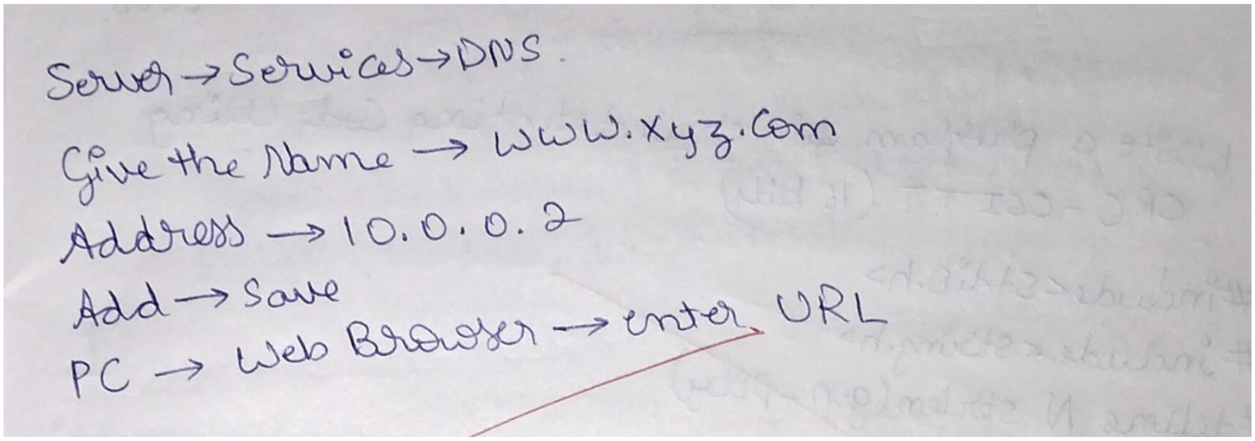
Aim of the program

Demonstration of WEB server and DNS using Packet Tracer

Topology



Procedure



Server → Services → DNS.

