VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

Computer Networks Lab

Submitted by

AKANSHA MEHROTRA (1BM20CS005)

Under the Guidance of
Dr. Shyamala G
Assistant Professor, BMSCE

in partial fulfillment 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 "LAB COURSE **COMPUTER NETWORKS**" carried out by **AKANSHA MEHROTRA(1BM20CS005),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment 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. Shyamala G
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

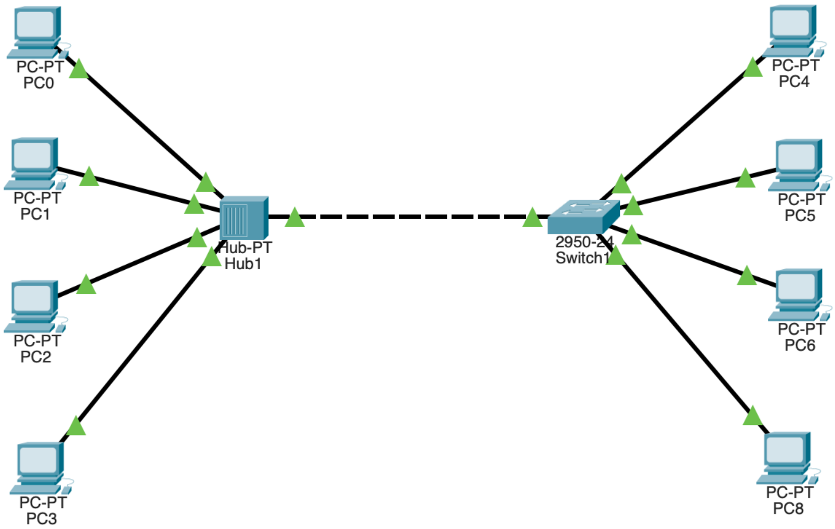
Index

Cycle-1

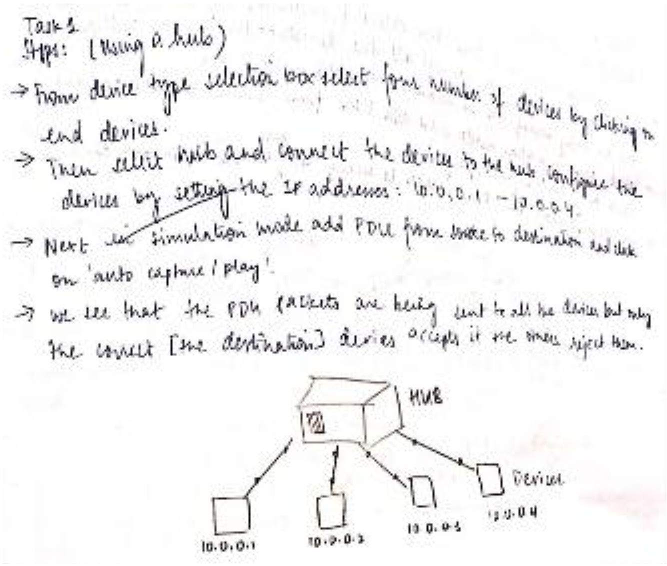
Experiment 1

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-


Task 1
Steps: (using a hub)

→ From device type selection box select four number of devices by clicking on end devices.

→ Then select hub and connect the devices to the hub, configure the devices by setting the IP addresses: 10.0.0.1 - 10.0.0.4

→ Next in simulation mode add PDU from source to destination and click on 'auto capture / play'.

→ we see that the PDU packets are being sent to all the devices but only the correct [the destination] devices accepts it the others reject them.

Event list.

Time (sec)	Last device	At device
0.000	----	PC0
0.001	PC0	Hub0
0.002	Hub0	PC1
0.002	Hub0	PC2
0.002	Hub0	PC3
0.003.	PC3	Hub0

Real time (event list)

Fire	Last status	Source	Destination	Time	Periodic
	Successful	PC0	PC3	0.000	N

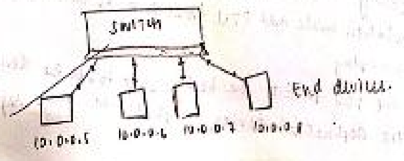
Command Prompt (Ping)

```
C:\> ping 10.0.0.4
  pinging 10.0.0.4 with 32 bytes of data
   reply from 10.0.0.4 : bytes:32 time=0ms TTL=128
   Statistics
      packets: sent=4, received=4, lost=0
   Round trip times
         minimum=0ms, Max=0ms, Avg=0ms
```

Task 2

Using a switch

→ From device type selection box select from devices by clicking on end device
→ Then select switch and connect all the devices to the switch, configure the devices by setting the IP address : 10.0.0.5 - 10.0.0.8
→ Next in simulation mode add PDV from source to destination and click on 'auto capture / play'
→ we observe that the PDV packet is being sent only to the destination device.



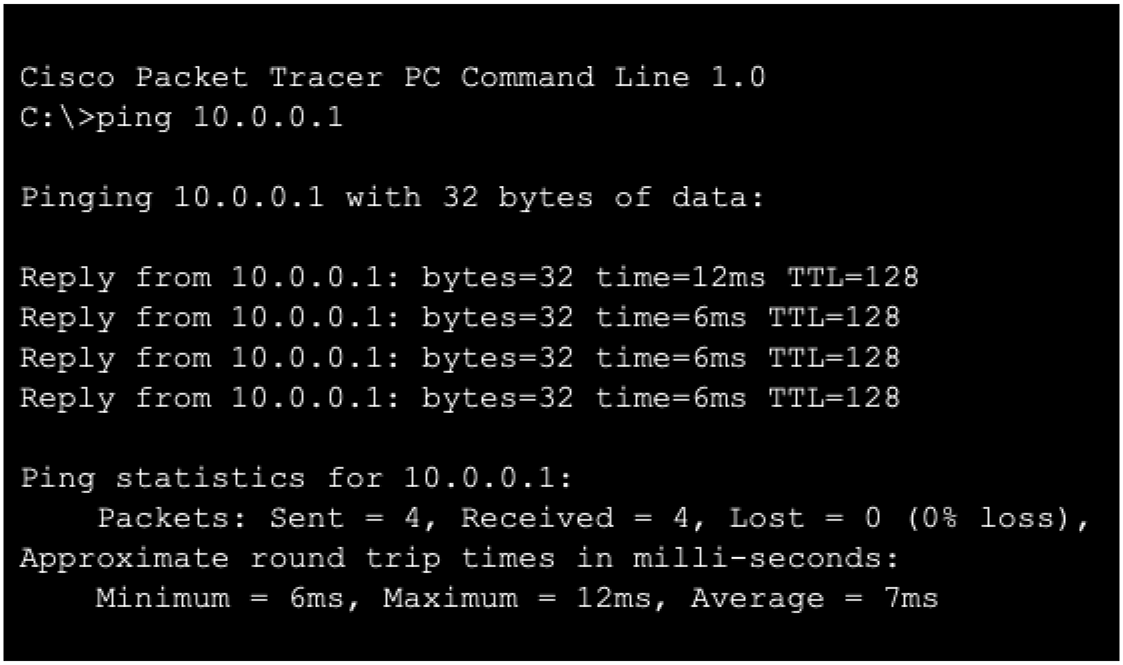
Simulation mode

Time	Last Device	At device
0.000	- - -	PC0
0.001	PC0	Hub0
0.002	Hub 0	Switch0
0.003	Switch 0	PC7
0.004	PC7	Switch0
0.005	Switch0	Hub0
0.006	Hub 0	PC4
0.006	Hub0	PC2
0.006	Hub0	PC3
0.006	Hub0	PC1

Ping

```
C:\> ping 10.0.0.8
       pinging 10.0.0.8 with 32 bytes of data

reply from 10.0.0.8 : bytes=32 time=1ms
reply from 10.0.0.8 : bytes=32 time=1ms
reply from 10.0.0.8 : bytes=32 time=1ms
reply from 10.0.0.8 : bytes=32 time=1ms

Statistics for 10.0.0.8
      packets : sent=4 , received=4 , lost=0
   Approx round trip
        minimum = 0ms        max = 1ms     Avg: 0ms.
```

Snapshot of Output-

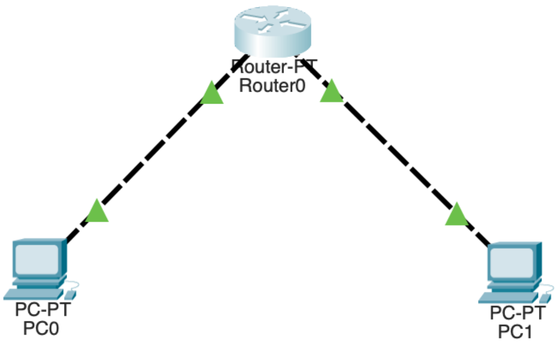


```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=12ms TTL=128
Reply from 10.0.0.1: bytes=32 time=6ms TTL=128
Reply from 10.0.0.1: bytes=32 time=6ms TTL=128
Reply from 10.0.0.1: bytes=32 time=6ms 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 = 6ms, Maximum = 12ms, Average = 7ms
```

