8/11/22

LAB-1

Aim - A simple from sourn to destination. using hub and switch as connecting devices.

Procedure -

→ Select 3-4 generic end devices and one generic hub. Connect them with default connections.

→ Configure the end devices under the fast ethernet tab. Fill in different I.P address of the selected end devices.

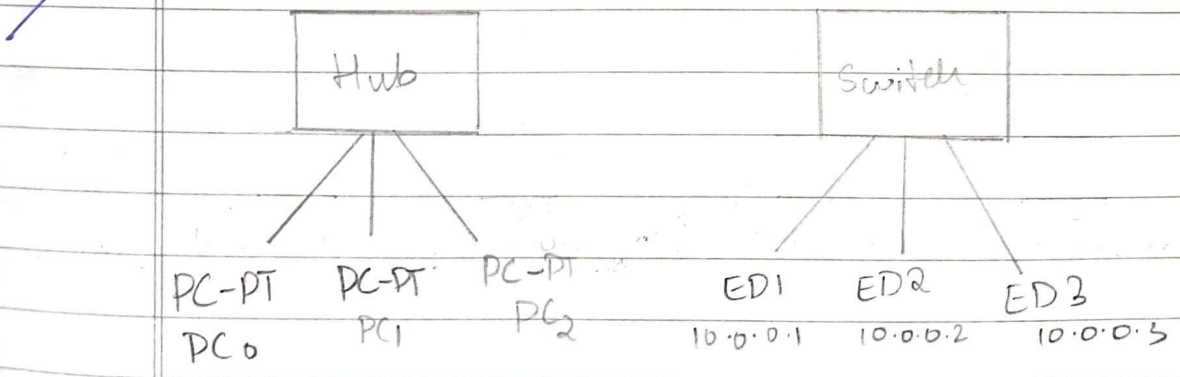
In simulation mode:

→ Add the message icon on source end device and then on destination end devices

→ capture the simulations

→ Use command prompt for pinging messages.

Topology:



Observation.

Hub broadcasts the message to all the end devices and will be accepted by the correct end devices which matches the IP address.

If a active host isn't connected to a internetwork, a message cannot be pinged and hence response will be timed out

Result: Transmitting of PDUs were successful between source and dest.

Experiment-2

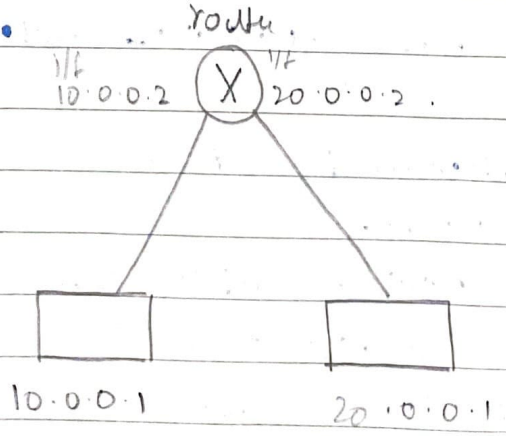
Aim: Create an end to end connection using router and configure IP address. Explore ping response, destination unreachable, ~~req~~ request timed out.

Procedure :- configure IP address of PCs.
configure router.
CLI:
no
enable
config terminal
interface Fa0/0
ip address 10.0.0.2 255.00.0
no shutdown
exit

PC1: 10.0.0.2 PC2: 20.0.0.2

- update gateway address of PCs.
- ping interface 10.0.0.2 and 20.0.0.1

Topography:



Observation:

PC 10.0.0.1 after gateway is set.

ping 10.0.0.2.

Replay from 10.0.0.2. byte =32
time = 0ms TTL=225

Reply from 10.0.0.2 bytes=32 time=0ms TTL=

Reply from 10.0.0.2. bytes = 32 time=0ms TTL=

Reply from 10.0.0.2 bytes = 32 time=0ms TTL=

ping statistics for 10.0.0.2:

Packets: sent = 4, Recieved = 4, lost = 0 (0% loss)

Approximate round trip times in milliseconds:

Minimum = 0ms, maximum = 0ms, Average = 0ms

ping 20.0.0.1

pinging 20.0.0.1 with 32 byte of data:

Request timed out.