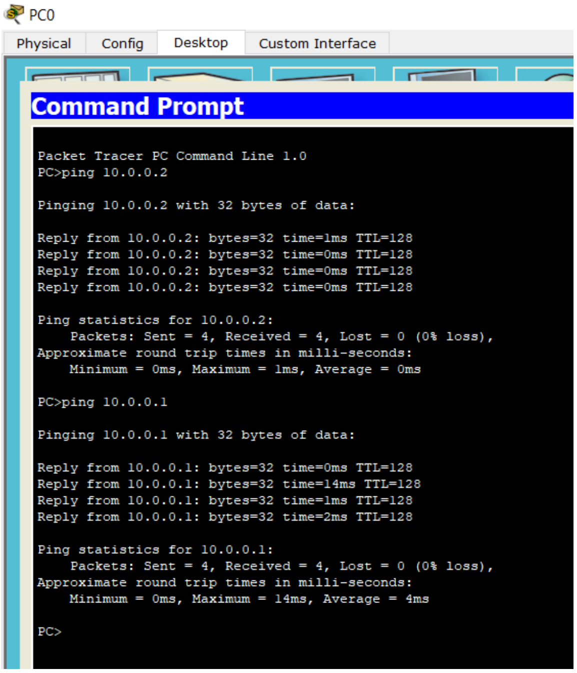
Give the Name → www.xyz.com

Address → 10.0.0.2

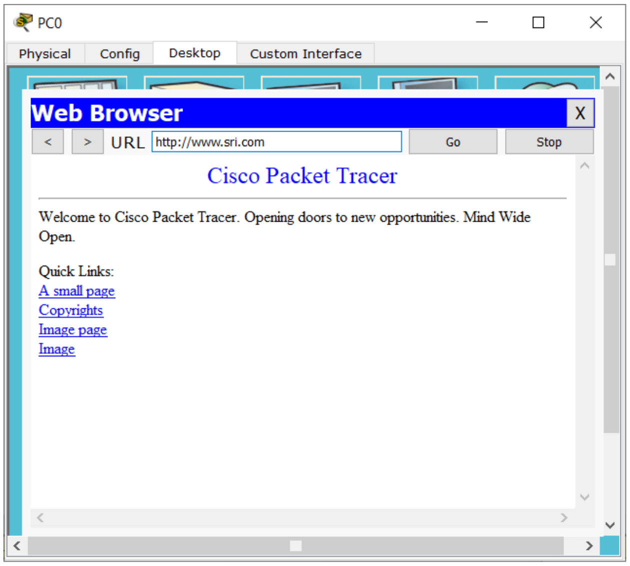
Add → Save

PC → Web Browser → enter URL

Output



```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=0ms TTL=128
Reply from 10.0.0.1: bytes=32 time=14ms TTL=128
Reply from 10.0.0.1: bytes=32 time=1ms TTL=128
Reply from 10.0.0.1: bytes=32 time=2ms TTL=128

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 14ms, Average = 4ms

PC>
```



URL http://www.sri.com

Cisco Packet Tracer

Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.

Quick Links:
A small page
Copyrights
Image page
Image

Cycle-2

Experiment 1

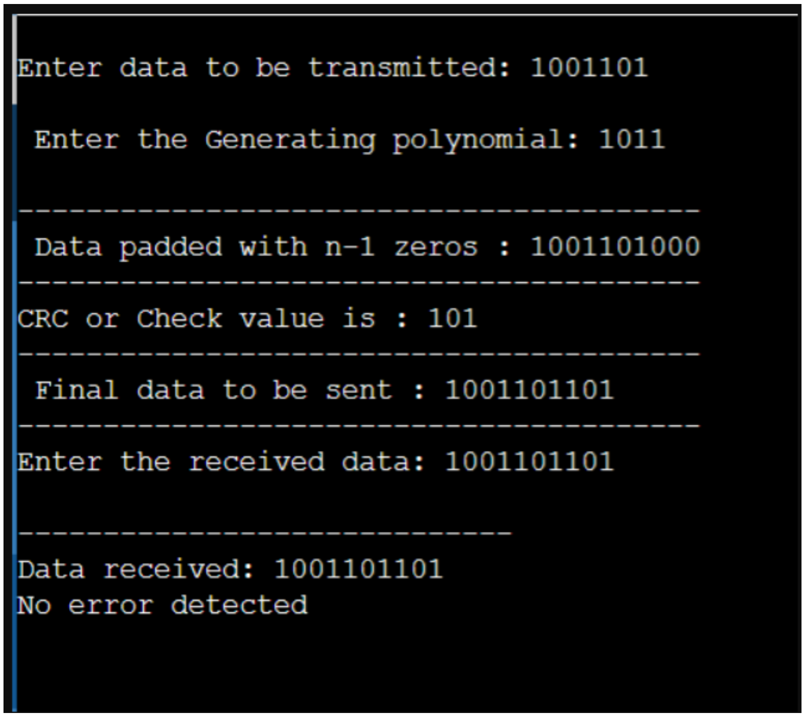
Aim of the Experiment

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

```
#include<stdio.h>
#include<string.h>
#define N strlen(gen_poly)
char data[28];
char check_value[28];
char gen_poly[10];
int data_length,i,j;
void XOR(){
   for(j = 1;j < N; j++)
   check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');
}
void receiver(){
   printf("Enter the received data: ");
   scanf("%s", data);
   printf("\n----------------------------------\n");
   printf("Data received: %s", data);
   crc();
   for(i=0;(i<N-1) && (check_value[i]!='1');i++);
      if(i<N-1)
         printf("\nError detected\n\n");
      else
         printf("\nNo error detected\n\n");}

void crc(){
    for(i=0;i<N;i++)
        check_value[i]=data[i];
    do{
        if(check_value[0]=='1')
            XOR();
        for(j=0;j<N-1;j++)
            check_value[j]=check_value[j+1];
        check_value[j]=data[i++];
    }while(i<=data_length+N-1);
}
int main()
{
    printf("\nEnter data to be transmitted: ");
    scanf("%s",data);
    printf("\n Enter the Generating polynomial: ");
    scanf("%s",gen_poly);
    data_length=strlen(data);
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n----------------------------------------");
    printf("\n Data padded with n-1 zeros : %s",data);
    printf("\n----------------------------------------");
    crc();
    printf("\nCRC or Check value is : %s",check_value);
    for(i=data_length;i<data_length+N-1;i++)
        data[i]=check_value[i-data_length];
    printf("\n----------------------------------------");
    printf("\n Final data to be sent : %s",data);
```

```
     printf("\n---------------------------------------\n");
     receiver();
        return 0;
  }
```

