VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Name: Apurba Koirala

Reg no: 22BCE3799

Subject Code: BCSE308P

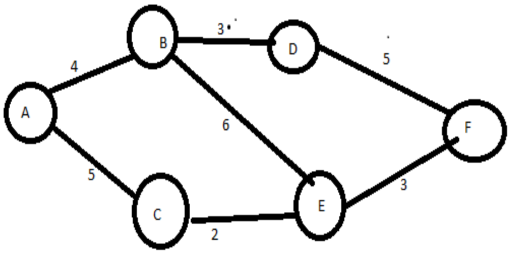
Course Title: Computer Networks Lab

Lab Slot: L31 + L32

Guided by: Dr. Arivoli A

Lab Assessment 4

1. Write a program to implement the Link state algorithm (Dijkstra's algorithm) to find the shortest path in the network from node "A".



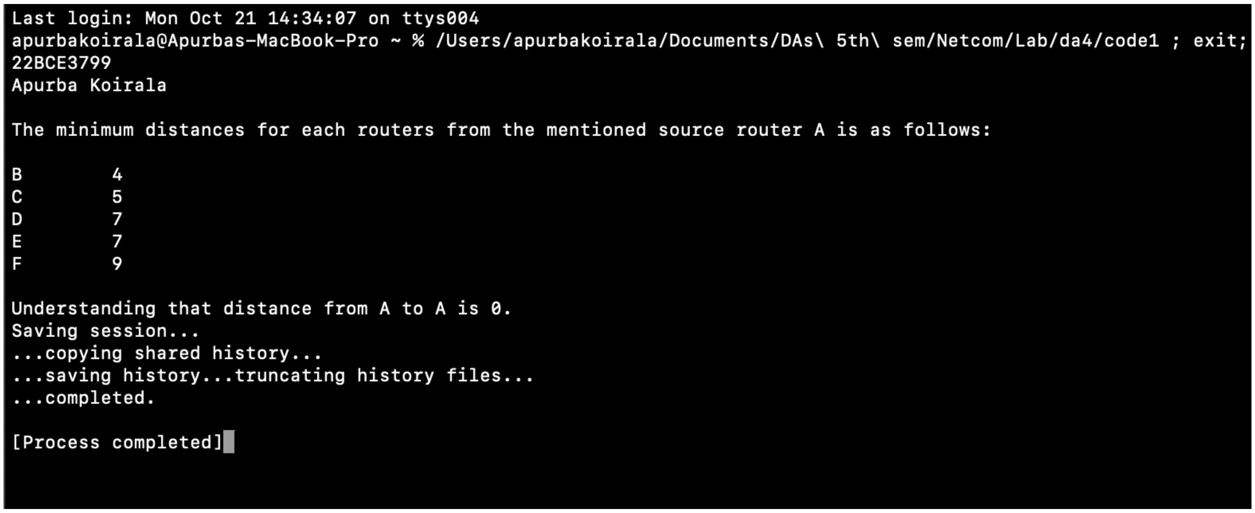
Solution:

Code:

```
#include <stdio.h>
#include <limits.h>

#define V 6

int minDistance(int dist[], int sptSet[]) {
   int min = INT_MAX, min_index;

   for (int v = 0; v < V; v++)
      if (sptSet[v] == 0 && dist[v] <= min)
         min = dist[v], min_index = v;

   return min_index;
}
```