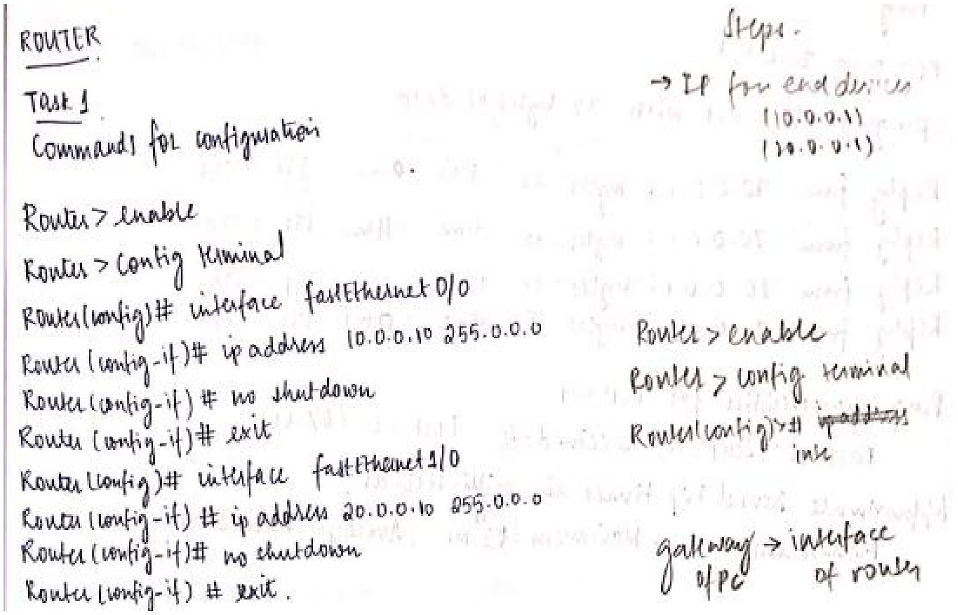
Experiment 2

Aim of the program- Configuring IP address to Routers in Packet Tracer. Explore the following messages: Ping Responses, Destination unreachable, Request timed out, Reply

Topology-



Procedure-



ROUTER

Task 1

Commands for configuration

```
Router> enable
Router > config terminal
Router(config)# interface fastEthernet 0/0
Router (config-if)# ip address 10.0.0.10 255.0.0.0
Router(config-if) # no shutdown
Router (config-if)# exit
Router (config)# interface fastEthernet 1/0
Router (config-if) # ip address 20.0.0.10 255.0.0.0
Router (config-if)# no shutdown
Router (config-if) # exit.
```

Steps.

→ IP for end devices
(10.0.0.1)
(20.0.0.1)

```
Router > enable
Router > config terminal
Router(config)># ~~ipaddress~~
                 int
```

gateway of PC → interface of router

Steps:

→ From routers select the generic router-PT and from end devices select two devices.

→ using 'automatically-choose connection type' from connection create connection between the devices and the router.

→ configure the ip address for both end devices.

→ To configure the router, select CLI from the device prompts and enter the commands mentioned above.

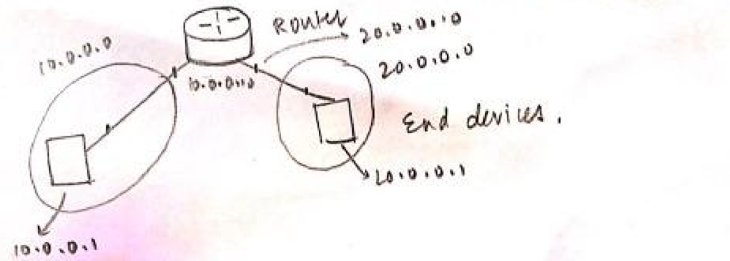
→ Once the configuration is done, the connection between the end device and router should turn green.

→ Now, for each device, set the gateway as the interface of the router.

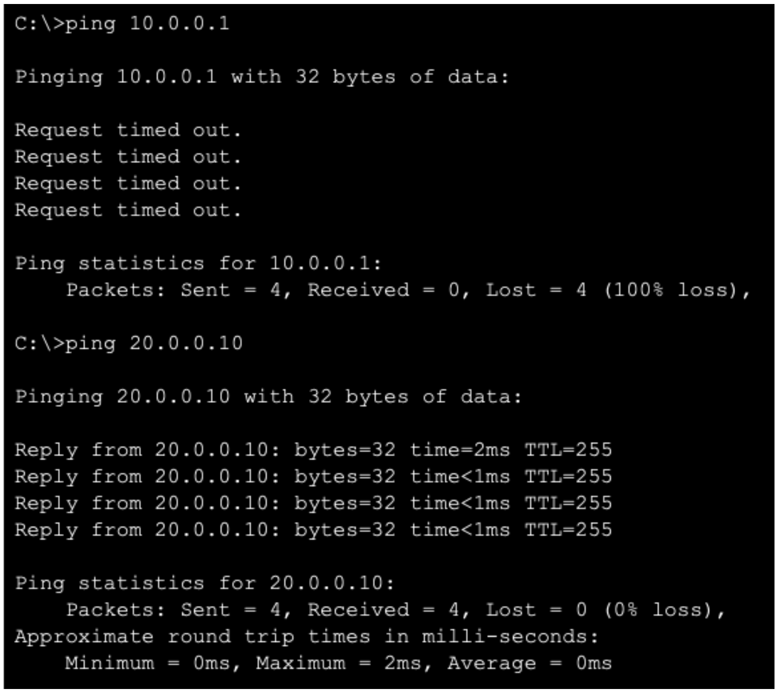
→ Add PDU from one end device (source) to another (destination).

→ we observe that the packet gets delivered successfully when checked on simulation using 'auto capture /play'.

Fire	Last Status	Source	Destination	Type	Time(sec)	Periodic	Num
	successful	PC0	PC1	ICMP	0.000	N	0.



Snapshot of Output-

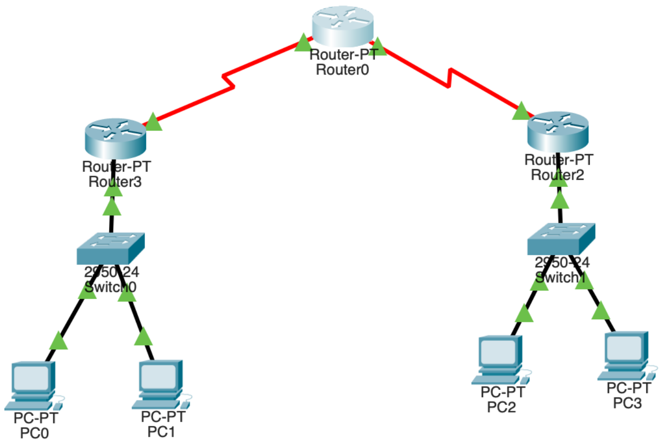


```
C:\>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

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

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ping 20.0.0.10

Pinging 20.0.0.10 with 32 bytes of data:

Reply from 20.0.0.10: bytes=32 time=2ms TTL=255
Reply from 20.0.0.10: bytes=32 time<1ms TTL=255
Reply from 20.0.0.10: bytes=32 time<1ms TTL=255
Reply from 20.0.0.10: bytes=32 time<1ms TTL=255

Ping statistics for 20.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms
```

Experiment 3

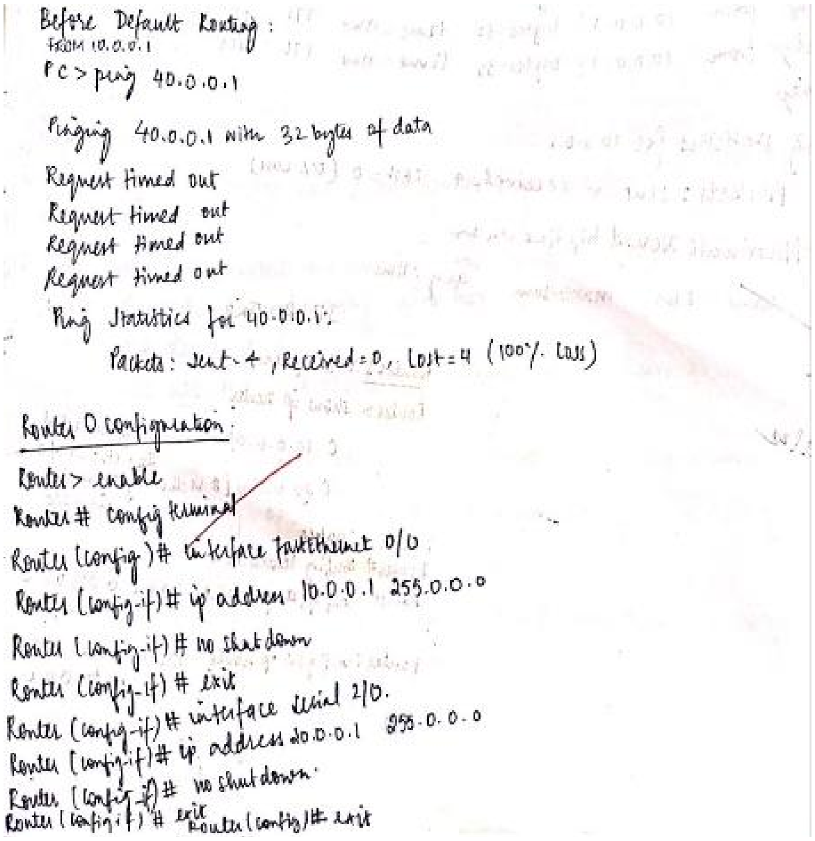
Aim of the program- Configuring default route to the Router