Output



```
Enter data to be transmitted: 1001101

 Enter the Generating polynomial: 1011

----------------------------------------
 Data padded with n-1 zeros : 1001101000
----------------------------------------
CRC or Check value is : 101
----------------------------------------
 Final data to be sent : 1001101101
----------------------------------------
Enter the received data: 1001101101

-----------------------------
Data received: 1001101101
No error detected
```

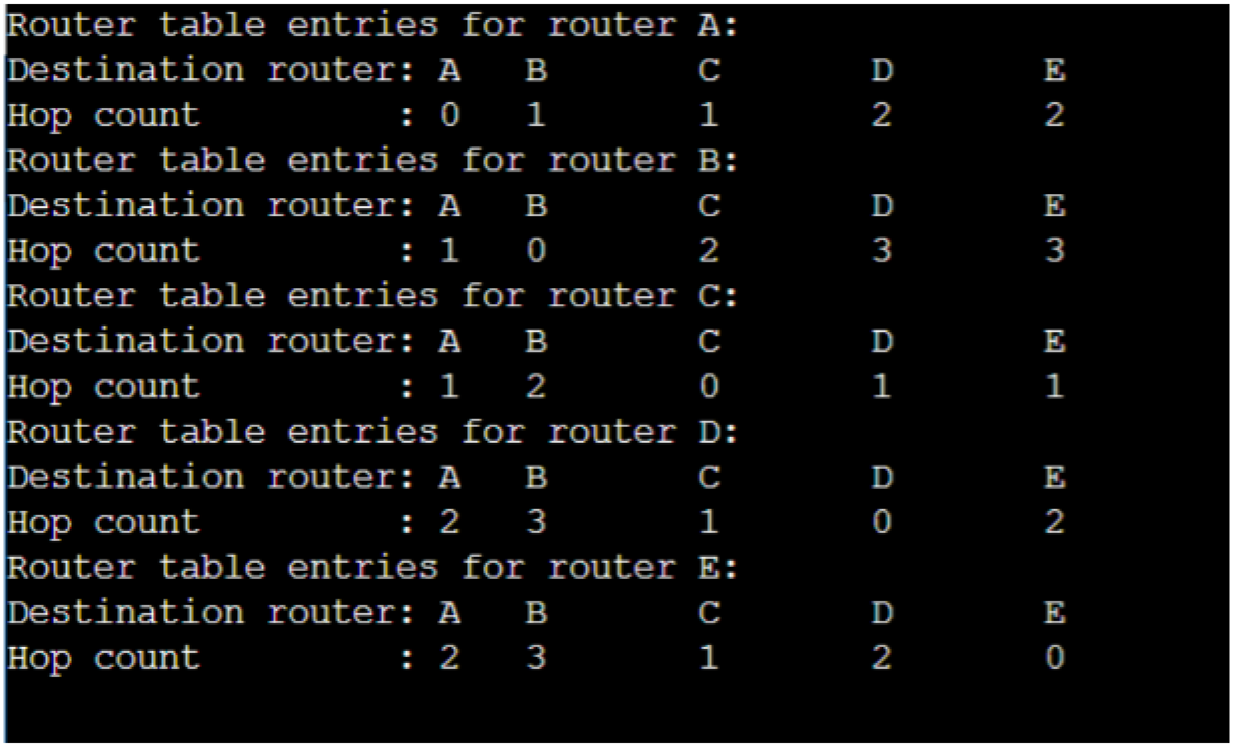
Experiment 2

Aim of the Experiment

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

```
#include<stdio.h>
#define INF 99999
#define n 5
void printSolution(int g[n])
{
printf("Hop count : ");
for(int j=0;j<n;j++)
{
if(g[j] == INF)
printf("INF\t");
else
printf("%d\t",g[j]);
}
printf("\n");
}
void findShortestPath(int dist[][n])
{
for(int k=0;k<n;k++)
{
for(int i=0;i<n;i++)
{
for(int j=0;j<n;j++)
{
if(dist[i][j] > dist[i][k] + dist[k][j]
&&(dist[i][k] != INF && dist[k][j] != INF))
{
dist[i][j] = dist[i][k] + dist[k][j];
}
}
}
}
char c = 'A';
for(int i=0; i<n; i++ )
{
printf("Router table entries for router %c:\n", c);
printf("Destination router: A\tB\tC\tD\tE\n");
printSolution(dist[i]);
c++;
}
}
int main()
{
int graph[][n] = { {0, 1, 1, INF, INF},
{1, 0, INF, INF, INF},
{1, INF, 0, 1, 1},
{INF, INF, 1, 0, INF},
{INF, INF, 1, INF, 0}};

findShortestPath(graph);
return 0;
}
```

Output:



```
Router table entries for router A:
Destination router: A   B       C       D       E
Hop count         : 0   1       1       2       2
Router table entries for router B:
Destination router: A   B       C       D       E
Hop count         : 1   0       2       3       3
Router table entries for router C:
Destination router: A   B       C       D       E