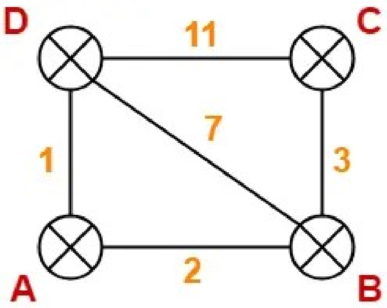
```
void printSolution(int dist[], int n) {
    printf("22BCE3799 \nApurba Koirala\n\n");
      printf("The minimum distances for each routers from the mentioned source router A is as follows: \n\n");
    for (int i = 1; i < V; i++)
        printf("%c \t %d\n", i + 'A', dist[i]);
    printf("\n");
    printf("Understanding that distance from A to A is 0.");
}

void dijkstra(int graph[V][V], int src) {
    int dist[V];
    int sptSet[V];

    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = 0;

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, sptSet);
        sptSet[u] = 1;

        for (int v = 0; v < V; v++)
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    printSolution(dist, V);
}

int main() {
    int graph[V][V] = {
        {0, 4, 5, 0, 0, 0},
        {4, 0, 0, 3, 0, 5},
        {5, 0, 0, 6, 2, 0},
        {0, 3, 6, 0, 0, 5},
        {0, 0, 2, 0, 0, 3},
        {0, 5, 0, 5, 3, 0},
    };

    dijkstra(graph, 0); // source node (router) A as per the question, if B was to be the source node, (graph,
1) was to be passed

    return 0;
}
```

Output:



```
Last login: Mon Oct 21 14:34:07 on ttys004
apurbakoirala@Apurbas-MacBook-Pro ~ % /Users/apurbakoirala/Documents/DAs\ 5th\ sem/Netcom/Lab/da4/code1 ; exit;
22BCE3799
Apurba Koirala

The minimum distances for each routers from the mentioned source router A is as follows:

B        4
C        5
D        7
E        7
F        9

Understanding that distance from A to A is 0.
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

2. Implement the Distance vector algorithm (Bellman-Ford) to calculate the routing tables for each route in the network.



Solution:

Code:

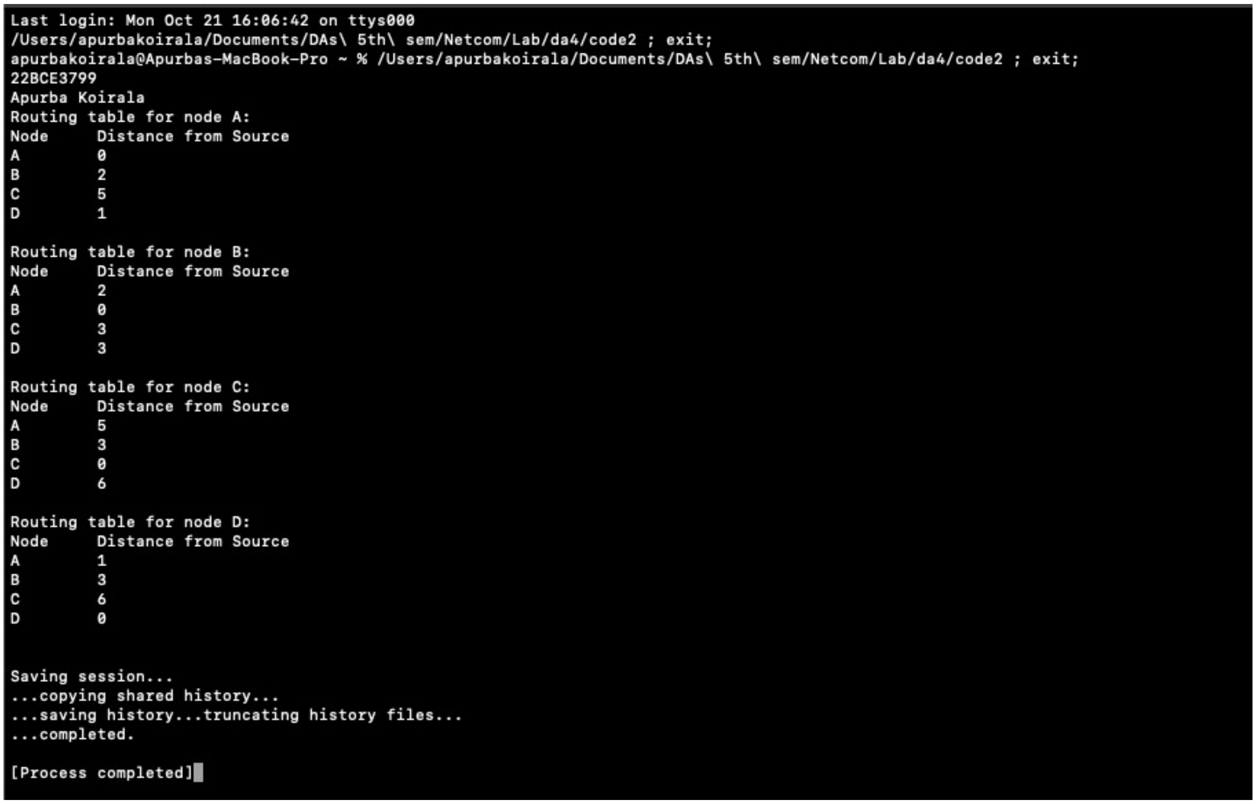
#include <stdio.h>

#include <limits.h>

#define NODES 4