Reply from 20.0.0.1 byte = 32 time=0ms TTL=127

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

Replay from 20.0.0.1 bytes = 32 time=0ms TTL=127

ping 20.0.0.2

(sucessful)

pinging 20.0.0.2 with 32 bytes of data:

Replay from 20.0.0.2 bytes = 32 bytes time=1ms
TTL=225

Befor gateway is set.

ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data:

~~Reply from 20.0.0.2.~~

Request timed out.

Request timed out.

Request timed out.

Request timed out.

ping statistics for 20.0.0.2:
Packets: sent 4, received =0, lost = 4 (100% los

Result:

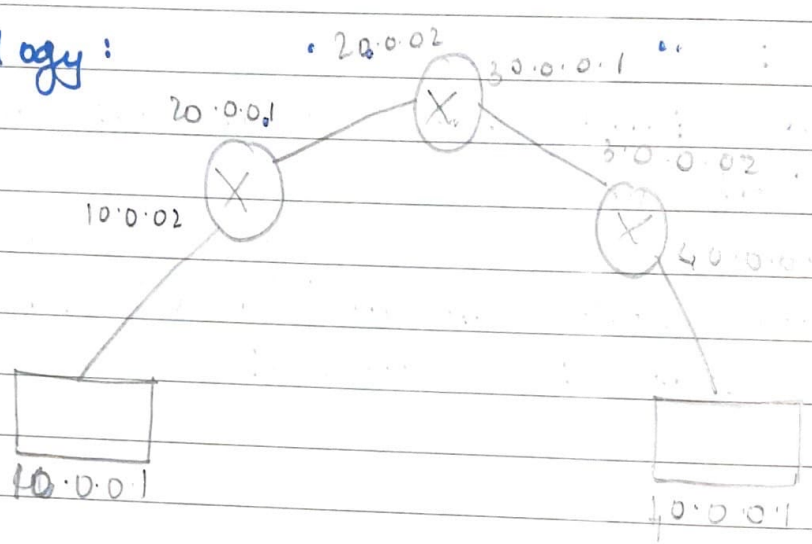
* Request timed out when gateway address is not set.

* When both gateway and IP address set ping is successful.

Experiment-3

Aim: To configure static IP address to the router.

Topology:



Procedure:

* Configure IP address of PC's.
* Configure router interface
* Set gateway address of PC's
* For each router check network it is directly connected to.
  router # show ip route.
* Connect to other networks in configure mode.
  router (config)# ip route 30.0.0.0 255.0.0.0 20.0.0.2
  Static → network, sub-network, next hop

Observation:

Before manually connecting networks:

ping 40.0.0.1

ping 40.0.0.1 with 32 bytes of data.
Reply from 10.0.0.2 Destination host unreachable

Reply from 10.0.0.2 Destination host unreachable
Reply from 10.0.0.2 Destination host unreachable

ping statistics for 40.0.0.1
packets: sent = 4, received = 0, lost = 4 (100% loss)

ping 20.0.0.2

pinging 20.0.0.2 with 32 bytes of data:
Request timed out
Request timed out
Request timed out

Ping statistics for 20.0.0.2:
Packets = sent = 4, Received = 0, lost = 4 (100% loss

the counting network.

ping 40.0.0.1

Pinging 40.0.0.1 with 32 byte of data:
Reply from 40.0.0.1 : byte = 32 time = 2ms TTL = 2
Reply from 40.0.0.1 : byte = 32 time 21 ms TTL = 1
Reply from 40.0.0.1 : byte = 32 time = 14ms TTL = 125

Ping statistics for 40.0.0.1:
Packets: sent = 4, recive = 4, lost (0% loss)
Approximate round trip time in milliseconds:
minimum = 2ms, Maximum = 21ms, Average = 13ms.