Hop count         : 1   2       0       1       1
Router table entries for router D:
Destination router: A   B       C       D       E
Hop count         : 2   3       1       0       2
Router table entries for router E:
Destination router: A   B       C       D       E
Hop count         : 2   3       1       2       0
```

}

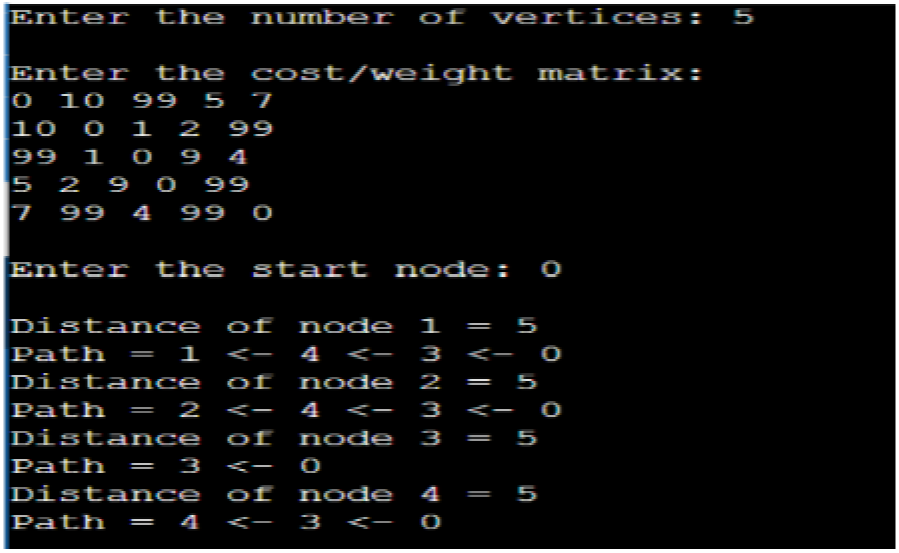
Experiment 3

Aim of the Experiment: Implement Dijkstra’s algorithm to compute the shortest path for a given topology.

```
#include <stdio.h>

#include <stdlib.h>

void dijkstra(int graph[10][10],int V)

{

int distance[V], predefine[V], visited[V];

int startnode, count, min_distance, nextnode, i, j;

printf("\nEnter the start node: ");

scanf("%d", &startnode);

for(i=0; i<V; i++) {

distance[i] = graph[startnode][i];

predefine[i] = startnode;

visited[i] = 0;

}

distance[startnode] = 0;

visited[startnode] = 1;

count = 1;

while(count<V-1) {

min_distance = 99;

for(i=0; i<V; i++) {

if(distance[i] < min_distance && visited[i]==0)

{

min_distance = distance[i];

nextnode = i;

}
}
visited[nextnode] = 1;
for(i=0;i<V;i++)
{
if(visited[i] == 0)
{
if((min_distance + graph[nextnode][i]) < distance[i])
{
distance[i] = min_distance + graph[nextnode][i];
predefine[i] = nextnode;
}}
}
count = count + 1;
}
for(i=0;i<V;i++) {
if(i!=startnode) {
printf("\nDistance of node %d = %d", i, distance[i]);
printf("\nPath = %d",i);
j = i;
do
{
j = predefine[j];
printf(" <- %d",j);
} while (j != startnode);
}
}
}
int main()
```

```
{
int i, j;
int V;
printf("Enter the number of vertices: ");
scanf("%d", &V);
int graph[V][V];
printf("\nEnter the cost/weight matrix: \n");
for(i=0; i<V; i++) {
for(j=0;j<V;j++) {
scanf("%d", &graph[i][j]);}
dijkstra(graph, V);
return 0;
}
```