Topology-

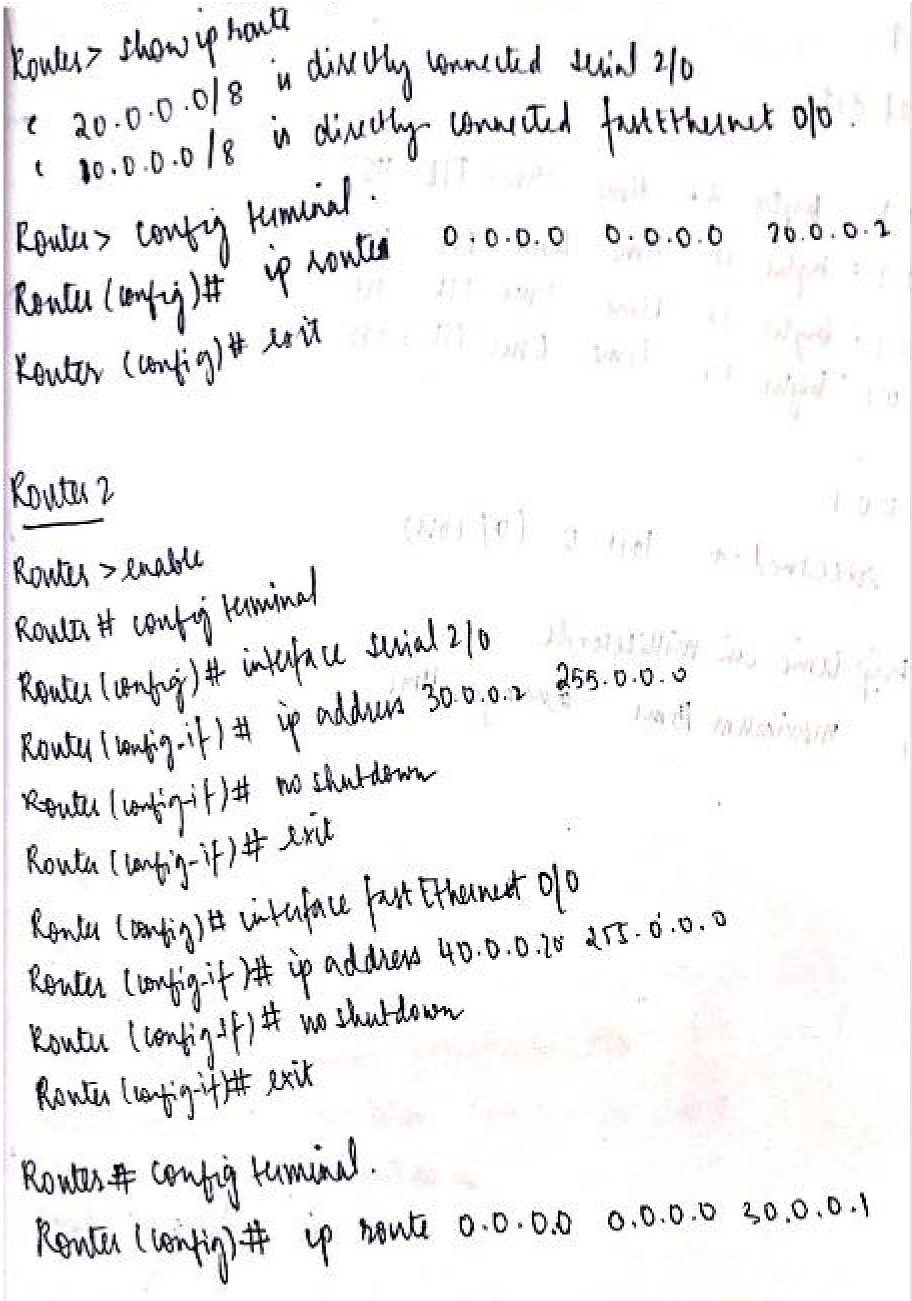


here Router3=Router0, Router0=Router1, Router2=Router2

Procedure-



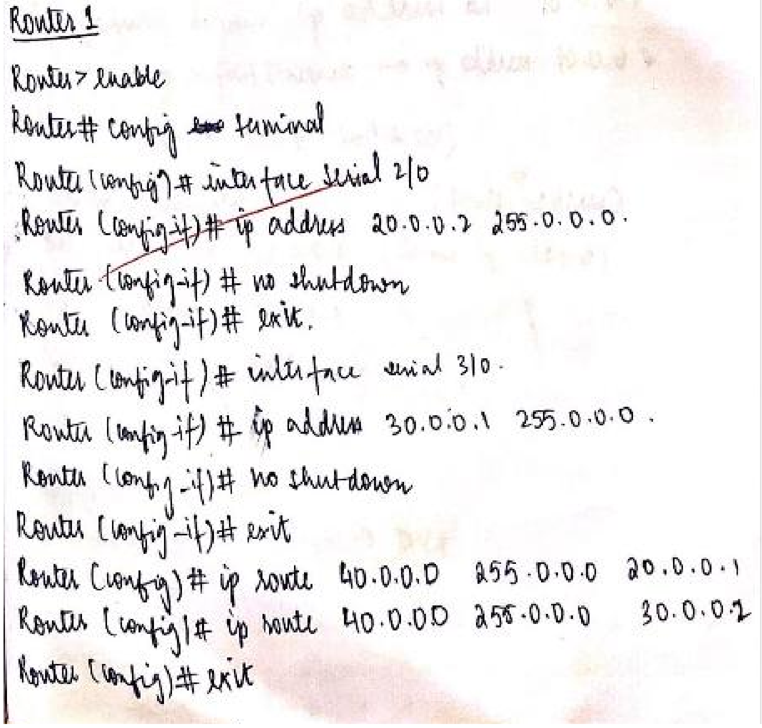
```
Before Default Routing :
FROM 10.0.0.1
PC > ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data
Request timed out
Request timed out
Request timed out
Request timed out
Ping Statistics for 40.0.0.1:
     Packets : sent-4 , Received = 0, Lost = 4 (100% Loss)

Router 0 configuration:
Router > enable
Router # config terminal
Router (config )# interface fastethernet 0/0
Router (config-if)# ip address 10.0.0.1 255.0.0.0
Router (config-if) # no shutdown
Router (config-if) # exit
Router (config-if)# interface serial 2/0.
Router (config-if)# ip address 20.0.0.1   255.0.0.0
Router (config-if)# no shutdown
Router (config-if) # exit
                 Router (config)# exit
```



```
Router> show ip route
 c  20.0.0.0/8  is directly connected serial 2/0
 c  10.0.0.0/8  is directly connected fastEthernet 0/0.

Router> config terminal.
Router(config)#  ip router  0.0.0.0  0.0.0.0  20.0.0.2
Router (config)# exit
```

Router 2

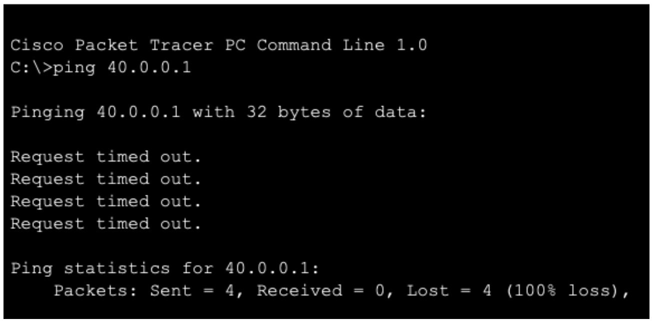
```
Router > enable
Router # config terminal
Router (config)# interface serial 2/0
Router (config-if)# ip address 30.0.0.2 255.0.0.0
Router (config-if)# no shutdown
Router (config-if)# exit
Router (config)# interface fast Ethernet 0/0
Router (config-if)# ip address 40.0.0.20 255.0.0.0
Router (config-if)# no shutdown
Router (config-if)# exit

Router# config terminal.
Router (config)#  ip route 0.0.0.0  0.0.0.0  30.0.0.1
```



```
Router 1
Router> enable
Router# config terminal
Router (config)# interface serial 2/0
Router (config-if)# ip address 20.0.0.2 255.0.0.0.
Router (config-if) # no shutdown
Router (config-if)# exit.
Router (config-if) # interface serial 3/0.
Router (config-if) # ip address 30.0.0.1 255.0.0.0 .
Router (config-if)# no shutdown
Router (config-if)# exit
Router (config)# ip route 40.0.0.0 255.0.0.0 20.0.0.1
Router (config)# ip route 40.0.0.0 255.0.0.0 30.0.0.2
Router (config)# exit
```