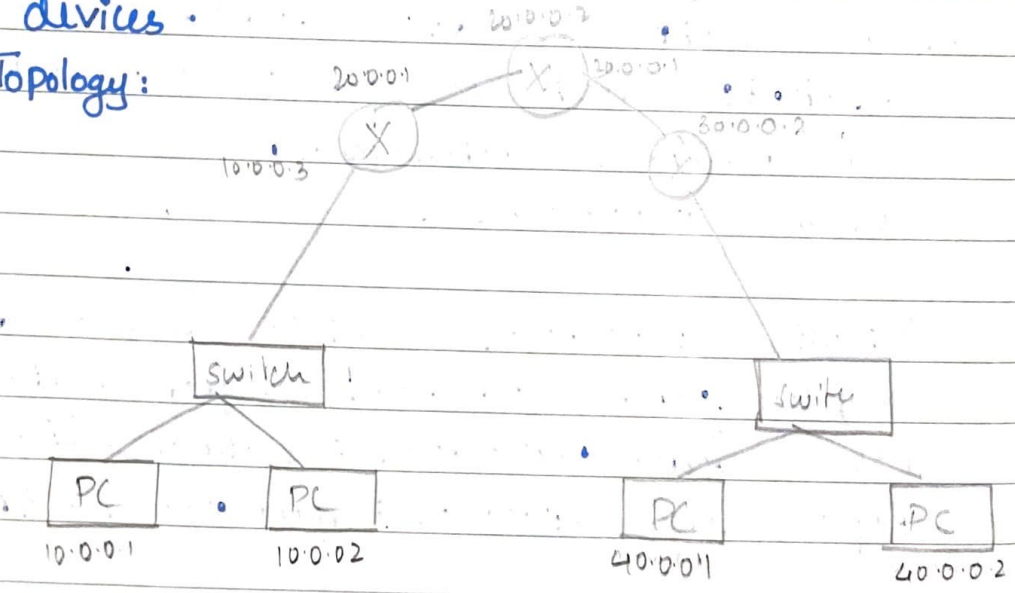
Result:- successful ping only on manually connecting network.

Experiment-4:

Aim: Configuring default route to the router.

Procedure: Connect switches to the routers.
configure IP address of PC's.
configure router interface.
set the gateway address.
set the ~~static~~ default route for the end devices.

Topology:



Default = ip route 0.0.0.0 0.0.0.0 20.0.0.2
default next hop.

Observation:

PC1 : Ping 10.0.0.3
10.0.0.1

Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3 bytes = 32 time=1ms TTL=
Reply from 10.0.0.3 bytes = 32 time=6ms TTL=
Reply from 10.0.0.3 bytes=32 time=0ms TTL=
Reply from 10.0.0.3 bytes=32 time=1ms TTL=

Ping analytics for 10.0.0.3:
Packets: Sent=4, Received=4, Lost=0

ping 20.0.0.1

pinging 20.0.0.1 with 32 bytes of data:
Reply from 20.0.0.1 : byte=32 time=0ms TTL=255
Reply from 20.0.0.1 : byte=32 time=5ms TTL=255
Reply from 20.0.0.1 : bytes=32 time=0ms TTL=255
Reply from 20.0.0.1 : bytes=32 time=0ms TTL=255
Ping statistics for 20.0.0.1:
packets: sent=4, Received=4, Lost=0

ping 20.0.0.2
pinging 20.0.0.2 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 20.0.0.2:
packets: sent=4, Received=0, Lost=4(100% loss)

ping 30.0.0.1
pinging 30.0.0.1 with 32 bytes of data:
Reply from 10.0.0.3
Reply from 10.0.0.3
Reply from 10.0.0.3
Reply from 10.0.0.3

After setting default router:

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=1ms TTL=254
Request timed out.

