**Server :**

```
#pragma GCC ignored -Wall
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 8080
#define PORT 1234
#define SA struct sockaddr
// Function designed for chat between client and server.
void func(int connfd)
{
	char buff[MAX];
	int n;
	// infinite loop for chat
	for (;;) {
		bzero(buff, MAX);
		// read the message from client and copy it in buffer
		read(connfd, buff, sizeof(buff));
		// print buffer which contains the client contents
		printf("From client: %s\t To client : ", buff);
		bzero(buff, MAX);
		n = 0;
		// copy server message in the buffer
		while ((buff[n++] = getchar()) != '\n');
		// and send that buffer to client
		write(connfd, buff, sizeof(buff));
		// if msg contains "Exit" then server exit and chat ended.
		if (strncmp("exit", buff, 4) == 0) {
			printf("Server Exit...\n");
```

```
                break;
            }
        }
}
// Driver function
int main()
{
        int sockfd, connfd, len;
        struct sockaddr_in servaddr, cli;
        // socket create and verification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
                printf("socket creation failed...\n");
                exit(0);
        }
        else
                printf("Socket successfully created..\n");
        bzero(&servaddr, sizeof(servaddr));
        // assign IP, PORT
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);
        // Binding newly created socket to given IP and verification
        if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
                printf("socket bind failed...\n");
                exit(0);
        }
        else
                printf("Socket successfully binded..\n");
        // Now server is ready to listen and verification
        if ((listen(sockfd, 5)) != 0) {
                printf("Listen failed...\n");
                exit(0);
```

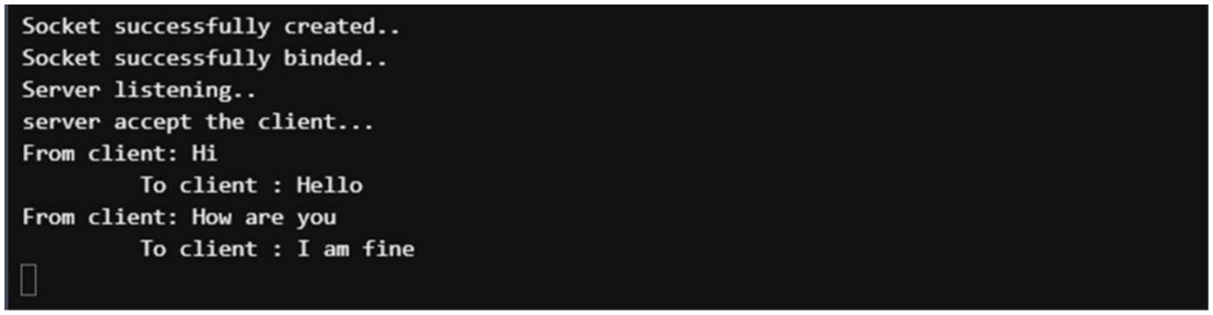
```
	}
	else
		printf("Server listening..\n");
	len = sizeof(cli);
	// Accept the data packet from client and verification
	connfd = accept(sockfd, (SA*)&cli, &len);
	if (connfd < 0) {
		printf("server accept failed...\n");
		exit(0);
	}
	else
		printf("server accept the client...\n");
	// Function for chatting between client and server
	func(connfd);
	// After chatting close the socket
	close(sockfd);
}
```

**Client:**

```
#pragma GCC ignored -Wall
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 8080
#define PORT 1234
#define SA struct sockaddr
void func(int sockfd)
{
   char buff[MAX];
   int n;
```

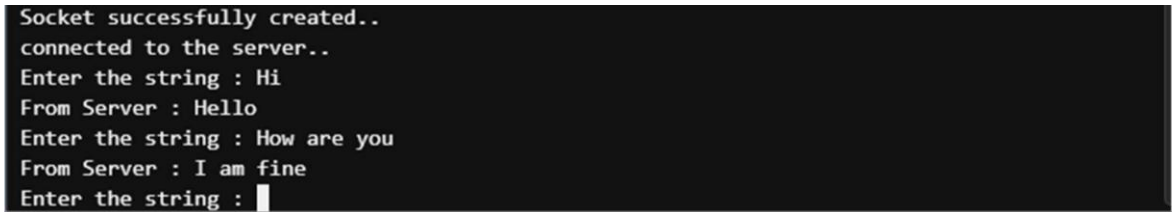
```
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
}
int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;
    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));
    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    // connect the client socket to server socket
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
        printf("connection with the server failed...\n");
        exit(0);
    }
    else
        printf("connected to the server..\n");
    // function for chat
    func(sockfd);
    // close the socket
    close(sockfd);
}
```

**Server Output :**



```
Socket successfully created..
Socket successfully binded..
Server listening..
server accept the client...
From client: Hi
         To client : Hello
From client: How are you
         To client : I am fine
```

**Client Output :**



```
Socket successfully created..
connected to the server..
Enter the string : Hi
From Server : Hello
Enter the string : How are you
From Server : I am fine
Enter the string :
```

**Result:**

**Server:**

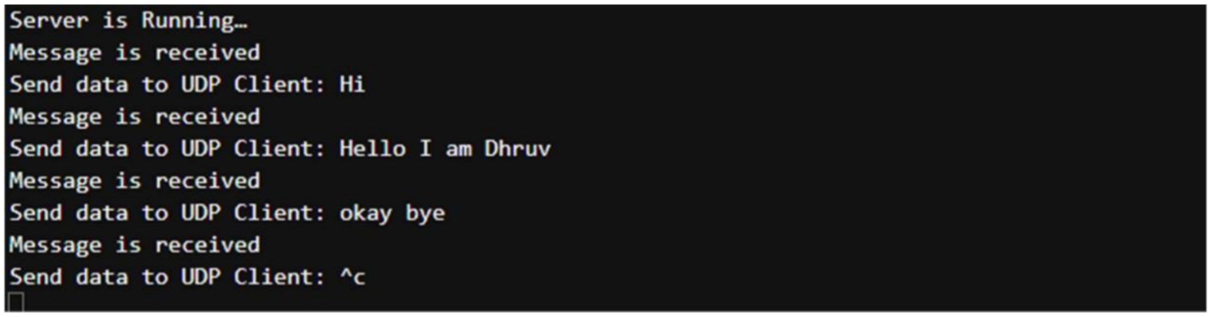
```
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<netinet/in.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<sys/types.h>
int main(int argc,char *argv[])
{
int sd;
char buff[1024];
struct sockaddr_in cliaddr,servaddr;
socklen_t clilen;
clilen=sizeof(cliaddr);
/*UDP socket is created, an Internet socket address structure is filled with wildcard
address & server's well known port*/
sd=socket(AF_INET,SOCK_DGRAM,0);
if (sd<0)
{
perror ("Cannot open Socket");
exit(1);
}
bzero(&servaddr,sizeof(servaddr));
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(5669);
 /*Bind function assigns a local protocol address to the socket*/
if(bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
perror("error in binding the port");
exit(1);
}
printf("%s","Server is Running…\n");
while(1)
{
bzero(&buff,sizeof(buff));
 /*Read the message from the client*/
if(recvfrom(sd,buff,sizeof(buff),0,(struct sockaddr*)&cliaddr,&clilen)<0)
{
perror("Cannot rec data");
exit(1);
 }
 printf("Message is received \n",buff);
/*Sendto function is used to echo the message from server to client side*/
 if(sendto(sd,buff,sizeof(buff),0,(struct sockadddr*)&cliaddr,clilen)<0)
 {
perror("Cannot send data to client");
exit(1);
 }
 printf("Send data to UDP Client: %s",buff);
}
cloSe(sd);
return 0;
}
```