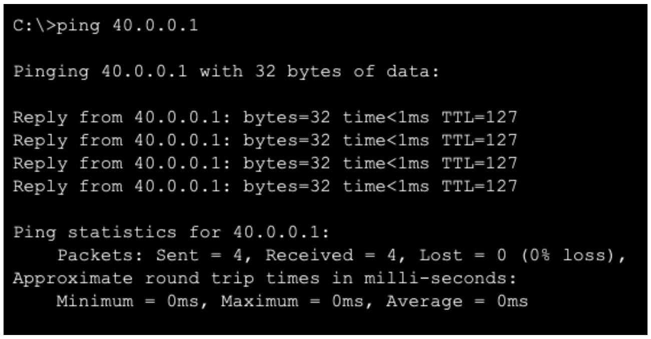
Snapshot of Output-

Before default routing-



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

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

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

After default routing-

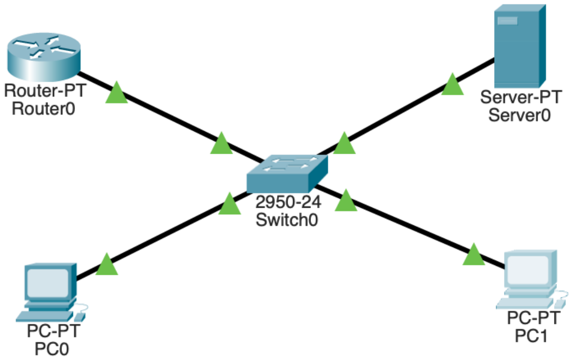


```
C:\>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 time<1ms TTL=127
Reply from 40.0.0.1: bytes=32 time<1ms TTL=127
Reply from 40.0.0.1: bytes=32 time<1ms TTL=127
Reply from 40.0.0.1: bytes=32 time<1ms TTL=127

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

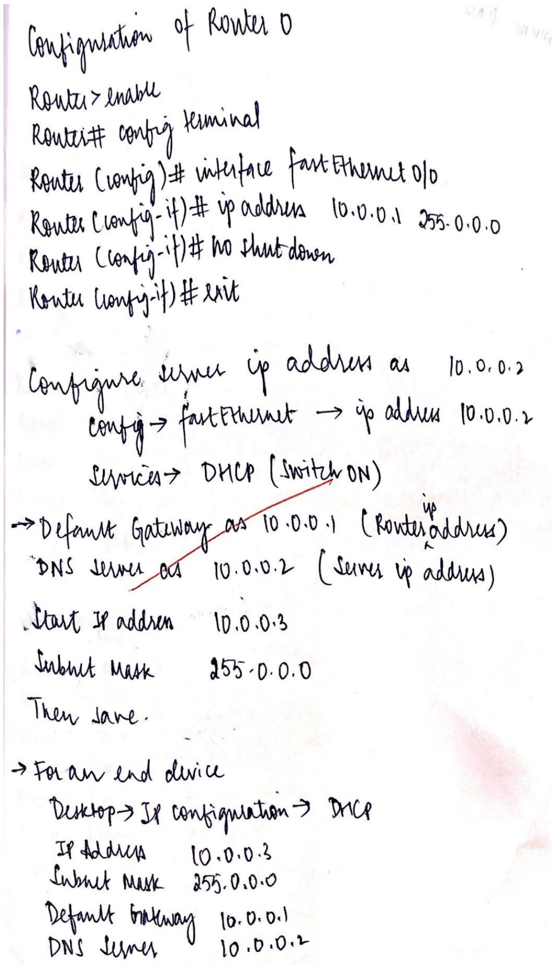
Experiment 4

Aim of the program- Configuring DHCP within a LAN in a packet Tracer

Topology-



Procedure-



Configuration of Router 0

```
Router>enable
Router# config terminal
Router (config)# interface fastEthernet 0/0
Router (config-if)# ip address 10.0.0.1 255.0.0.0
Router (config-if)# no shutdown
Router (config-if)# exit
```

Configure server ip address as 10.0.0.2

config → fastEthernet → ip address 10.0.0.2

Services → DHCP (Switch ON)

→ Default Gateway as 10.0.0.1 (Router ip address)

DNS server as 10.0.0.2 (Server ip address)

Start IP address 10.0.0.3

Subnet Mask 255.0.0.0

Then save.

→ For an end device

Desktop → IP configuration → DHCP

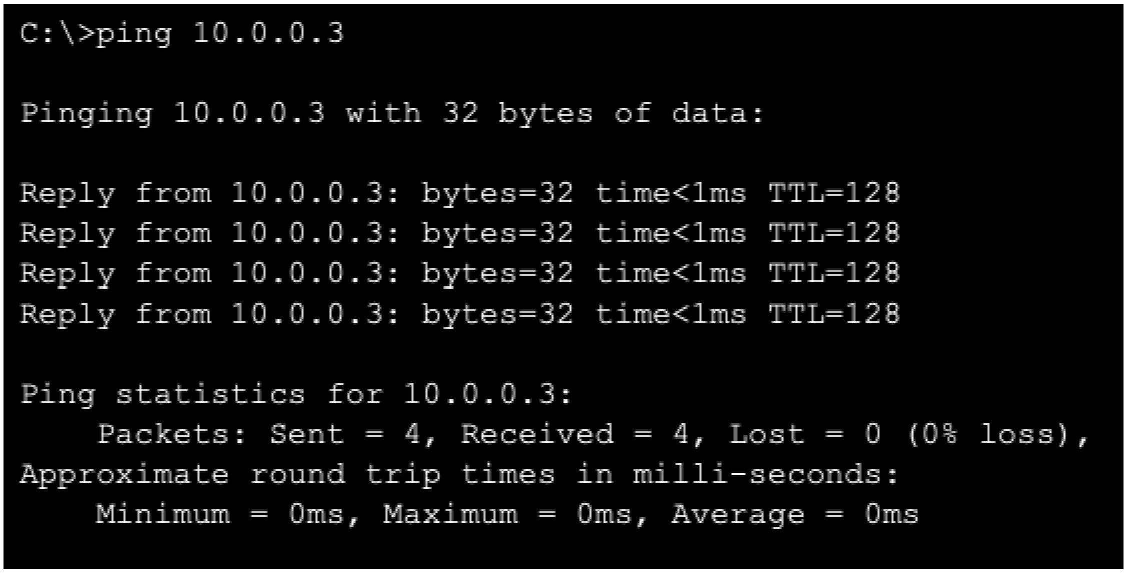
IP Address 10.0.0.3

Subnet Mask 255.0.0.0

Default Gateway 10.0.0.1

DNS Server 10.0.0.2

Snapshot of Output-

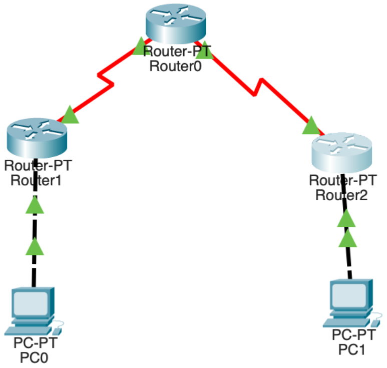


```
C:\>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time<1ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