Reply from 30.0.0.1 : bytes = 32 time = 1ms TTL = 254

Ping statistics for 30.0.0.1:
packets : sent = 4, received = 2, lost = 2 (50% loss)

ping 40.0.0.2.
Ping 40.0.0.2 with 32 bytes of data:
Reply from 40.0.0.2 : bytes = 32 time = 11ms TTL = 12
Reply from 40.0.0.2 : bytes = 32 time = 3ms TTL = 12
Reply from 40.0.0.2 : bytes = 32 time = 3ms TTL = 12
Reply from 40.0.0.2 : bytes = 32 time = 3ms TTL = 12

ping statistics for 40.0.0.2:
packets: sent = 4, received = 4, lost = 0 (0% loss)

Result:
Successfully set default route to the router.

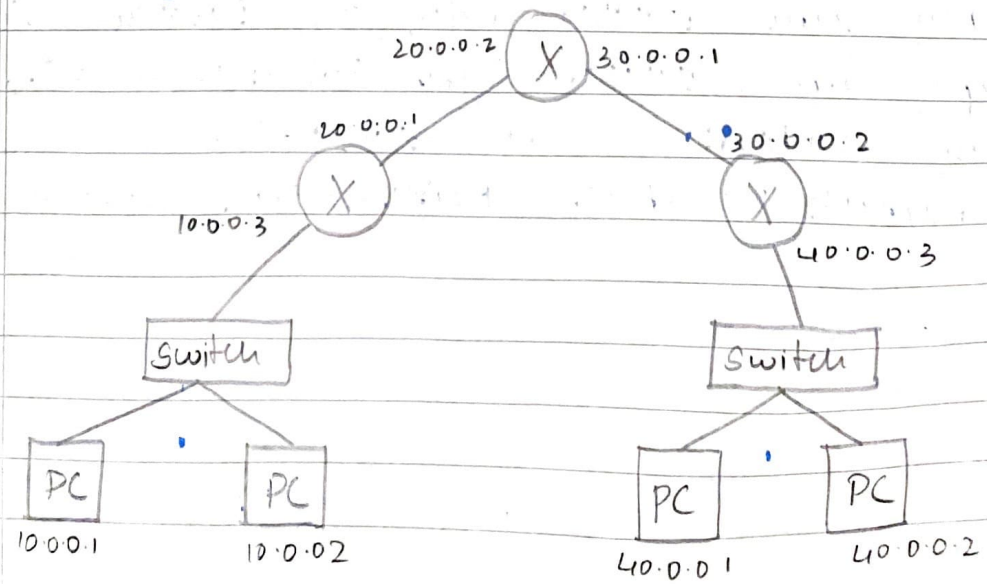
Experiment-5

Aim: Configuring RIP protocols in Router

Procedure:
Select the end devices and make all the connections.
Configure IP address of PC's
Configure router interface.
CLI (only for serial connections)
encapsulation ppp
Clock rate 64000 - (only for end routers)
no shutdown.
exit
- update gateway address of PC's
- configure default route to the router.

CLI
ip route 0.0.0.0 0.0.0.0 20.0.0.2
router rip
network 10.0.0.0
network 20.0.0.0

Topology:



Observation:

PC1: ping 10.0.0.3
10.0.0.1

pinging 10.0.0.3 with 32 bytes of data.
Reply from 10.0.0.3 bytes = 32 time=1ms TTL=
Reply from 10.0.0.3 bytes=32 time=6ms TTL=
Reply from 10.0.0.3 bytes=32 time=0ms TTL=
Reply from 10.0.0.3 byte=32 time=1ms TTL=25

ping analysis for 10.0.0.3
Packets: sent=4, Recieved=4, lost=0 (0% loss

>ping 20.0.10.1
pinging 20.0.0.1 with 32 bytes of data
Reply from 20.0.0.1 bytes=32 time=0ms TTL=255
Reply from 20.0.0.1 bytes=32 time=5ms TTL=255
Reply from 20.0.0.1 bytes=32 time=0ms TTL=255
Reply from 20.0.0.1 Bytes=32 time=0ms TTL=255
ping statistics for 20.0.0.1
Packets: sent=4, Recieved=4, lost=0 (0% loss)