**Client :**

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
```

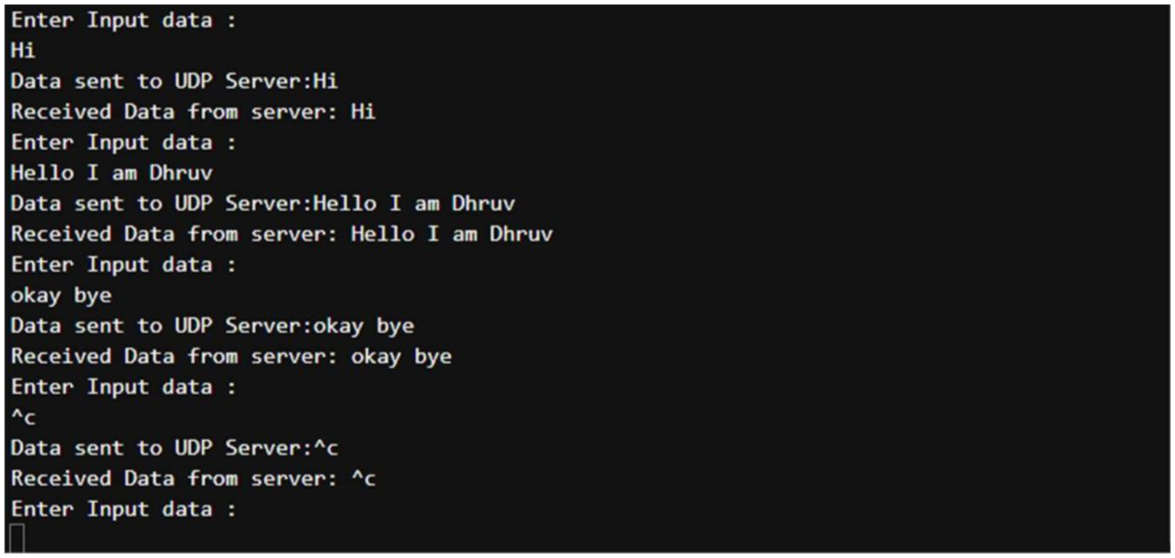
```
#include<unistd.h>
#include<string.h>
#include<netinet/in.h>
#include<netdb.h>
int main(int argc,char*argv[])
{
int sd;
char buff[1024];
struct sockaddr_in servaddr;
socklen_t len;
len=sizeof(servaddr);
 /*UDP socket is created, an Internet socket address structure is filled with
 wildcard address & server’s well known port*/
sd = socket(AF_INET,SOCK_DGRAM,0);
if(sd<0)
{
perror("Cannot open socket");
exit(1);
}
bzero(&servaddr,len);
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(5669);
while(1)
{
printf("Enter Input data : \n");
bzero(buff,sizeof(buff));
 /*Reads the message from standard input*/
fgets(buff,sizeof (buff),stdin);
 /*sendto is used to transmit the request message to the server*/
if(sendto (sd,buff,sizeof (buff),0,(struct sockaddr*)&servaddr,len)<0)
 {
 perror("Cannot send data");
 exit(1);
 }
printf("Data sent to UDP Server:%s",buff);
bzero(buff,sizeof(buff));
/*Receiving the echoed message from server*/
if(recvfrom (sd,buff,sizeof(buff),0,(struct sockaddr*)&servaddr,&len)<0)
 {
 perror("Cannot receive data");
 exit(1);
 }
printf("Received Data from server: %s",buff);
 }
close(sd);
return 0;
}
```

**Server Output:**



```
Server is Running…
Message is received
Send data to UDP Client: Hi
Message is received
Send data to UDP Client: Hello I am Dhruv
Message is received
Send data to UDP Client: okay bye
Message is received
Send data to UDP Client: ^c
```

**Client Output:**



```
Enter Input data :
Hi
Data sent to UDP Server:Hi
Received Data from server: Hi
Enter Input data :
Hello I am Dhruv
Data sent to UDP Server:Hello I am Dhruv
Received Data from server: Hello I am Dhruv
Enter Input data :
okay bye
Data sent to UDP Server:okay bye
Received Data from server: okay bye
Enter Input data :
^c
Data sent to UDP Server:^c
Received Data from server: ^c
Enter Input data :
```

**Server :**

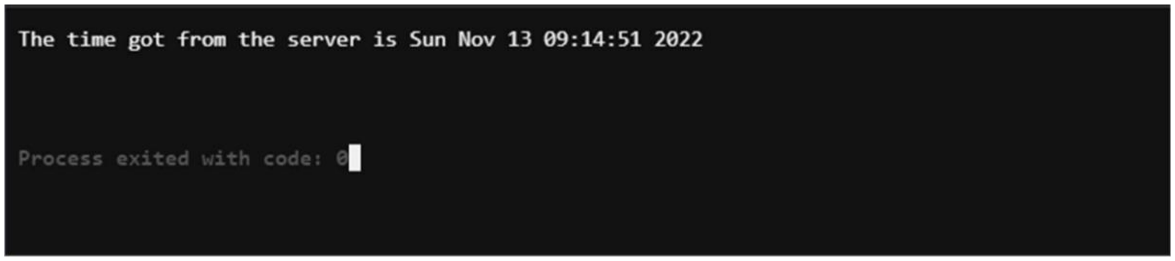
```
# server.py
import socket
import time
# create a socket object
serversocket = socket.socket(
            socket.AF_INET, socket.SOCK_STREAM)
# get local machine name
host = socket.gethostname()
port = 9999
# bind to the port
serversocket.bind((host, port))
# queue up to 5 requests
serversocket.listen(5)
while True:
  # establish a connection
  clientsocket,addr = serversocket.accept()
  print("Got a connection from %s" % str(addr))
  currentTime = time.ctime(time.time()) + "\r\n"
  clientsocket.send(currentTime.encode('ascii'))
  clientsocket.close()
```

**Client :**

```
# client.py
import socket
# create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# get local machine name
host = socket.gethostname()
port = 9999
```

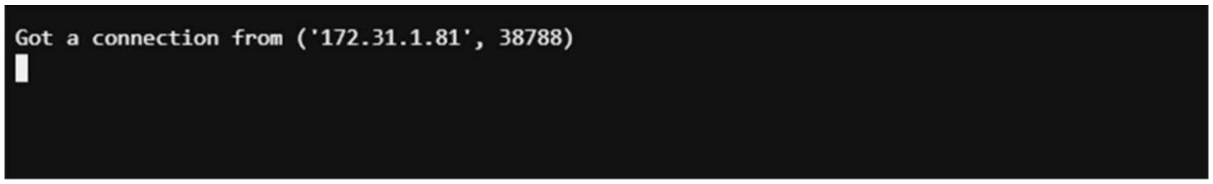
```
# connection to hostname on the port.
s.connect((host, port))
# Receive no more than 1024 bytes
tm = s.recv(1024)
s.close()
print("The time got from the server is %s" % tm.decode('ascii'))
```

**Server Output:**