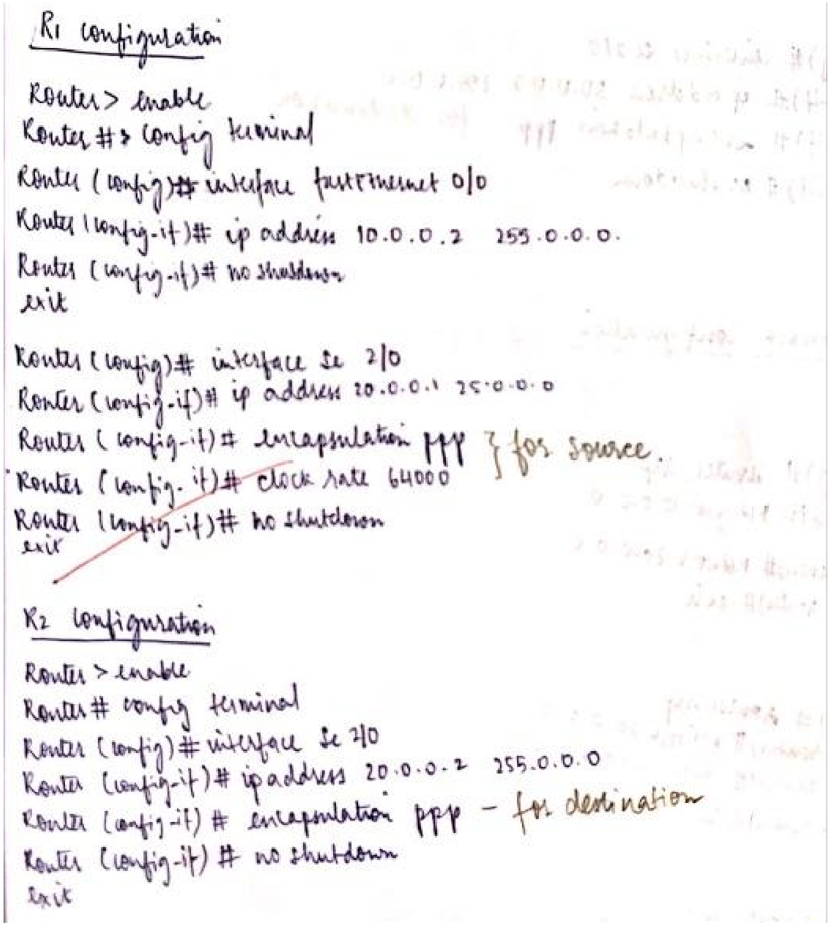
Experiment 5

Aim of the program- Configuring RIP Routing Protocol in Routers

Topology-



Procedure-

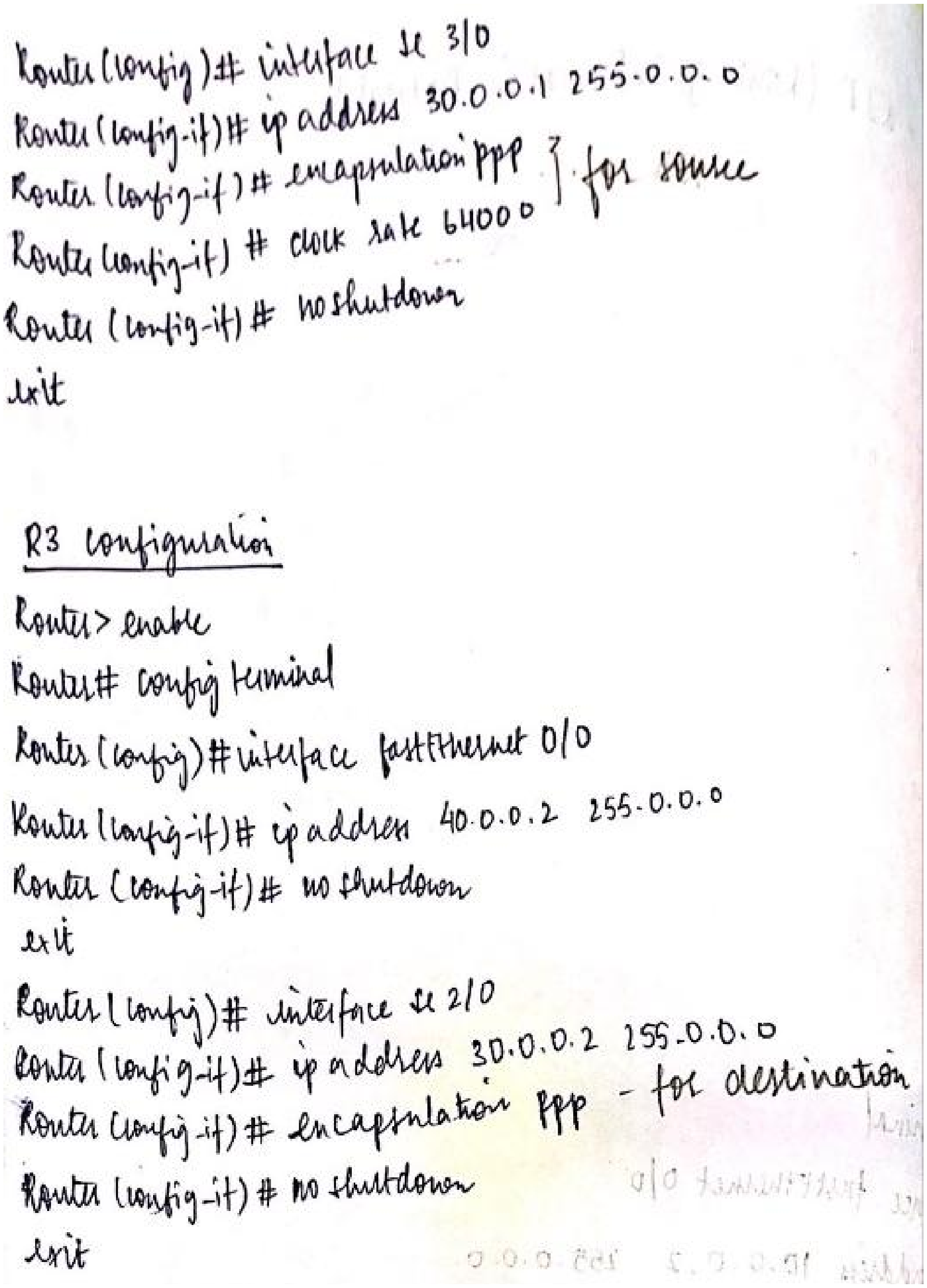


R1 configuration

```
Router> enable
Router #> config terminal
Router (config)# interface fastethernet 0/0
Router (config-it)# ip address 10.0.0.2   255.0.0.0.
Router (config-it)# no shutdown
exit

Router (config)# interface se 2/0
Router (config-if)# ip address 20.0.0.1 25.0.0.0
Router ( config-it)# encapsulation ppp } for source.
Router (config- it)# clock rate 64000
Router (config-it)# no shutdown
exit
```

R2 configuration

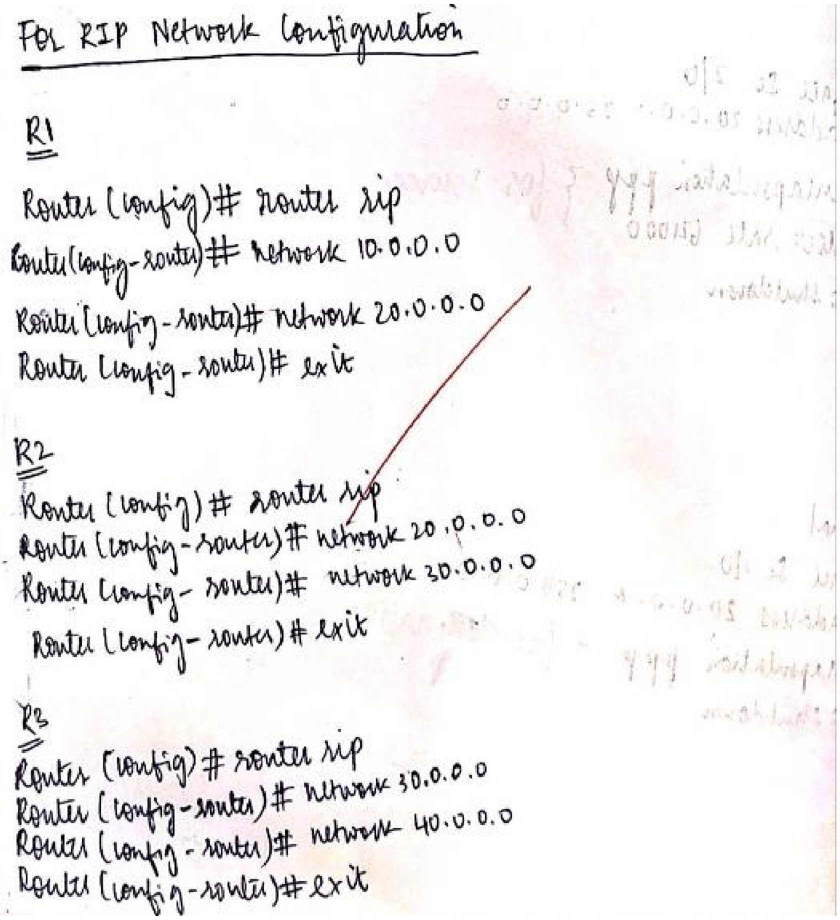
```
Router > enable
Router # config terminal
Router (config)# interface se 2/0
Router (config-it)# ipaddress 20.0.0.2  255.0.0.0
Router (config-it) # encapsulation ppp - for destination
Router (config-it) # no shutdown
exit
```



```
Router(config)# interface se 3/0
Router(config-if)# ip address 30.0.0.1 255.0.0.0
Router(config-if)# encapsulation ppp  } for source
Router(config-if)# clock rate 64000
Router(config-if)# noshutdown
exit
```

R3 configuration

```
Router> enable
Router# config terminal
Router(config)# interface fastEthernet 0/0
Router(config-if)# ip address 40.0.0.2 255.0.0.0
Router(config-if)# no shutdown
exit
Router(config)# interface se 2/0
Router(config-if)# ip address 30.0.0.2 255.0.0.0
Router(config-if)# encapsulation ppp  - for destination
Router(config-if)# no shutdown
exit
```



For RIP Network Configuration

R1

```
Router (config)# router rip
Router(config-router)# network 10.0.0.0
Router(config-router)# network 20.0.0.0
Router (config-router)# exit
```

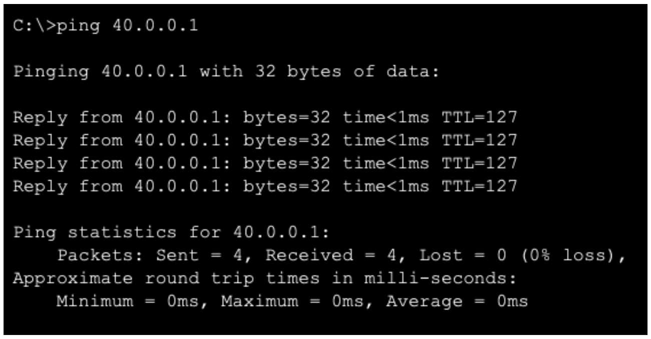
R2

```
Router (config) # router rip
Router (config-router)# network 20.0.0.0
Router (config- router)# network 30.0.0.0
Router (config- router) # exit
```

R3

```
Router (config) # router rip
Router (config-router) # network 30.0.0.0
Router (config - router)# network 40.0.0.0
Router (config-router)# exit
```