```
struct Connection {
    int start, end, cost;
};

void displayTable(int distances[], int size) {
    printf("Node \t Distance from Source\n");
    for (int i = 0; i < size; i++) {
        printf("%c \t %d\n", i + 'A', distances[i]);
    }
}

void calculatePaths(struct Connection connections[], int numConnections, int source) {
    int distances[NODES];
    for (int i = 0; i < NODES; i++)
        distances[i] = INT_MAX;
    distances[source] = 0;

    for (int i = 0; i < NODES - 1; i++) {
        for (int j = 0; j < numConnections; j++) {

            int startNode = connections[j].start;

            int endNode = connections[j].end;

            int connectionCost = connections[j].cost;

            if (distances[startNode] != INT_MAX && distances[startNode] + connectionCost < distances[endNode])

                distances[endNode] = distances[startNode] + connectionCost;

        }

    }

    displayTable(distances, NODES);

}



int main() {

    //constructing the graph given in the question

    struct Connection connections[] = {

        {0, 1, 2}, {1, 0, 2}, {0, 3, 1}, {3, 0, 1}, {1, 2, 3},

        {2, 1, 3}, {2, 3, 11}, {3, 2, 11}, {3, 1, 7}, {1, 3, 7}

    };

    int numConnections = sizeof(connections) / sizeof(connections[0]);

    printf("22BCE3799\nApurba Koirala\n");
    //printing final routing table for each node.
    for (int i = 0; i < NODES; i++) {
        printf("Routing table for node %c:\n", i + 'A');
        calculatePaths(connections, numConnections, i);
        printf("\n");
    }
    return 0;
}
```

Output:



```
Last login: Mon Oct 21 16:06:42 on ttys000
/Users/apurbakoirala/Documents/DAs\ 5th\ sem/Netcom/Lab/da4/code2 ; exit;
apurbakoirala@Apurbas-MacBook-Pro ~ % /Users/apurbakoirala/Documents/DAs\ 5th\ sem/Netcom/Lab/da4/code2 ; exit;
22BCE3799
Apurba Koirala
Routing table for node A:
Node     Distance from Source
A        0
B        2
C        5
D        1

Routing table for node B:
Node     Distance from Source
A        2
B        0
C        3
D        3

Routing table for node C:
Node     Distance from Source
A        5
B        3
C        0
D        6

Routing table for node D:
Node     Distance from Source
A        1
B        3
C        6
D        0


Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

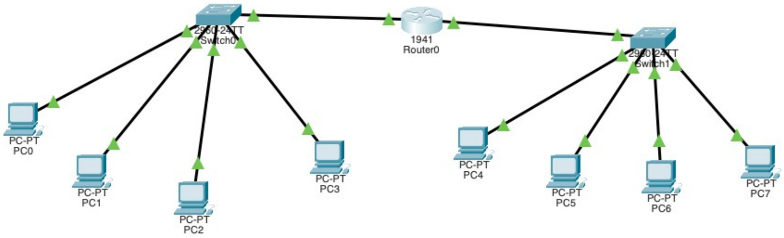
3. An ISP is granted a block of addresses starting with 150.80.0.0/16. The ISP wants to distribute these blocks to 2600 customers as follows. Implement the subnet using CISCO packet tracer for any two group of subnet. Each subnet having 4 addresses.

a. The first group has 200 medium-size businesses; each needs 128 addresses.

b. The second group has 400 small businesses; each needs 16 addresses.

c. The third group has 2000 households; each needs 4 addresses.

Solution:



For small businesses:

- Each requires 16 addresses, so we'll allocate a /28 subnet (which provides 16 addresses).
- A total of 400 such subnets are needed.

For households:

- Each requires 4 addresses, so we'll allocate a /30 subnet (which provides 4 addresses).
- A total of 2000 such subnets are required.

Small Business Subnet:

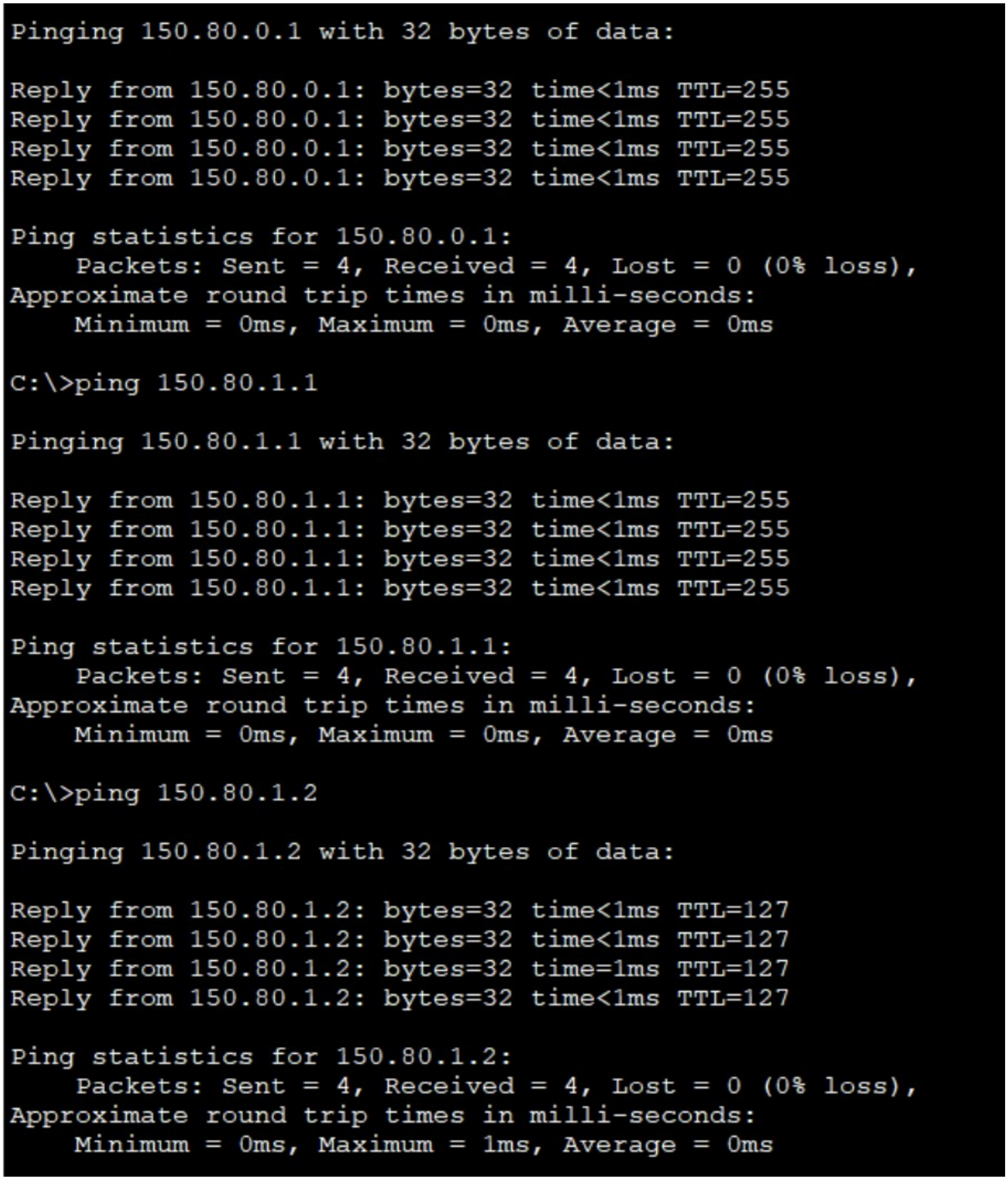
- Network: 150.80.0.0/28
- Usable IP range: 150.80.0.1 - 150.80.0.14
- Gateway: 150.80.0.1