```
The time got from the server is Sun Nov 13 09:14:51 2022


Process exited with code: 0
```

**Client Output :**



```
Got a connection from ('172.31.1.81', 38788)
```

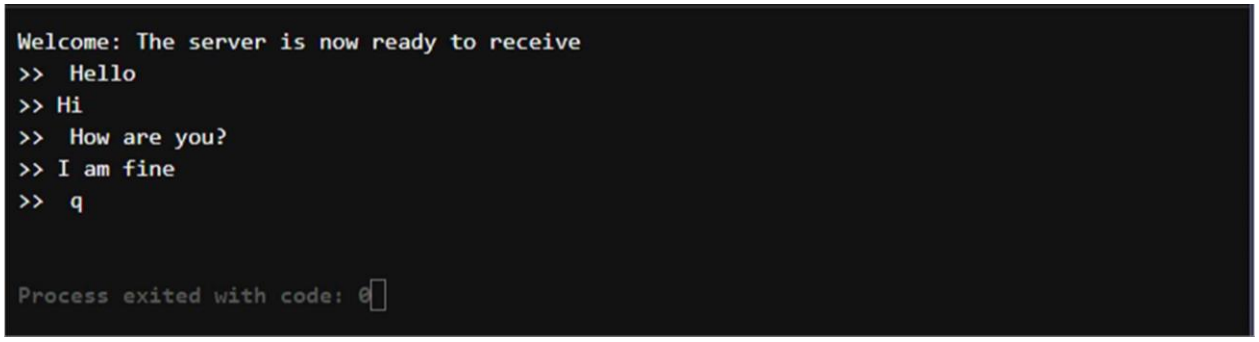
**Server :**

```
# Server Side Script
# Supports Python v3.*
from socket import *
server_port = 8008
server_socket = socket(AF_INET,SOCK_STREAM)
server_socket.bind(('',server_port))
server_socket.listen(1)
print ("Welcome: The server is now ready to receive")
connection_socket, address = server_socket.accept()
while True:
  sentence = connection_socket.recv(2048).decode()
  print('>> ',sentence)
  if(sentence == 'q'):
    break
  message = input(">> ")
connection_socket.send(message.encode())
connection_socket.close()
```

**Client :**

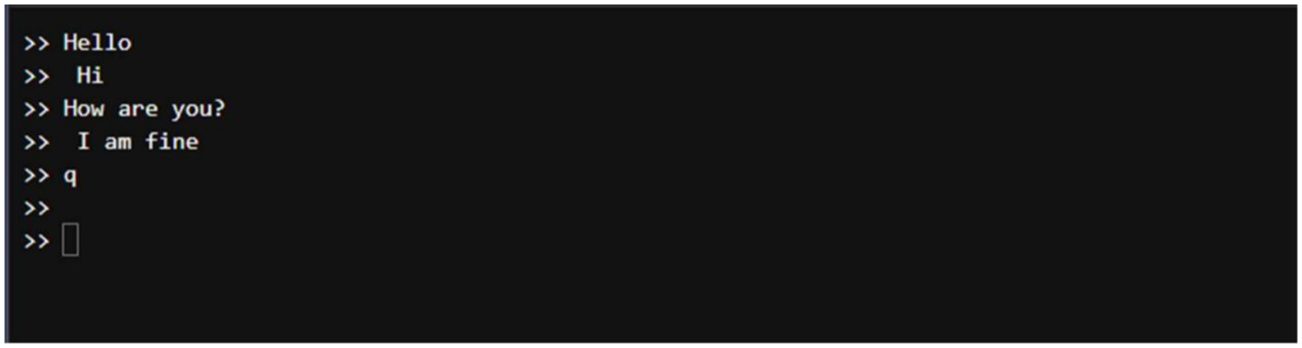
```
# Server Side Script
# Supports Python v3.*
from socket import *
server_port = 8008
server_socket = socket(AF_INET,SOCK_STREAM)
server_socket.bind(('',server_port))
server_socket.listen(1)
print ("Welcome: The server is now ready to receive")
connection_socket, address = server_socket.accept()
while True:
  sentence = connection_socket.recv(2048).decode()
  print('>> ',sentence)
  if(sentence == 'q'):
    break
  message = input(">> ")
connection_socket.send(message.encode())
connection_socket.close()
```

**Server Output:**



```
Welcome: The server is now ready to receive
>>  Hello
>> Hi
>>  How are you?
>> I am fine
>>  q


Process exited with code: 0
```

**Client Output:**



```
>> Hello
>>  Hi
>> How are you?
>>  I am fine
>> q
>>
>>
```

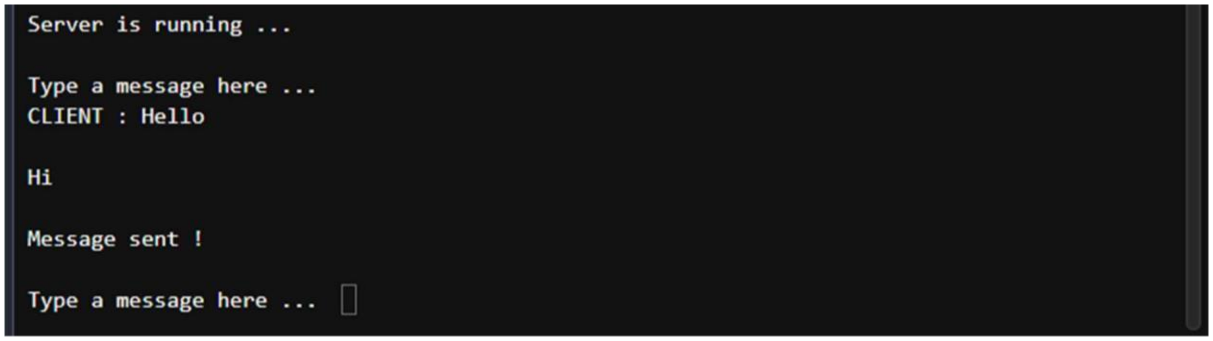
**Server:**

```
#include<sys/types.h>

#include<sys/socket.h>

#include<stdio.h>

#include<unistd.h>

#include<netdb.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<string.h>

int main(int argc,char *argv[])

{

int clientSocketDescriptor,socketDescriptor;

struct sockaddr_in serverAddress,clientAddress;

socklen_t clientLength;

char recvBuffer[1000],sendBuffer[1000];

pid_t cpid;

bzero(&serverAddress,sizeof(serverAddress));

serverAddress.sin_family=AF_INET;

serverAddress.sin_addr.s_addr=htonl(INADDR_ANY);

serverAddress.sin_port=htons(5500);

socketDescriptor=socket(AF_INET,SOCK_STREAM,0);

bind(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));

listen(socketDescriptor,5);

printf("%s\n","Server is running ...");

clientSocketDescriptor=accept(socketDescriptor,(struct sockaddr*)&clientAddress,&clientLength);

cpid=fork();

if(cpid==0){

while(1){

bzero(&recvBuffer,sizeof(recvBuffer));

recv(clientSocketDescriptor,recvBuffer,sizeof(recvBuffer),0);

printf("\nCLIENT : %s\n",recvBuffer);
}
}
else{
while(1){
bzero(&sendBuffer,sizeof(sendBuffer));
printf("\nType a message here ... ");
fgets(sendBuffer,10000,stdin);
send(clientSocketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
printf("\nMessage sent !\n");
}
}
return 0;}
```

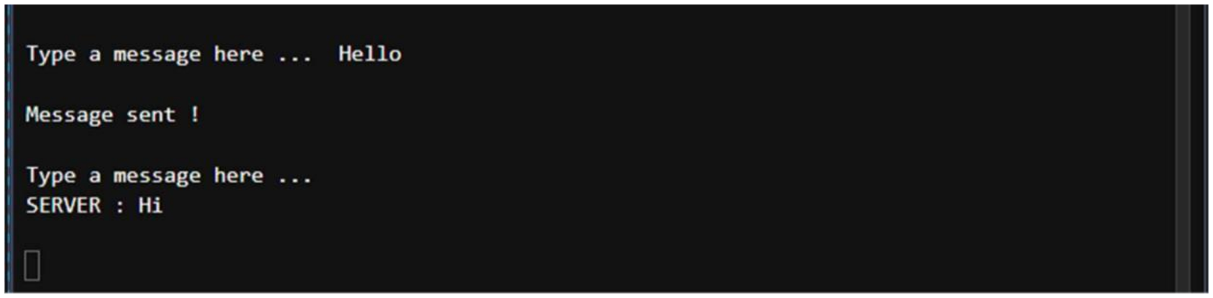
**Client :**