Snapshot of Output-

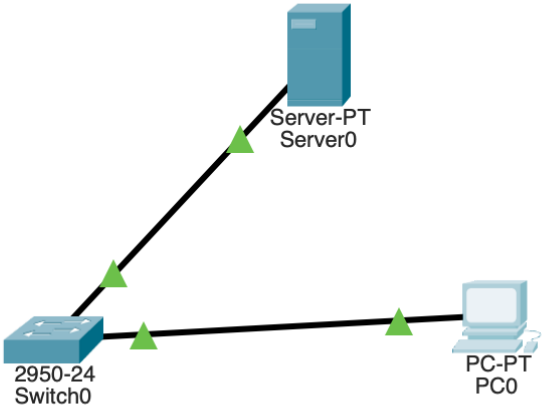


```
C:\>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 time<1ms TTL=127
Reply from 40.0.0.1: bytes=32 time<1ms TTL=127
Reply from 40.0.0.1: bytes=32 time<1ms TTL=127
Reply from 40.0.0.1: bytes=32 time<1ms TTL=127

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

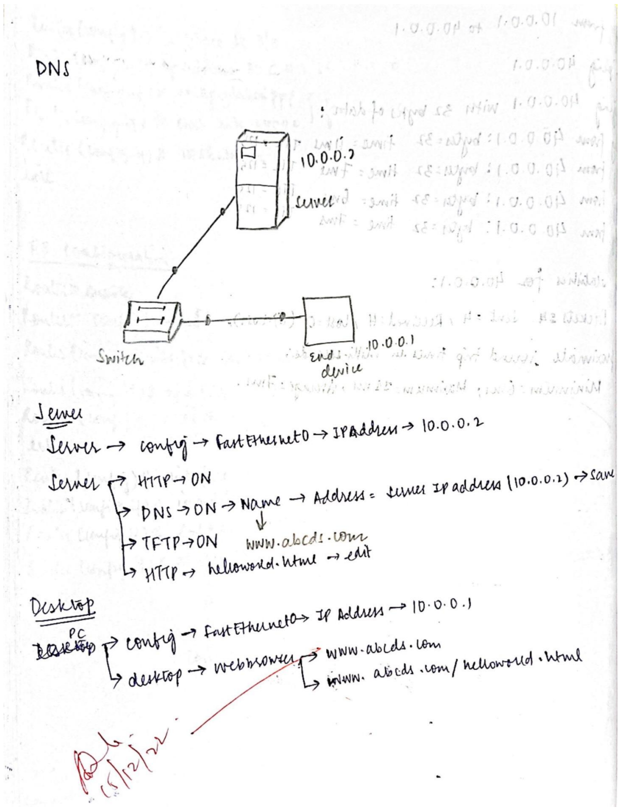
Experiment 6

Aim of the program- Demonstration of WEB server and DNS using Packet Tracer

Topology-



Procedure-


10.0.0.2
server
Switch
End device
10.0.0.1

DNS

Server

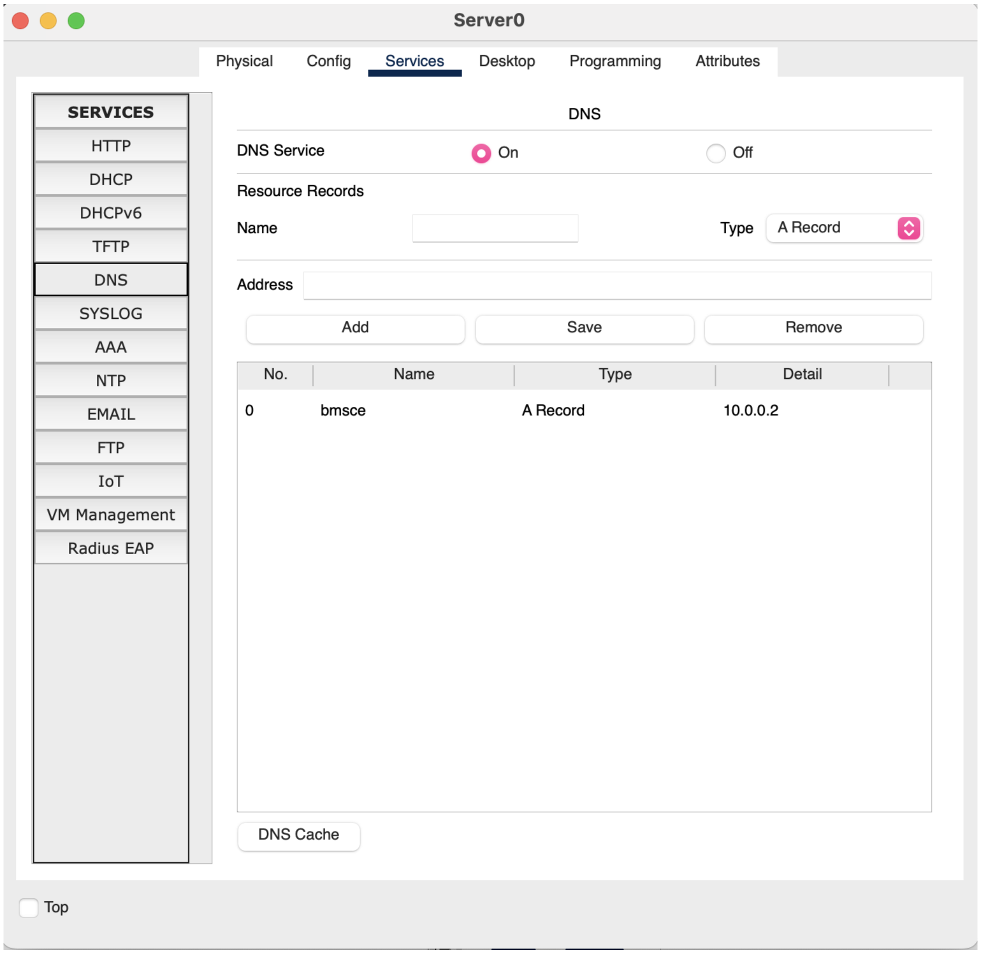
Server → config → FastEthernet0 → IPAddress → 10.0.0.2

Server → HTTP → ON
→ DNS → ON → Name → Address = server IP address (10.0.0.2) → save
↓ www.abcds.com
→ TFTP → ON
→ HTTP → helloworld.html → edit

Desktop

~~Desktop~~ PC → config → FastEthernet0 → IP Address → 10.0.0.1
→ desktop → webbrowser → www.abcds.com
→ www. abcds .com/ helloworld .html

15/12/22



DNS

DNS Service ● On ○ Off

Resource Records

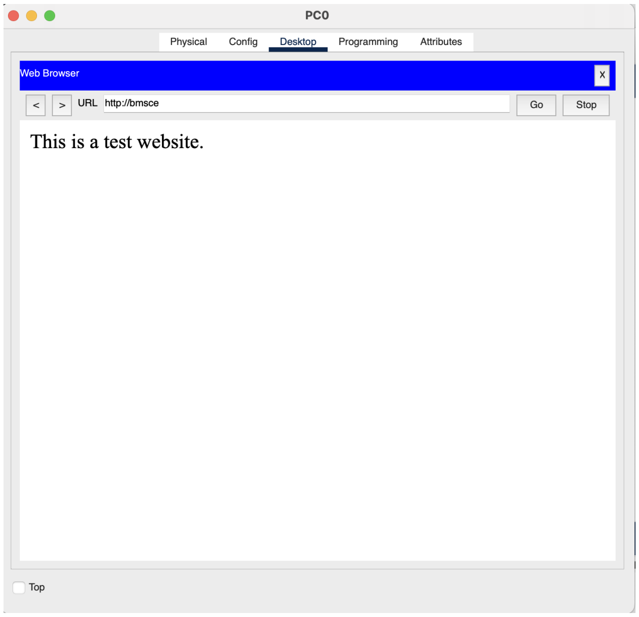
Name

Type A Record

Address

No.	Name	Type	Detail
0	bmsce	A Record	10.0.0.2

Snapshot of Output-



URL http://bmsce

This is a test website.

Cycle-2

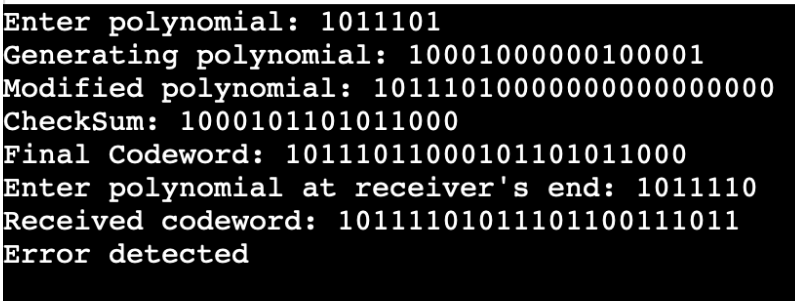
Experiment 1

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

```
import java.util.*;
public class crc{
   public static int n;
   public static void main(String[] args)
   {
      Scanner in=new Scanner(System.in);
      crc ob=new crc();
      String code,copy,zero="0000000000000000";
      System.out.print("Enter polynomial: ");
      code=in.nextLine();
      System.out.println("Generating polynomial: 10001000000100001");
      n=code.length();
      copy=code;
      code+=zero;
      System.out.println("Modified polynomial: "+code);
      code=ob.divide(code);
      System.out.println("CheckSum: "+code.substring(n));
      copy=copy.substring(0,n)+code.substring(n);
      String results=copy;
      System.out.println("Final Codeword: "+copy);
      System.out.print("Enter polynomial at receiver's end: ");
      code=in.nextLine();
      n=code.length();
      copy=code;
      code+=zero;
```

```
    code=ob.divide(code);
    copy=copy.substring(0,n)+code.substring(n);
    if (copy.equals(results)==true)
    {
        System.out.println("No Error Detected");
    }
    else
    {
        System.out.println("Received codeword: "+copy);
        System.out.println("Error detected");
    }
}
public String divide(String s){
    int i,j;
    char x;
    String div="10001000000100001";

    for(i=0;i<n;i++){
        x=s.charAt(i);

        for(j=0;j<17;j++){
            if(x=='1'){
                if(s.charAt(i+j)!=div.charAt(j))
                    s=s.substring(0,i+j)+"1"+s.substring(i+j+1);
                else
                    s=s.substring(0,i+j)+"0"+s.substring(i+j+1);
            }
        }
        }
        return s;
    }
}
```

Output-