>ping 40.0.0.2.
pinging 40.0.0.2 with 32 bytes of data.
Reply from 40.0.0.2 bytes=32 time=11ms TTL=125
Reply from 40.0.0.2 bytes=32 time=16ms TTL=125
Reply from 40.0.0.2 bytes=32 time=8ms TTL=125
Reply from 40.0.0.2 byte=32 time=2ms TTL=125
ping statistics for 40.0.0.2.
Packets: sent=4, Recieved=4. Lost=0 (0% loss

Result:

We use PPP protocol while configuring ip address of interface of 2 routers. It is point to point protocol of the data link layer that is used to transit multiprotocol data between two directly connected devices.

15/12

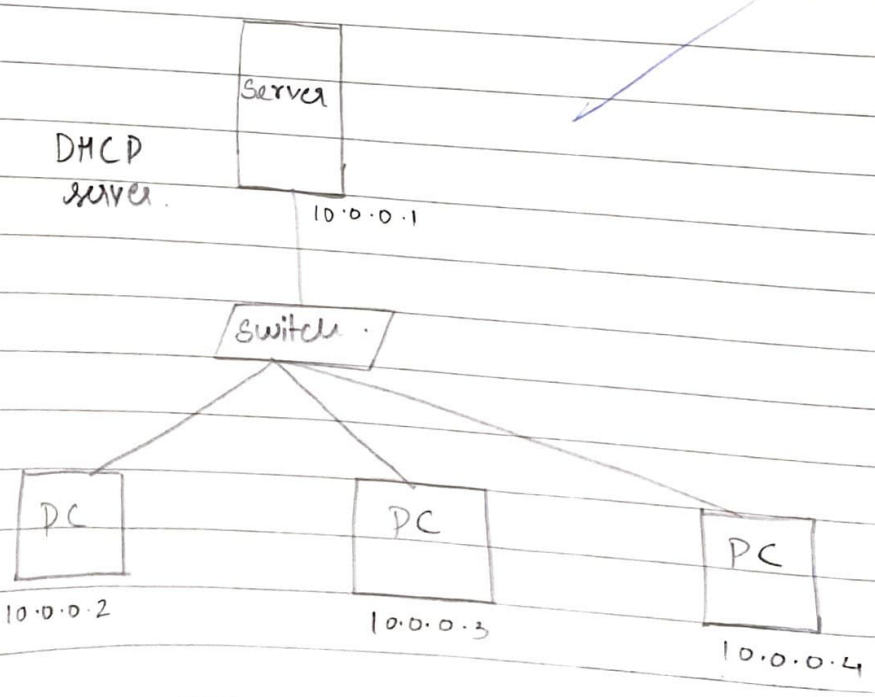
/12/22

Experiment-6

Aim: Configuring DHCP server.

Procedure:
Select three pc's and server from end devices.
Select switch and connect them.
Configure IP address of server as 10.0.0.1
- In server - go to services and click on DHCP and set it to "on". Then set the start IP address as 10.0.0.1 and click on save.
To configure PC's
- select DHCP in IP configurations.

Topology:



Observations:

Once the DHCP configuration is made the IP address of the PC's is dynamically set. The IP address of PC's starts from the starting IP address of the server.

Result:

DHCP is a client/server protocol that automatically provides an IP host with its IP address and other related configuration info such as the subnet mask and default gateway.