```
#include  "stdio.h"
#include  "stdlib.h"
#include "string.h"
//headers for socket and related functions
#include <sys/types.h>
#include <sys/socket.h>
//for including structures which will store information needed
#include <netinet/in.h>
#include <unistd.h>
//for gethostbyname
#include "netdb.h"
#include "arpa/inet.h"
int main()
{
int socketDescriptor;
struct sockaddr_in serverAddress;
char sendBuffer[1000],recvBuffer[1000];
pid_t cpid;
bzero(&serverAddress,sizeof(serverAddress));
serverAddress.sin_family=AF_INET;
serverAddress.sin_addr.s_addr=inet_addr("127.0.0.1");
serverAddress.sin_port=htons(5500);
/*Creating a socket, assigning IP address and port number for that socket*/
socketDescriptor=socket(AF_INET,SOCK_STREAM,0);
/*Connect establishes connection with the server using server IP address*/
connect(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));
/*Fork is used to create a new process*/
cpid=fork();
if(cpid==0){
while(1){
bzero(&sendBuffer,sizeof(sendBuffer));
printf("\nType a message here ...  ");
/*This function is used to read from server*/
fgets(sendBuffer,10000,stdin);
/*Send the message to server*/
send(socketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
printf("\nMessage sent !\n");
}
}
else{
while(1){
bzero(&recvBuffer,sizeof(recvBuffer));
/*Receive the message from server*/
recv(socketDescriptor,recvBuffer,sizeof(recvBuffer),0);
printf("\nSERVER : %s\n",recvBuffer);
}
}
return 0;}
```

**Server Output:**


```
Server is running ...

Type a message here ...
CLIENT : Hello

Hi

Message sent !

Type a message here ...
```

**Client Output:**


```

Type a message here ...  Hello

Message sent !

Type a message here ...
SERVER : Hi

```

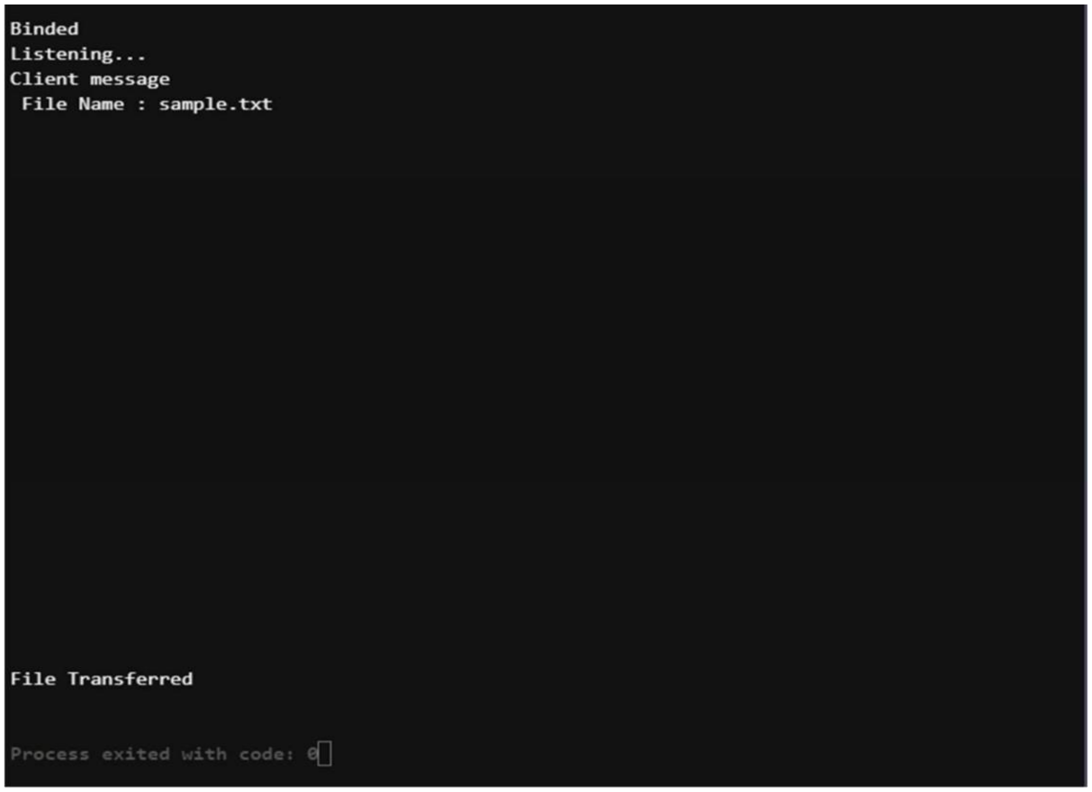
**Server:**

```
#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv)
{
    int sockfd, newsockfd, clength;
    struct sockaddr_in serv_addr,cli_addr;
    char t[MAX], str[MAX];
    strcpy(t,"exit");
    sockfd=socket(AF_INET, SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    printf("\nBinded");
    bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
    printf("\nListening...");
    listen(sockfd, 5);
    clength=sizeof(cli_addr);
    newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clength);

    close(sockfd);
    read(newsockfd, &str, MAX);
    printf("\nClient message\n File Name : %s\n", str);
    f1=fopen(str, "r");
    while(fgets(buff, 4096, f1)!=NULL){
        write(newsockfd, buff,MAX);
        printf("\n");
    }
    fclose(f1);
    printf("\nFile Transferred\n");
    return 0;
}
```

**Client :**

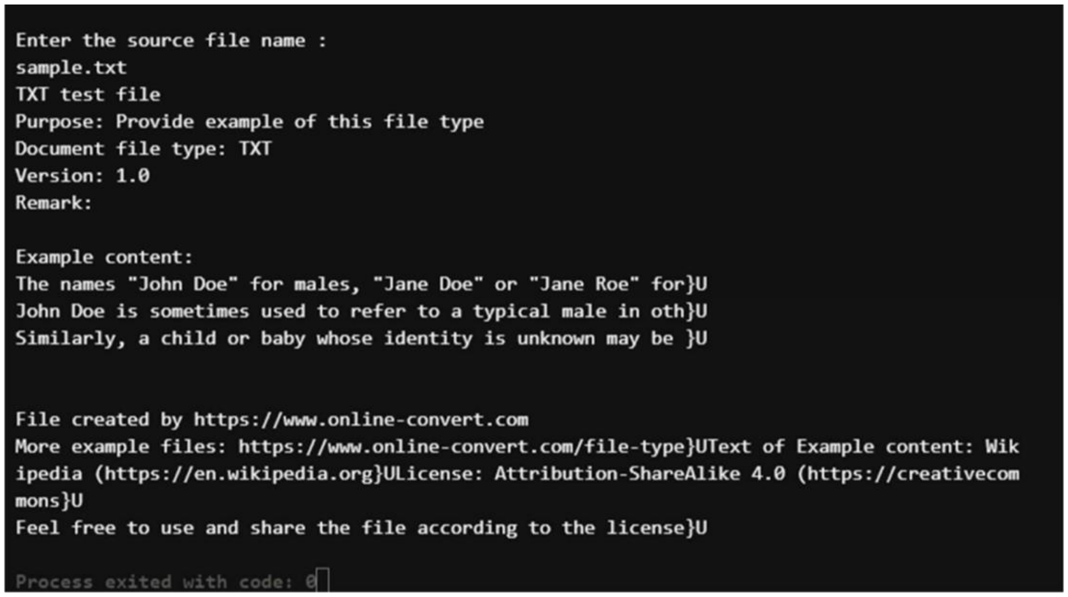
```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int main(int arg,char*argv[])
{
    int sockfd,n;
    struct sockaddr_in serv_addr;
    struct hostent*server;
    char send[MAX],recvline[MAX],s[MAX],name[MAX];
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;

    serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
    printf("\nEnter the source file name : \n");
    scanf("%s",send);
    write(sockfd,send,MAX);
    while((n=read(sockfd,recvline,MAX))!=0){
      printf("%s",recvline);
    }
    close(sockfd);
    return 0;}
```

**Server Output:**



```
Binded
Listening...
Client message
 File Name : sample.txt









File Transferred

Process exited with code: 0
```

**Client Output:**



```
Enter the source file name :
sample.txt
TXT test file
Purpose: Provide example of this file type
Document file type: TXT
Version: 1.0
Remark:

Example content:
The names "John Doe" for males, "Jane Doe" or "Jane Roe" for}U
John Doe is sometimes used to refer to a typical male in oth}U
Similarly, a child or baby whose identity is unknown may be }U


File created by https://www.online-convert.com
More example files: https://www.online-convert.com/file-type}UText of Example content: Wik
ipedia (https://en.wikipedia.org}ULicense: Attribution-ShareAlike 4.0 (https://creativecom
mons}U
Feel free to use and share the file according to the license}U

Process exited with code: 0
```

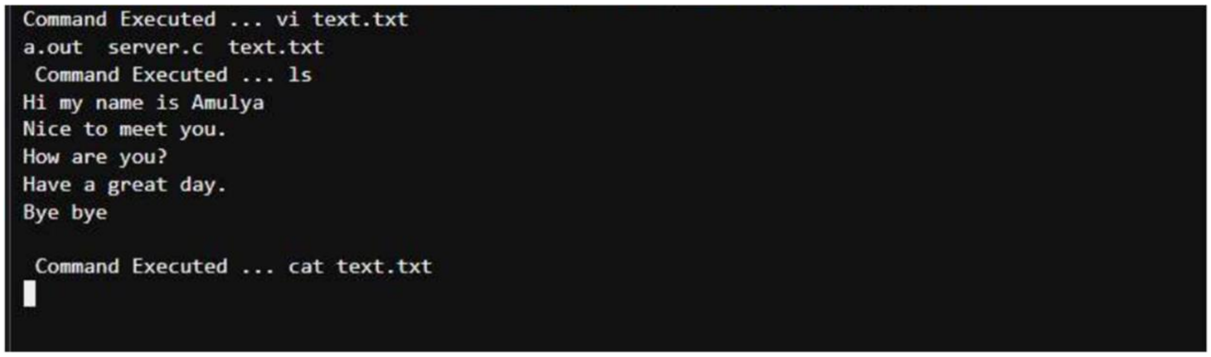
**Server :**

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <string.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <unistd.h>
#define MAX 1000
int main()
{
 int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
 int size;
 char buffer[MAX], message[] = "Command Successfully executed !";
 struct sockaddr_in clientAddress, serverAddress;
 socklen_t clientLength = sizeof(clientAddress);
 bzero(&serverAddress, sizeof(serverAddress));
 serverAddress.sin_family = AF_INET;
 serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
 serverAddress.sin_port = htons(8079);
 bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
 while (1)
 {
 bzero(buffer, sizeof(buffer));
recvfrom(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr *)&clientAddress,
&clientLength);
 system(buffer);
 printf("Command Executed ... %s ", buffer);
sendto(serverDescriptor, message, sizeof(message), 0, (struct sockaddr *)&clientAddress,
clientLength);
 }
 close(serverDescriptor);
 return 0;
}
```

**Client :**

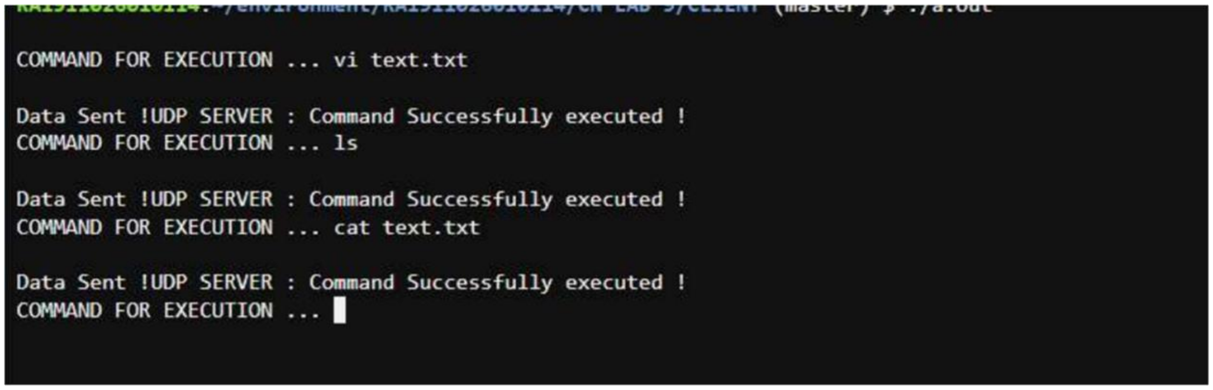
```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <unistd.h>
#include <netdb.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
#define MAX 1000
int main()
{
 int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
 char buffer[MAX], message[MAX];
 struct sockaddr_in cliaddr, serverAddress;
 socklen_t serverLength = sizeof(serverAddress);
 bzero(&serverAddress, sizeof(serverAddress));
 serverAddress.sin_family = AF_INET;
 serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1");
 serverAddress.sin_port = htons(8079);
bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
 while (1)
 {
 printf("\nCOMMAND FOR EXECUTION ... ");
 fgets(buffer, sizeof(buffer), stdin);
 sendto(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr *)&serverAddress, serverLength);
 printf("\nData Sent !");
 recvfrom(serverDescriptor, message, sizeof(message), 0, (struct sockaddr
 *)&serverAddress, &serverLength);
 printf("UDP SERVER : %s", message);
 }
 return 0;
}
```

**Server Output:**



```
Command Executed ... vi text.txt
a.out  server.c  text.txt
 Command Executed ... ls
Hi my name is Amulya
Nice to meet you.
How are you?
Have a great day.
Bye bye

 Command Executed ... cat text.txt
```