Output:



```
Enter the number of vertices: 5

Enter the cost/weight matrix:
0 10 99 5 7
10 0 1 2 99
99 1 0 9 4
5 2 9 0 99
7 99 4 99 0

Enter the start node: 0

Distance of node 1 = 5
Path = 1 <- 4 <- 3 <- 0
Distance of node 2 = 5
Path = 2 <- 4 <- 3 <- 0
Distance of node 3 = 5
Path = 3 <- 0
Distance of node 4 = 5
Path = 4 <- 3 <- 0
```

Experiment 4

Aim of the Experiment: Using TCP/IP 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.

Server:

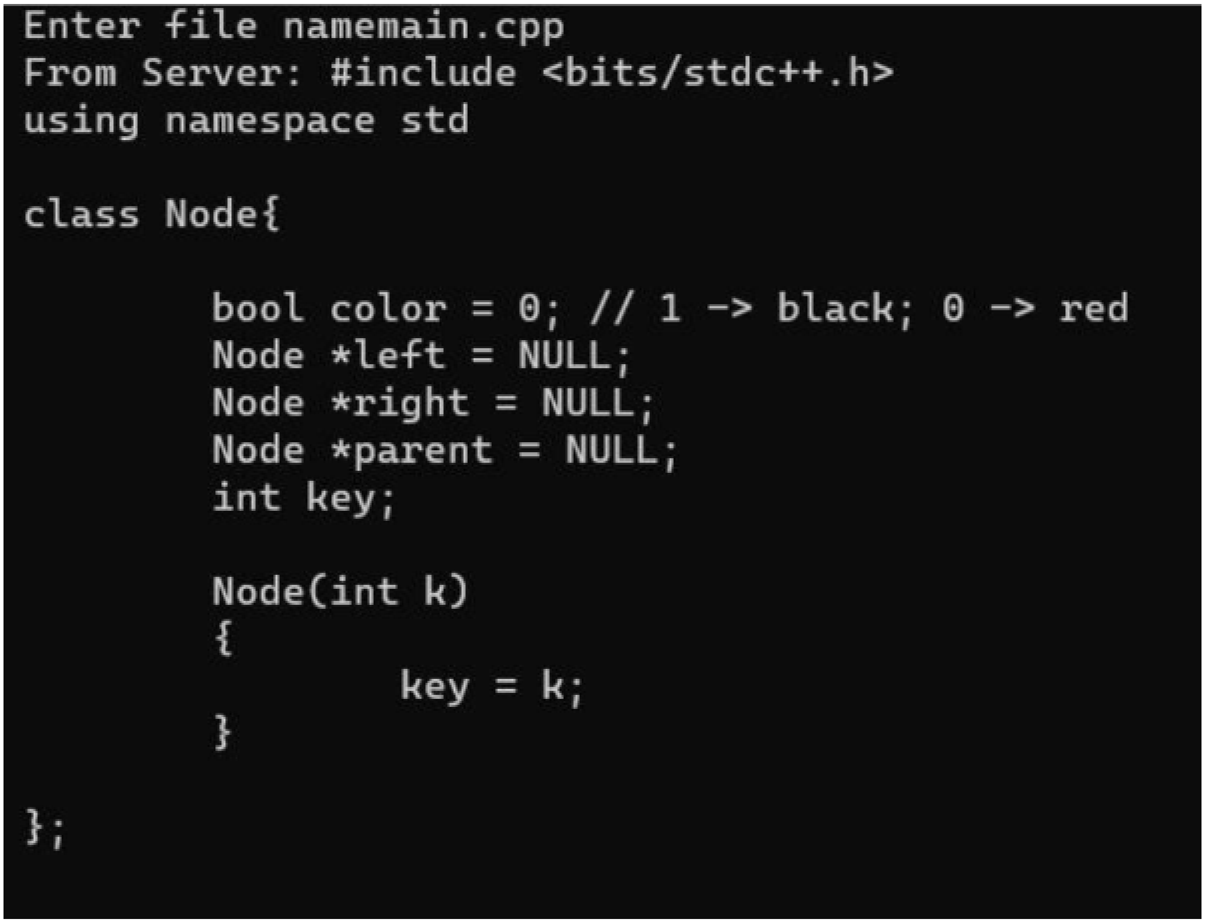
```
from socket import *
serverName = ''
serverPort = 12530
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
try:
file = open(sentence,"r")
l = file.read(1024)
connectionSocket.send(l.encode())
file.close()
except Exception as e:
message = "No such file exist"
connectionSocket.send(message.encode())
connectionSocket.close()
```