15/12

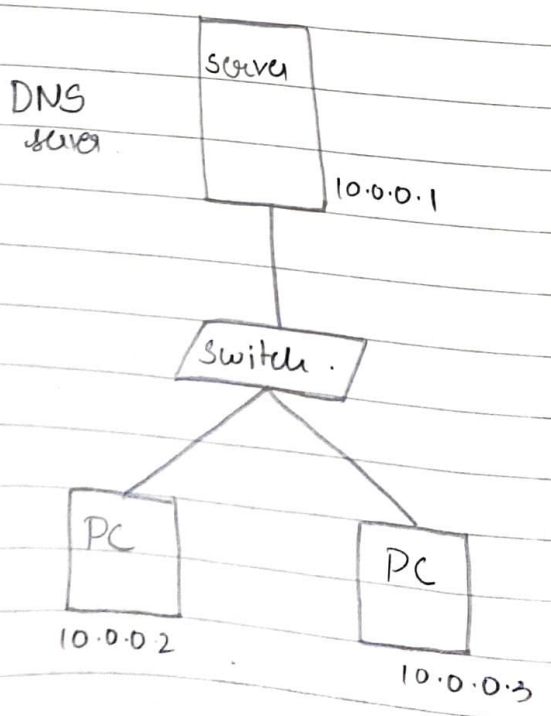
15/12/22

Experiment - 7

Aim: Demonstration of WEB server and DNS using Packet Tracer

Procedure:
Select PC's and server from end Devices and connect them.
Configure IP address of server and PC's
Set gateway for PC's
In Server open services and select HTTP and set it to "on".
Then select DNS - set it to "on"
Type the Name and Address (10.0.0.1) and click add.
- Open web browser on the PC's and type the URL and click go.

Topology:



Observation:

When we type the domain name that we added in DNS server in the web browser of the PC's we get the response as website.

Result:

The DNS server is a server that is specifically used for matching website host names (like bms.com) to their corresponding internet protocol (IP address)

15/12

29/12/22

LAB 8

Aim - Write a program for error detection using CRC 16 bit

```
import java.util.*;
class crc
{
   void div(int a[], int k)
   {
      int gx[]={1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1};
      int count=0;
      for (int i=0; i<k; i++)
      {
         if (a[i]==gx[0])
         {
            for (int j=i; j<17+i; j++)
            {
               a[j]= a[j] ^ gx[count++];
            }
            count=0;
         }
      }
   }

   public static void main (String args[])
   {
      int a[] = new int[50];
      int b[] = new int[50];
      int len, k;
      crc ob = new crc();
      System.out.println("Enter the length of
      Scanner scan = new Scanner(System.in);
```

```
len = scan.nextInt();
int flag=0;
System.out.println("Enter the Dataword:");
for(int i=0; i<len; i++)
{
   a[i]= scan.nextInt();
}
for (int i=0; i<16; i++)
{
   a[len++]=0;
}
K = len-16
for(int i=0; i<len; i++)
{
   b[i]=a[i];
}
ob.div(a,k);
for(int i=0; i<len; i++)
a[i]=a[i]^b[i];
System.out.println("Data to be transmitted:");
for(int i=0; i<len; i++)
{
   System.out.print(a[i]+" ");
}
System.out.println();
System.out.println("Enter the Recived data:");
for(int i=0; i<len; i++)
{
   a[i]= scan.nextInt();
}
ob.div(a,k);
for(int i=0; i<len; i++)
```

```
{
  if(a[i] != 0)
  {
     flag = 1;
     break;
  }
}
if(flag == 1)
System.out.println("error in data");
else
System.out.println("no error");
}
}
```

Output:

```
Enter the length of Data Frame:
7
Enter the Dataword:
1 0 1 1 1 0 1
Data output:
1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0
Data recived:
1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0
no error.
```

12/1/2023

5/1/23

LAB-9

Aim - Write a program for congestion using Leaky-bucket algorithm.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int no_of_queries, storage, output_pkt_size;
    int input_pkt_size, bucket_size, size_left;
    storage = 0
    int u
    printf("Enter the number of queries");
    scanf("%d", &no_of_queries);
    int query[no_of_queries];
    printf("Enter the number of packets to be
            transmitted for each query");
    for(int i=0; i<no_of_queries; i++)
    {
        scanf("%d", &query[i]);
    }
    int b;
    printf("Enter bucket size");
    scanf("%d", &bucket_size);
    printf("Enter the constant output packet rate\n");
    scanf("%d", &output_packet_size);
    for(int i=0; i<no_of_queries; i++)
    {
        size_left = bucket_size - storage;
        if(query[i] <= size_left)
        {
            storage += query[i]; }
```