**Client Output:**

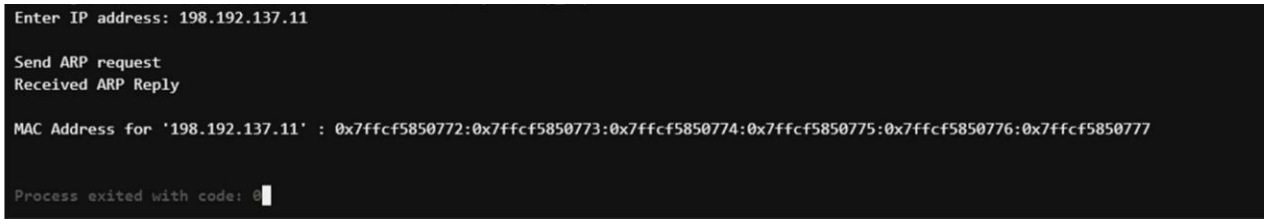


```
RA1911028010114:~/environment/RA1911028010114/CN LAB 9/CLIENT (master) $ ./a.out

COMMAND FOR EXECUTION ... vi text.txt

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ... ls

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ... cat text.txt

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ...
```

**Code :**

```
#include<sys/types.h>
#include<sys/socket.h>
#include<net/if_arp.h>
#include<sys/ioctl.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<math.h>
#include<complex.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<stdlib.h>
int main()
{
struct sockaddr_in sin={0};
struct arpreq myarp={{0}};
unsigned char *ptr;
int sd;
sin.sin_family=AF_INET;
printf("Enter IP address: ");
char ip[20];
scanf("%s", ip);
if(inet_pton(AF_INET,ip,&sin.sin_addr)==0){
printf("IP address Entered '%s' is not valid \n",ip);
exit(0);
}
memcpy(&myarp.arp_pa,&sin,sizeof(myarp.arp_pa));
strcpy(myarp.arp_dev,"echo");
sd=socket(AF_INET,SOCK_DGRAM,0);
printf("\nSend ARP request\n");
if(ioctl(sd,SIOCGARP,&myarp)==1){
printf("No Entry in ARP cache for '%s'\n",ip);
exit(0);
}
ptr=&myarp.arp_pa.sa_data[0];
printf("Received ARP Reply\n");
printf("\nMAC Address for '%s' : ",ip);
printf("%p:%p:%p:%p:%p:%p\n",ptr,(ptr+1),(ptr+2),(ptr+3),(ptr+4),(ptr+5));
return 0;
}
```

**Output:**

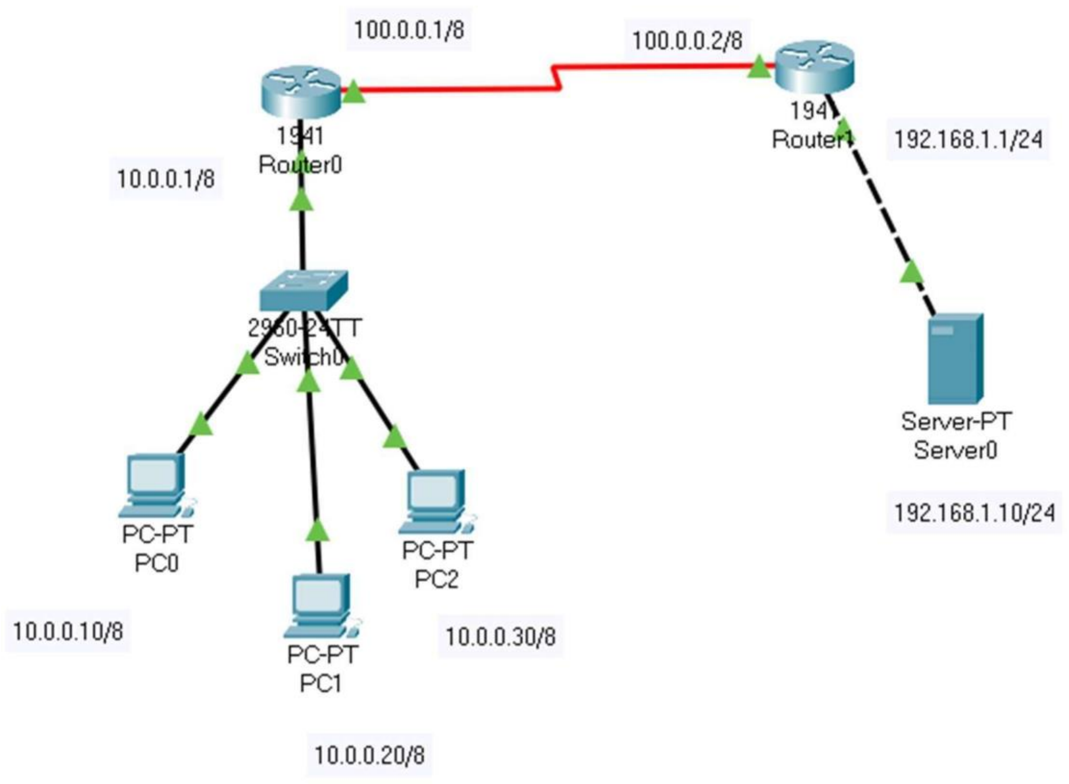


```
Enter IP address: 198.192.137.11

Send ARP request
Received ARP Reply

MAC Address for '198.192.137.11' : 0x7ffcf5850772:0x7ffcf5850773:0x7ffcf5850774:0x7ffcf5850775:0x7ffcf5850776:0x7ffcf5850777


Process exited with code: 0
```

|  |  |
|---|---|
| **Ex.No:** |  |
| **Date:** |  |

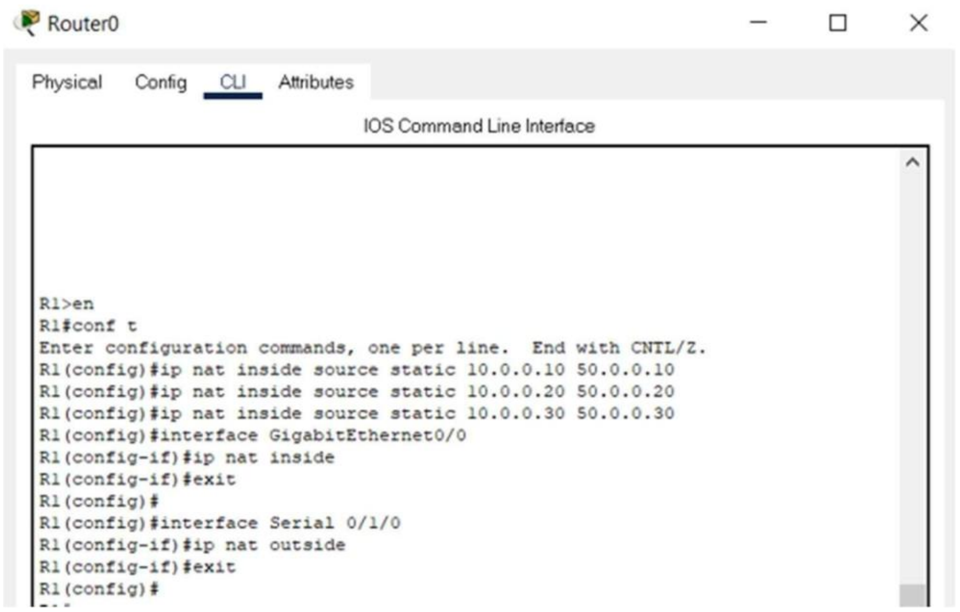
**Aim**:

**Procedure**:

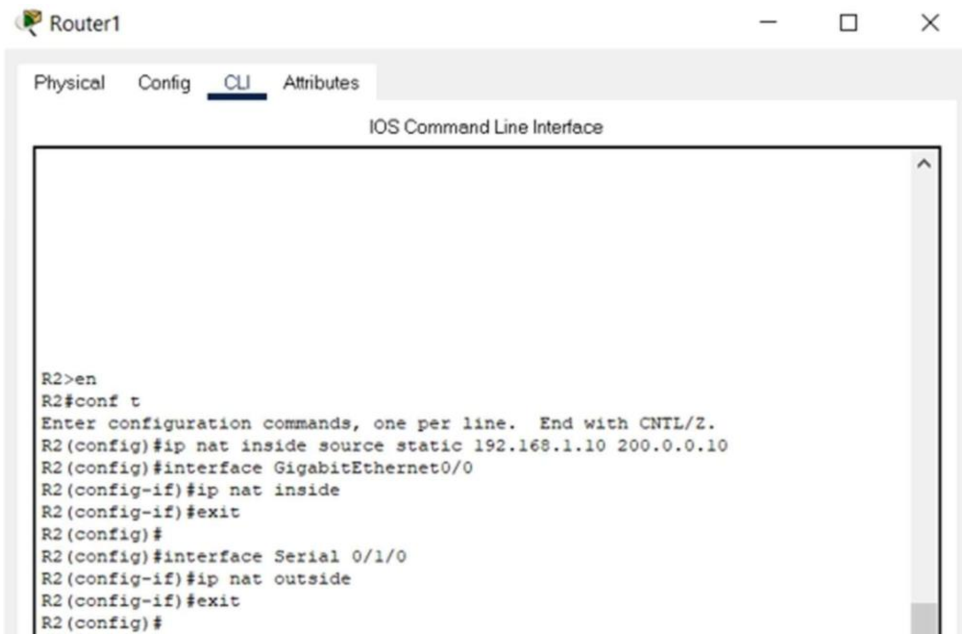


```
Router(config)#ip nat inside source static [inside local ip address] [inside global IP address]
```

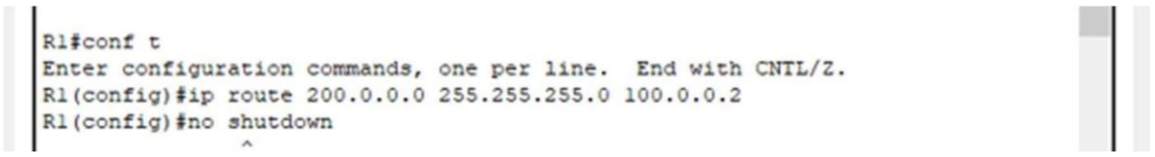
```
Router(config-if)#ip nat inside
Router(config-if)#ip nat outside
```



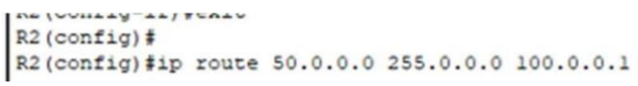
```
R1>en
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip nat inside source static 10.0.0.10 50.0.0.10
R1(config)#ip nat inside source static 10.0.0.20 50.0.0.20
R1(config)#ip nat inside source static 10.0.0.30 50.0.0.30
R1(config)#interface GigabitEthernet0/0
R1(config-if)#ip nat inside
R1(config-if)#exit
R1(config)#
R1(config)#interface Serial 0/1/0
R1(config-if)#ip nat outside
R1(config-if)#exit
R1(config)#
```



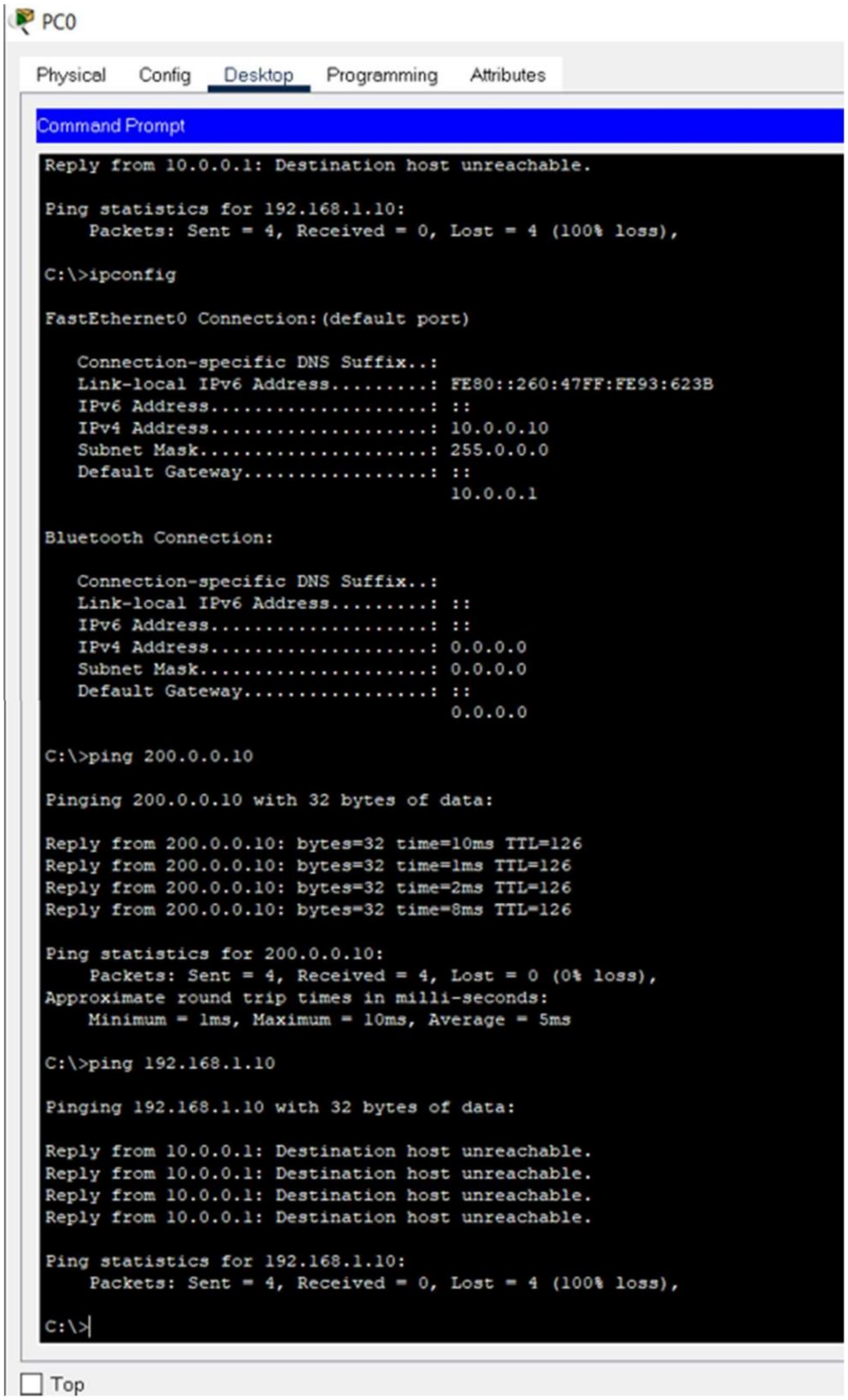
```
R2>en
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#ip nat inside source static 192.168.1.10 200.0.0.10
R2(config)#interface GigabitEthernet0/0
R2(config-if)#ip nat inside
R2(config-if)#exit
R2(config)#
R2(config)#interface Serial 0/1/0
R2(config-if)#ip nat outside
R2(config-if)#exit
R2(config)#
```