Client:

```
from socket import *
serverName = '192.168.1.104'
serverPort = 12530
clientSocket = socket(AF_INET, SOCK_STREAM)
```

```
clientSocket.connect((serverName,serverPort))
sentence = input("Enter file name")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ('From Server:', filecontents)
clientSocket.close()
```

Output



```
Enter file namemain.cpp
From Server: #include <bits/stdc++.h>
using namespace std

class Node{

        bool color = 0; // 1 -> black; 0 -> red
        Node *left = NULL;
        Node *right = NULL;
        Node *parent = NULL;
        int key;

        Node(int k)
        {
                key = k;
        }

};
```

Experiment 5

Aim of the Experiment

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

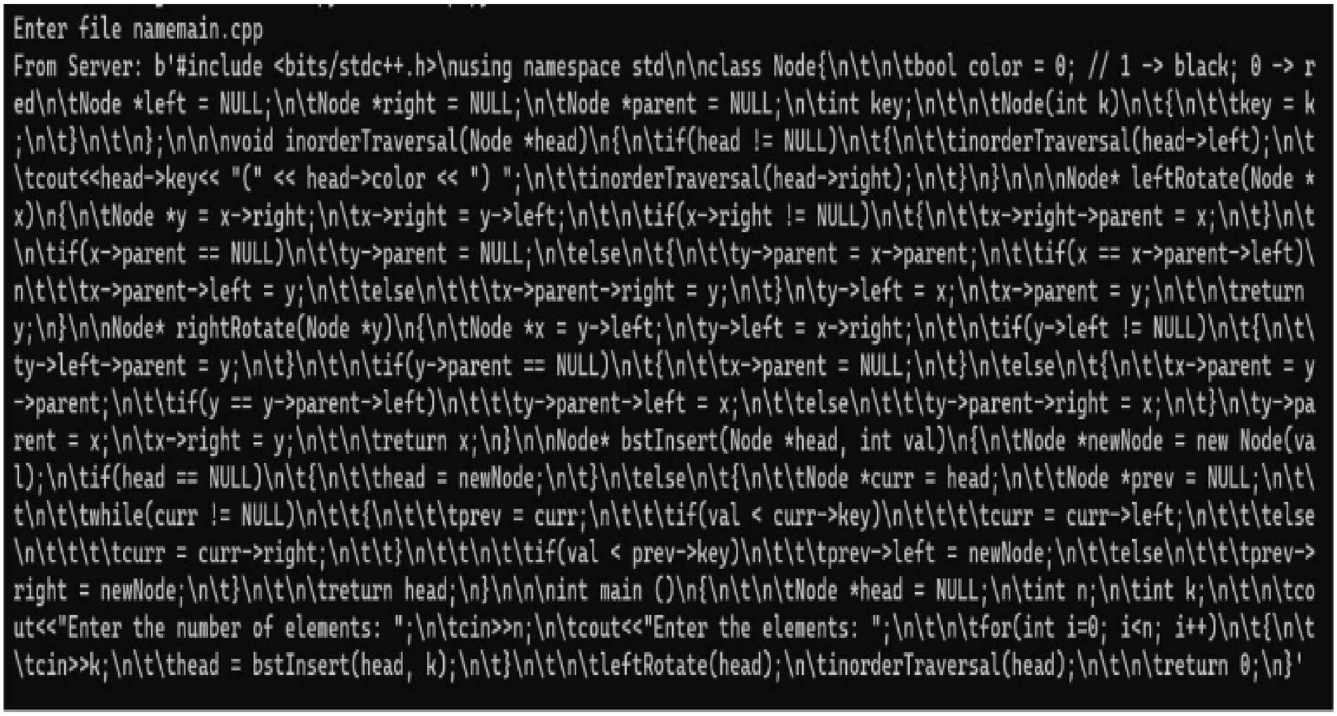
Server:

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
sentence,clientAddress = serverSocket.recvfrom(2048)
file=open(sentence,"r")
l=file.read(2048)
serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
print("sent back to client",l)
file.close()
```

Client:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('From Server:', filecontents)
clientSocket.close()
```

Output



```
Enter file namemain.cpp
From Server: b'#include <bits/stdc++.h>\nusing namespace std\n\nclass Node{\n\t\n\tbool color = 0; // 1 -> black; 0 -> r
ed\n\tNode *left = NULL;\n\tNode *right = NULL;\n\tNode *parent = NULL;\n\tint key;\n\t\n\tNode(int k)\n\t{\n\t\tkey = k
;\n\t}\n\t\n};\n\n\nvoid inorderTraversal(Node *head)\n{\n\tif(head != NULL)\n\t{\n\t\tinorderTraversal(head->left);\n\t
\tcout<<head->key<< "(" << head->color << ") ";\n\t\tinorderTraversal(head->right);\n\t}\n}\n\n\nNode* leftRotate(Node *
x)\n{\n\tNode *y = x->right;\n\tx->right = y->left;\n\t\n\tif(x->right != NULL)\n\t{\n\t\tx->right->parent = x;\n\t}\n\t
\n\tif(x->parent == NULL)\n\t\ty->parent = NULL;\n\telse\n\t{\n\t\ty->parent = x->parent;\n\t\tif(x == x->parent->left)\
n\t\t\tx->parent->left = y;\n\t\telse\n\t\t\tx->parent->right = y;\n\t}\n\ty->left = x;\n\tx->parent = y;\n\t\n\treturn
y;\n}\n\nNode* rightRotate(Node *y)\n{\n\tNode *x = y->left;\n\ty->left = x->right;\n\t\n\tif(y->left != NULL)\n\t{\n\t\
ty->left->parent = y;\n\t}\n\t\n\tif(y->parent == NULL)\n\t{\n\t\tx->parent = NULL;\n\t}\n\telse\n\t{\n\t\tx->parent = y
->parent;\n\t\tif(y == y->parent->left)\n\t\t\ty->parent->left = x;\n\t\telse\n\t\t\ty->parent->right = x;\n\t}\n\ty->pa
rent = x;\n\tx->right = y;\n\t\n\treturn x;\n}\n\nNode* bstInsert(Node *head, int val)\n{\n\tNode *newNode = new Node(va
l);\n\tif(head == NULL)\n\t{\n\t\thead = newNode;\n\t}\n\telse\n\t{\n\t\tNode *curr = head;\n\t\tNode *prev = NULL;\n\t\
t\n\t\twhile(curr != NULL)\n\t\t{\n\t\t\tprev = curr;\n\t\t\tif(val < curr->key)\n\t\t\t\tcurr = curr->left;\n\t\t\telse
\n\t\t\t\tcurr = curr->right;\n\t\t}\n\t\t\n\t\tif(val < prev->key)\n\t\t\tprev->left = newNode;\n\t\telse\n\t\t\tprev->
right = newNode;\n\t}\n\t\n\treturn head;\n}\n\n\nint main ()\n{\n\t\n\tNode *head = NULL;\n\tint n;\n\tint k;\n\t\n\tco
ut<<"Enter the number of elements: ";\n\tcin>>n;\n\tcout<<"Enter the elements: ";\n\t\n\tfor(int i=0; i<n; i++)\n\t{\n\t
\tcin>>k;\n\t\thead = bstInsert(head, k);\n\t}\n\t\n\tleftRotate(head);\n\tinorderTraversal(head);\n\t\n\treturn 0;\n}'
```