```
else
{
  printf ("Packet loss = %d \n", query[i]);
}
printf ("Buffer size = %d out of bucket size = %d\n",
    storage, bucket_size);
    storage = output_pkt_size;
  }
return 0;
}
```

Output:

Enter the number of queries to be made: 4
Enter the number of packets to be transmitted for each query.
2
7
5
6
Enter the bucket size
10
Enter the constant output
1

Bucket size = 2
Bucket size = 8
Packet loss 5
Buffer size = 7
Packet loss 6
Buffer size 6

12/1/2023

12/1/23

LAB-10

Write a program for distance vector algorithm.

```
#include <stdio.h>
#include <stdlib.h>
int BellmanFord (int G[20][20], int V, int E, int edge[20][2])
{
   int i,u,v,k, distance[20], parent[20], S, flag=1;
   for(i=0; i<V; i++)
      distance[i] = 1000; parent[i] = -1;
      printf("Enter source:");
      scanf("%d",&S);
      distance[S-1]=0;
   for(i=0; i<V-1; i++)
   {
      for(k=0; k<E; k++)
      {
         u=edge[k][0], v=edge[k][1];
         if(distance[u]+G[u][v]<distance[v])
            distance[v] = distance[u] + G[u][v], parent[v]=u;
      }
   }
   for(k=0; k<E; k++)
   {
      u=edge[k][0], v=edge[k][1];
      if(distance[u]+G[u][v]<distance[v])
         flag=0;
   }
   if(flag)
      for(i=0; i<V; i++)
         printf("Vertex %d -> cost = %d parent = %d\n",
            i+1, distance[i], parent[i]+1);
   return flag; }
```

```
int main()
{
  int V, edge[20][2], G[20][20], i, j, k=0;
  printf("BELLMAN FORD\n");
  printf("Enter no. of vertices: ");
  scanf("%d", &V);
  printf("Enter graph in matrix form:\n");
  for(i=0; i<V; i++)
     for(j=0; j<V; j++)
     {
        scanf("%d", &G[i][j]);
        if(G[i][j]!=0)
           edge[k][0]=i, edge[k++][1]=j;
     }
  if(bellmanFord(G, V, k, edge))
     printf("\nNo negative weight cycle\n");
  else printf("\nNegative weight cycle exists\n");
  return 0;
}
```

Output:

```
BELLMAN FORD
Enter no. of vertices: 5
Enter graph.
0   2   2   99  99
2   0   99  3   99
99  99  0   6   4
99  3   6   0   4
99  99  4   5   0
Enter source = 1
Vertex 1 = cost = 0  parent = 0
Vertex 2 = cost = 2  parent = 1
Vertex 3 = cost = 2  parent = 1
No Negative cycle.
```

12/1/2023

12/1/23

LAB-11

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

```
#include < stdio.h >
#include <conio.h>
# define INFINITY 9999
# define MAX 10
void dijkstra (int G[MAX][MAX], int n, int startnode);
int main()
{
  int G[MAX][MAX], i, j, n, u;
  printf ("Enter no. of vertices : ");
  scanf ("%d", &n);
     printf ("\n Enter the adjacency matrix\n");
  for(i=0; i<n; i++)
    for (j=0.; j<n ;j++)
      scanf ("%d", &n);
       dijkstra (G, n, u);
          return 0; }
void dijkstra (int G[MAX][MAX], int n, int startnode)
{
  int cost [MAX][MAX], distance [MAX], pred [MAX];
  int visited [MAX], count, mindistance,
  next node, i, j;
  for (i=0; i<n; i++)
    for (j=0; j<n; j++)
      if (G[i][j] == 0)
        cost [i][j] = INFINITY;
      else cost [i][j] = G[i][j];
        for (i=0; i<n; i++)
        {
```