```
Enter polynomial: 1011101
Generating polynomial: 10001000000100001
Modified polynomial: 10111010000000000000000
CheckSum: 1000101101011000
Final Codeword: 10111011000101101011000
Enter polynomial at receiver's end: 1011110
Received codeword: 10111101011101100111011
Error detected
```

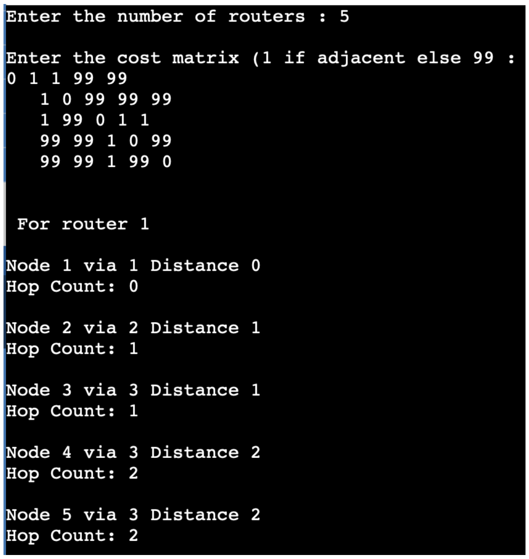
Experiment 2

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

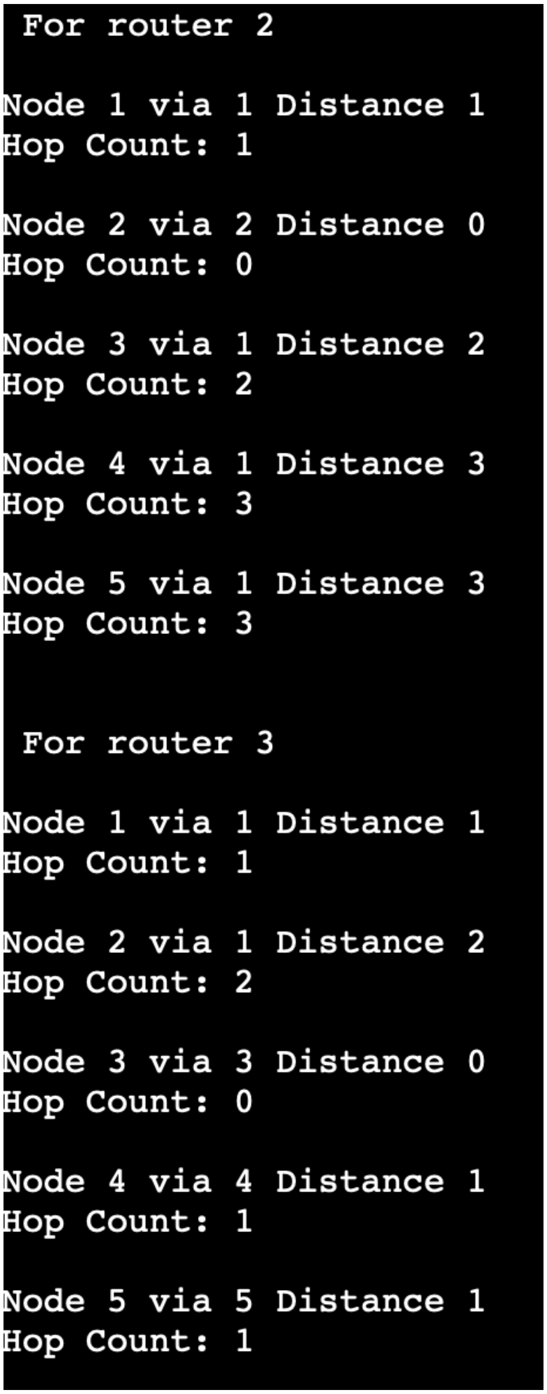
```
#include<stdio.h>
struct node
{
    unsigned dist[20];
    unsigned from[20];
    int hop[10];
}rt[10];

int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of routers : ");
    scanf("%d",&nodes);//Enter the nodes
    printf("\nEnter the cost matrix (1 if adjacent else 99 :\n");
 for(i=0;i<nodes;i++)
 {
    for(j=0;j<nodes;j++)
    {
       scanf("%d",&costmat[i][j]);
       costmat[i][i]=0;
       if (costmat[i][j]>0)
       {
          rt[i].hop[j]=1;
       }
       else
          rt[i].hop[j]=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 arbitary 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].hop[j]=rt[i].hop[k]+rt[k].hop[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("\t\nNode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
                printf("\nHop Count: %d\n",rt[i].hop[j]);
            }
        }
    printf("\n\n");
    getch();
}
```

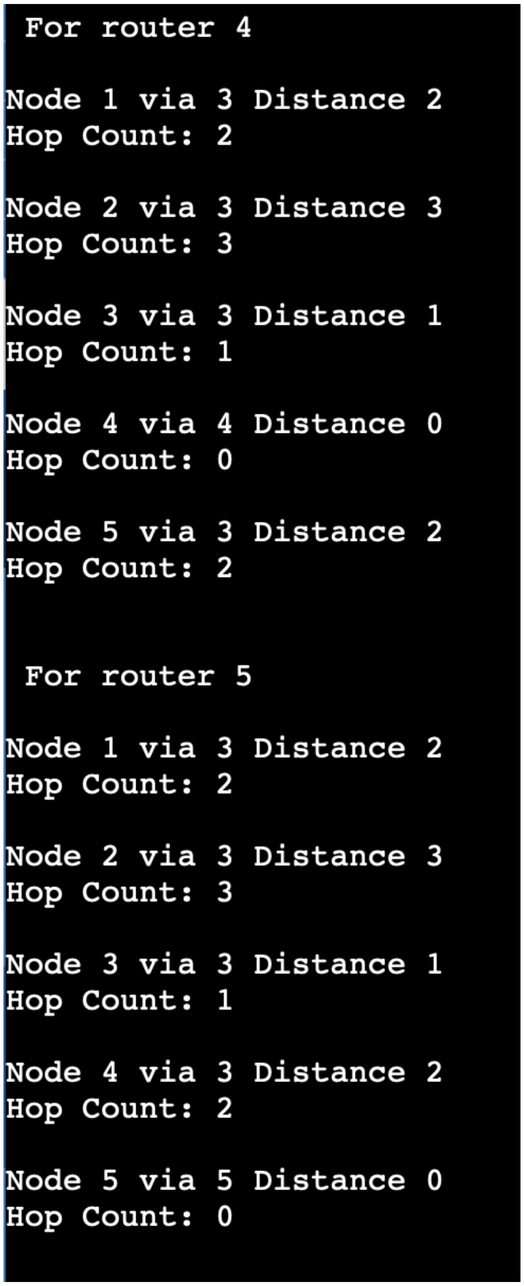
Output-



```
Enter the number of routers : 5

Enter the cost matrix (1 if adjacent else 99 :
0 1 1 99 99
   1 0 99 99 99
   1 99 0 1 1
   99 99 1 0 99
   99 99 1 99 0


 For router 1

Node 1 via 1 Distance 0
Hop Count: 0

Node 2 via 2 Distance 1
Hop Count: 1

Node 3 via 3 Distance 1
Hop Count: 1

Node 4 via 3 Distance 2
Hop Count: 2

Node 5 via 3 Distance 2
Hop Count: 2
```



```
 For router 2

Node 1 via 1 Distance 1
Hop Count: 1

Node 2 via 2 Distance 0
Hop Count: 0

Node 3 via 1 Distance 2
Hop Count: 2

Node 4 via 1 Distance 3
Hop Count: 3

Node 5 via 1 Distance 3
Hop Count: 3


 For router 3

Node 1 via 1 Distance 1
Hop Count: 1

Node 2 via 1 Distance 2
Hop Count: 2

Node 3 via 3 Distance 0
Hop Count: 0

Node 4 via 4 Distance 1
Hop Count: 1

Node 5 via 5 Distance 1
Hop Count: 1
```



```
 For router 4

Node 1 via 3 Distance 2
Hop Count: 2

Node 2 via 3 Distance 3
Hop Count: 3

Node 3 via 3 Distance 1
Hop Count: 1

Node 4 via 4 Distance 0
Hop Count: 0

Node 5 via 3 Distance 2
Hop Count: 2


 For router 5

Node 1 via 3 Distance 2
Hop Count: 2

Node 2 via 3 Distance 3
Hop Count: 3

Node 3 via 3 Distance 1
Hop Count: 1

Node 4 via 3 Distance 2
Hop Count: 2

Node 5 via 5 Distance 0
Hop Count: 0
```

Experiment 3

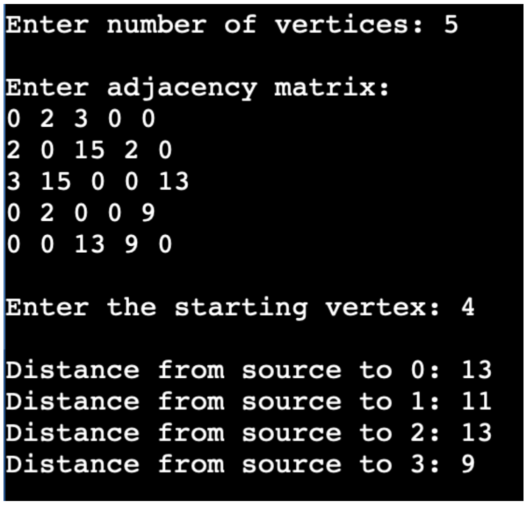
Program- Implement Dijkstra’s algorithm to compute the shortest path for a given topology.

```
#include <stdio.h>
#define INFINITY 9999
#define MAX 10

void Dijkstra(int Graph[MAX][MAX], int n, int start)
{

   int cost[MAX][MAX], distance[MAX], pred[MAX];
   int visited[MAX], count, mindistance, nextnode, i, j;
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         if (Graph[i][j] == 0)
            cost[i][j] = INFINITY;
         else
            cost[i][j] = Graph[i][j];

   for (i = 0; i < n; i++)
   {
      distance[i] = cost[start][i];
      pred[i] = start;
      visited[i] = 0;
   }

   distance[start] = 0;
   visited[start] = 1;
   count = 1;
```

```
while (count < n - 1)
{
    mindistance = INFINITY;

    for (i = 0; i < n; i++)
        if (distance[i] < mindistance && !visited[i])
        {
            mindistance = distance[i];
            nextnode = i;
        }

    visited[nextnode] = 1;
    for (i = 0; i < n; i++)
        if (!visited[i])
            if (mindistance + cost[nextnode][i] < distance[i])
            {
                distance[i] = mindistance + cost[nextnode][i];
                pred[i] = nextnode;
            }
    count++;
}

for (i = 0; i < n; i++)
    if (i != start)
    {
        printf("\nDistance from source to %d: %d", i, distance[i]);
    }
```