Household Subnet:

- Network: 150.80.1.0/30
- Usable IP range: 150.80.1.1 - 150.80.1.2
- Gateway: 150.80.1.1

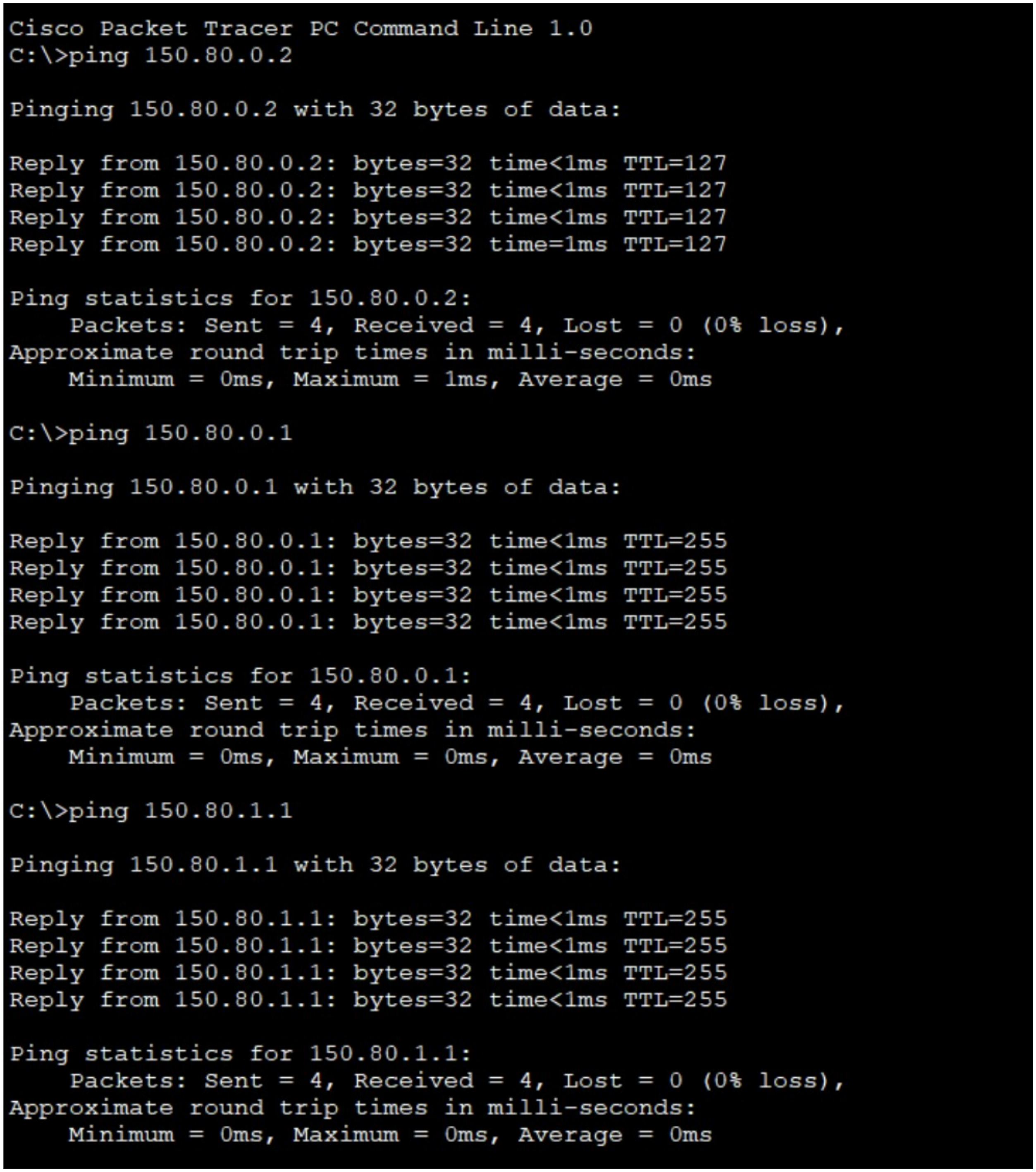
Testing Connectivity

For small business PCs:



```
Pinging 150.80.0.1 with 32 bytes of data:

Reply from 150.80.0.1: bytes=32 time<1ms TTL=255
Reply from 150.80.0.1: bytes=32 time<1ms TTL=255
Reply from 150.80.0.1: bytes=32 time<1ms TTL=255
Reply from 150.80.0.1: bytes=32 time<1ms TTL=255

Ping statistics for 150.80.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 150.80.1.1

Pinging 150.80.1.1 with 32 bytes of data:

Reply from 150.80.1.1: bytes=32 time<1ms TTL=255
Reply from 150.80.1.1: bytes=32 time<1ms TTL=255
Reply from 150.80.1.1: bytes=32 time<1ms TTL=255
Reply from 150.80.1.1: bytes=32 time<1ms TTL=255

Ping statistics for 150.80.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 150.80.1.2

Pinging 150.80.1.2 with 32 bytes of data:

Reply from 150.80.1.2: bytes=32 time<1ms TTL=127
Reply from 150.80.1.2: bytes=32 time<1ms TTL=127
Reply from 150.80.1.2: bytes=32 time=1ms TTL=127
Reply from 150.80.1.2: bytes=32 time<1ms TTL=127