```
}
int main()
{
   int Graph[MAX][MAX], i, j, n, u;
   printf("\nEnter number of vertices: ");
   scanf("%d",&n);
   printf("\nEnter adjacency matrix: \n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
         scanf("%d",&Graph[i][j]);
      }
   }
   printf("\nEnter the starting vertex: ");
   scanf("%d",&u);
   Dijkstra(Graph, n, u);
 return 0;
}
```

Output-



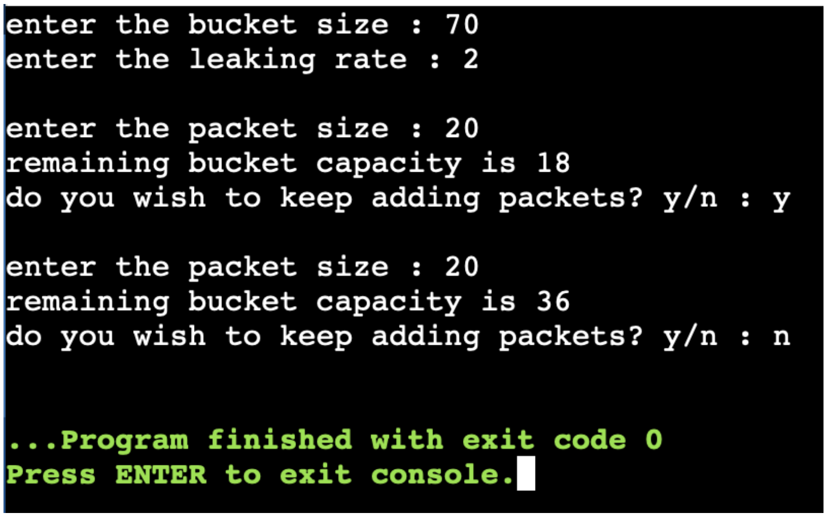
```
Enter number of vertices: 5

Enter adjacency matrix:
0 2 3 0 0
2 0 15 2 0
3 15 0 0 13
0 2 0 0 9
0 0 13 9 0

Enter the starting vertex: 4

Distance from source to 0: 13
Distance from source to 1: 11
Distance from source to 2: 13
Distance from source to 3: 9
```

Experiment 4

Program- Write a program for congestion control using Leaky bucket algorithm.

```
#include <iostream>
using namespace std;
int main() {
    int capacity=0,packet=0,bsize=0,rate=0;
    char ans='y';
    cout<<"enter the bucket size : ";
    cin>>capacity;
    cout<<"enter the leaking rate : ";
    cin>>rate;
    while(ans=='y')
    {
        cout<<"\nenter the packet size : ";
        cin>>packet;
        if((bsize+packet) > capacity)
        {
            cout<<"\n buffer full at the moment ";
        }
        else if((bsize+packet) <= capacity)
        {
            bsize+=packet;
        }
        bsize-=rate;
        cout<<"remaining bucket capacity is "<<bsize;
        cout<<"\ndo you wish to keep adding packets? y/n : ";
        cin>>ans;
    }
    return 0;
}
```

Output-



```
enter the bucket size : 70
enter the leaking rate : 2

enter the packet size : 20
remaining bucket capacity is 18
do you wish to keep adding packets? y/n : y

enter the packet size : 20
remaining bucket capacity is 36
do you wish to keep adding packets? y/n : n


...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment 5

Program- 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='localhost'
serverPort = 9999
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()
  file=open(sentence,"r")
  l=file.read(1024)
  connectionSocket.send(l.encode())
  file.close()
  connectionSocket.close()
```

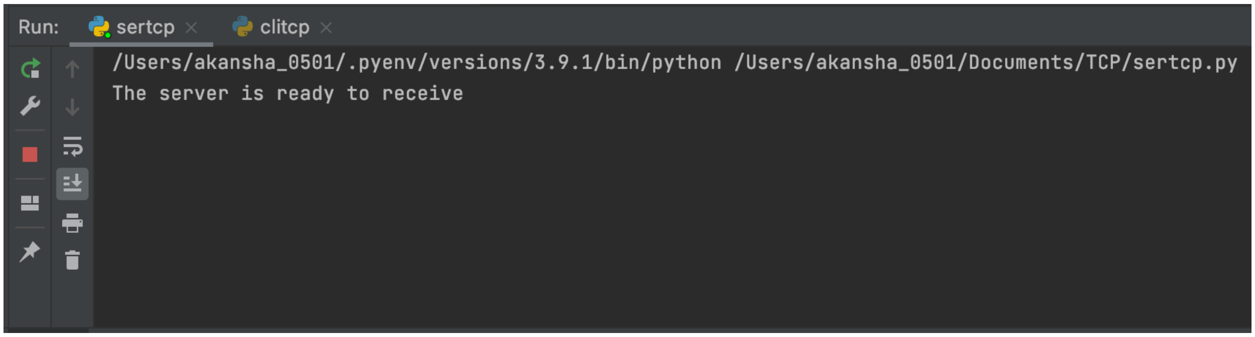
Client-

```
from socket import *
serverName = 'localhost'
serverPort = 9999
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()
```

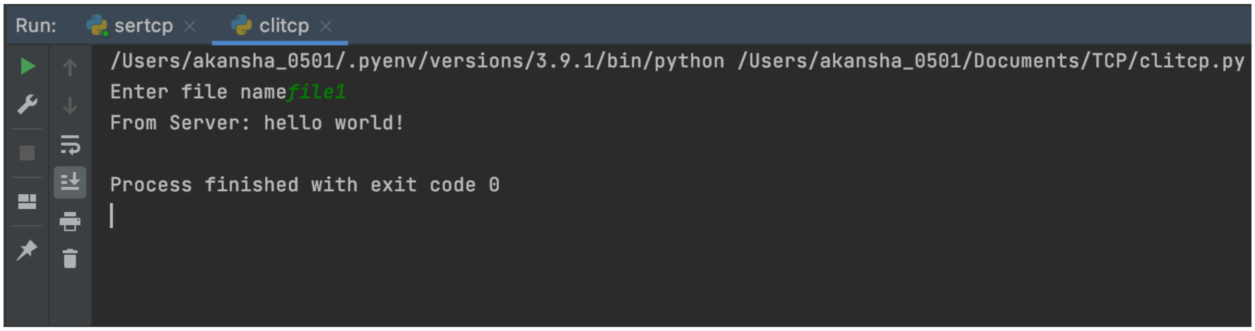
Output-

Server-



```
/Users/akansha_0501/.pyenv/versions/3.9.1/bin/python /Users/akansha_0501/Documents/TCP/sertcp.py
The server is ready to receive
```

Client-



```
/Users/akansha_0501/.pyenv/versions/3.9.1/bin/python /Users/akansha_0501/Documents/TCP/clitcp.py
Enter file namefile1
From Server: hello world!

Process finished with exit code 0
```

Experiment 6

Program- 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.

Server-

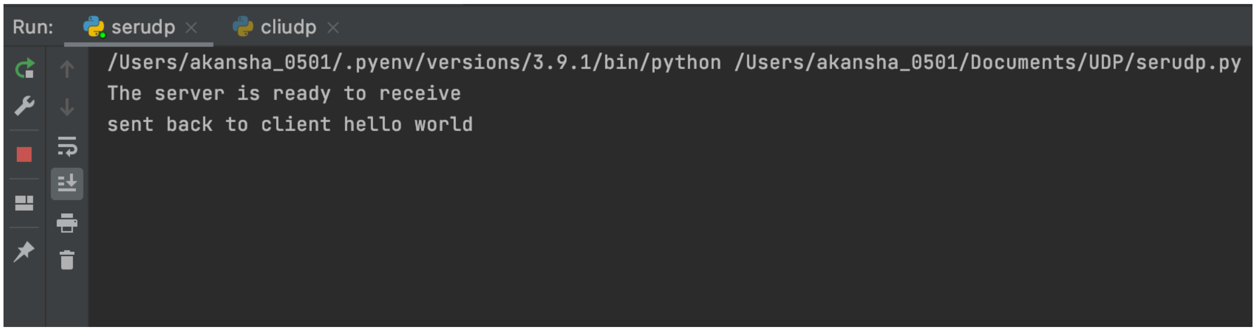
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence,clientAddress = serverSocket.recvfrom(2048)
    file=open(sentence,"r")
    l=file.read(2048)
    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)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

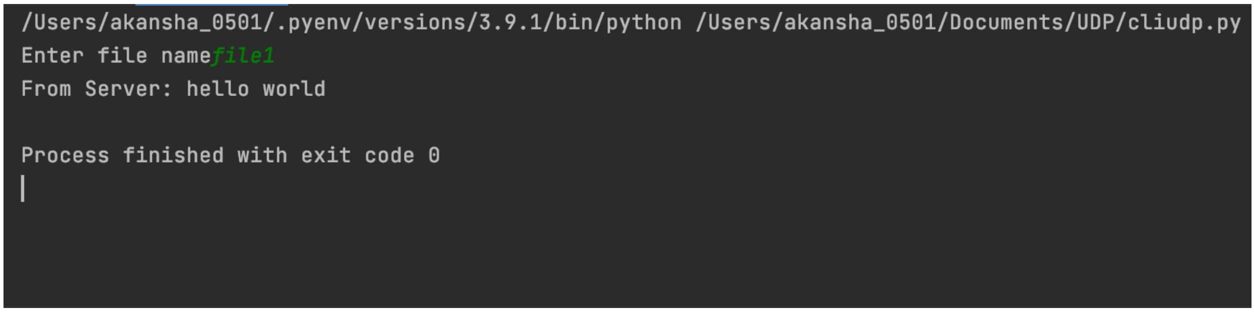
Output-

Server-



```
/Users/akansha_0501/.pyenv/versions/3.9.1/bin/python /Users/akansha_0501/Documents/UDP/serudp.py
The server is ready to receive
sent back to client hello world
```

Client-



```
/Users/akansha_0501/.pyenv/versions/3.9.1/bin/python /Users/akansha_0501/Documents/UDP/cliudp.py
Enter file namefile1
From Server: hello world

Process finished with exit code 0
```