Ping statistics for 150.80.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

For Household PCs



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 150.80.0.2

Pinging 150.80.0.2 with 32 bytes of data:

Reply from 150.80.0.2: bytes=32 time<1ms TTL=127
Reply from 150.80.0.2: bytes=32 time<1ms TTL=127
Reply from 150.80.0.2: bytes=32 time<1ms TTL=127
Reply from 150.80.0.2: bytes=32 time=1ms TTL=127

Ping statistics for 150.80.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 150.80.0.1

Pinging 150.80.0.1 with 32 bytes of data:

Reply from 150.80.0.1: bytes=32 time<1ms TTL=255
Reply from 150.80.0.1: bytes=32 time<1ms TTL=255
Reply from 150.80.0.1: bytes=32 time<1ms TTL=255
Reply from 150.80.0.1: bytes=32 time<1ms TTL=255

Ping statistics for 150.80.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 150.80.1.1

Pinging 150.80.1.1 with 32 bytes of data:

Reply from 150.80.1.1: bytes=32 time<1ms TTL=255
Reply from 150.80.1.1: bytes=32 time<1ms TTL=255
Reply from 150.80.1.1: bytes=32 time<1ms TTL=255
Reply from 150.80.1.1: bytes=32 time<1ms TTL=255

Ping statistics for 150.80.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```