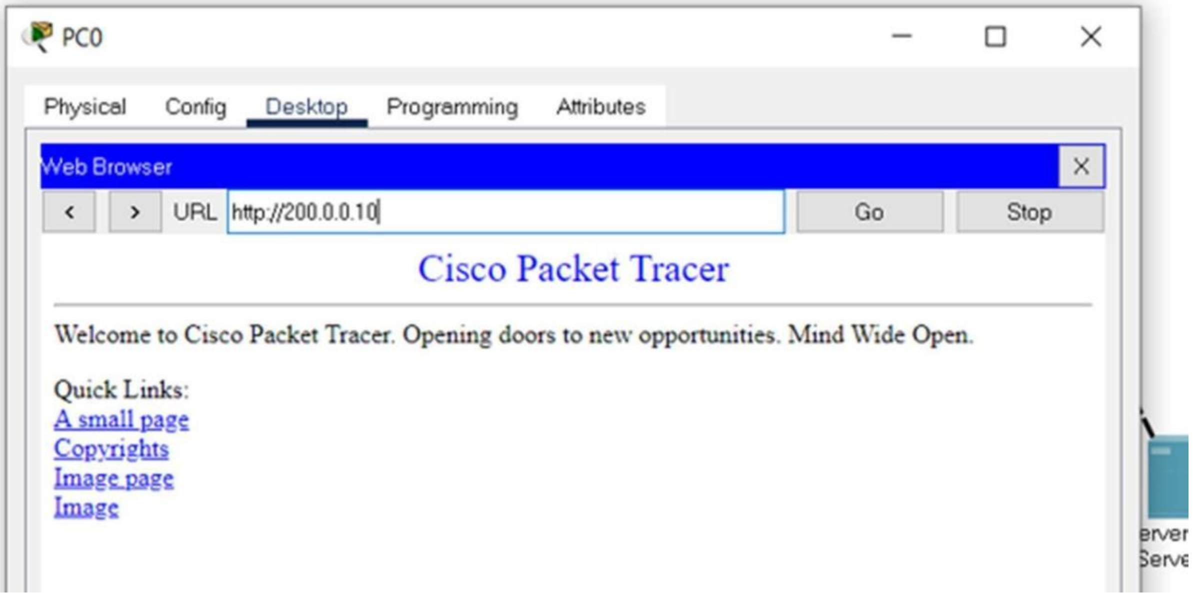
```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip route 200.0.0.0 255.255.255.0 100.0.0.2
R1(config)#no shutdown
               ^
```



```
R2(config-if)#exit
R2(config)#
R2(config)#ip route 50.0.0.0 255.0.0.0 100.0.0.1
```



```
Reply from 10.0.0.1: Destination host unreachable.

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ipconfig

FastEthernet0 Connection:(default port)

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........: FE80::260:47FF:FE93:623B
   IPv6 Address....................: ::
   IPv4 Address....................: 10.0.0.10
   Subnet Mask.....................: 255.0.0.0
   Default Gateway.................: ::
                                     10.0.0.1

Bluetooth Connection:

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........: ::
   IPv6 Address....................: ::
   IPv4 Address....................: 0.0.0.0
   Subnet Mask.....................: 0.0.0.0
   Default Gateway.................: ::
                                     0.0.0.0

C:\>ping 200.0.0.10

Pinging 200.0.0.10 with 32 bytes of data:

Reply from 200.0.0.10: bytes=32 time=10ms TTL=126
Reply from 200.0.0.10: bytes=32 time=1ms TTL=126
Reply from 200.0.0.10: bytes=32 time=2ms TTL=126
Reply from 200.0.0.10: bytes=32 time=8ms TTL=126

Ping statistics for 200.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 10ms, Average = 5ms

C:\>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:

Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

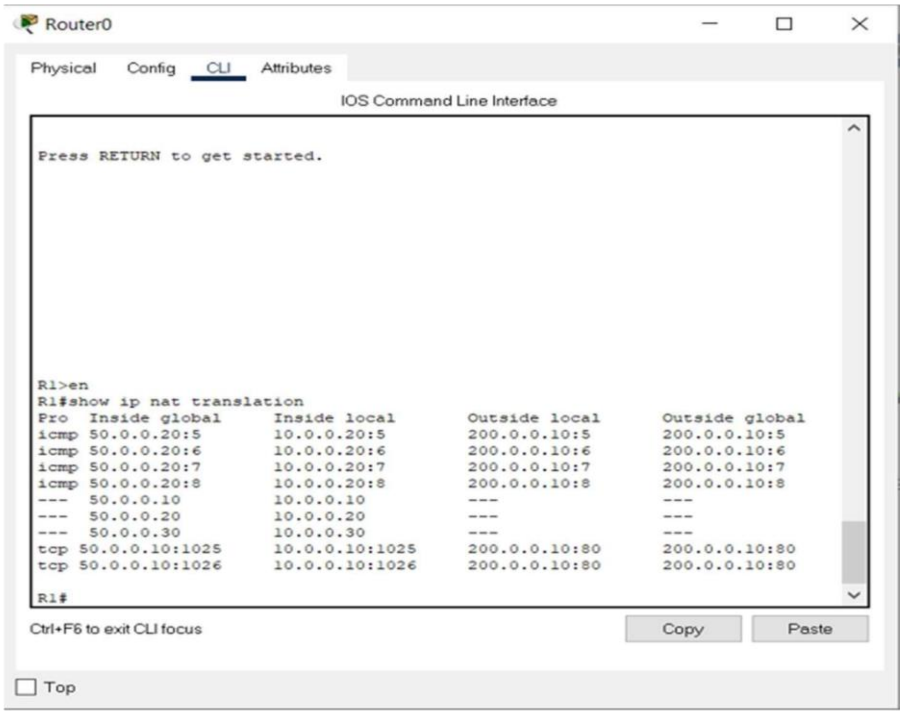


http://200.0.0.10

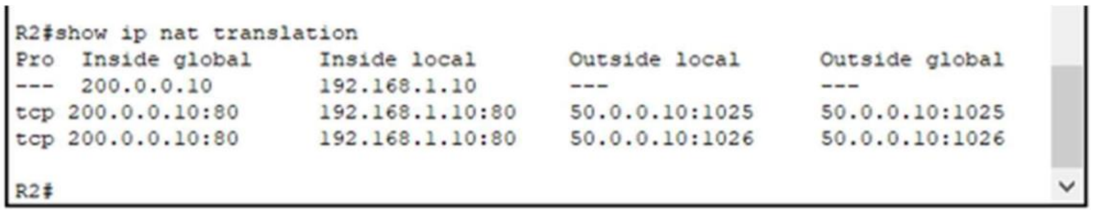
Cisco Packet Tracer

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

Quick Links:
A small page
Copyrights
Image page
Image



```
Press RETURN to get started.











R1>en
R1#show ip nat translation
Pro  Inside global     Inside local       Outside local      Outside global
icmp 50.0.0.20:5       10.0.0.20:5        200.0.0.10:5       200.0.0.10:5
icmp 50.0.0.20:6       10.0.0.20:6        200.0.0.10:6       200.0.0.10:6
icmp 50.0.0.20:7       10.0.0.20:7        200.0.0.10:7       200.0.0.10:7
icmp 50.0.0.20:8       10.0.0.20:8        200.0.0.10:8       200.0.0.10:8
---  50.0.0.10         10.0.0.10          ---                ---
---  50.0.0.20         10.0.0.20          ---                ---
---  50.0.0.30         10.0.0.30          ---                ---
tcp 50.0.0.10:1025     10.0.0.10:1025     200.0.0.10:80      200.0.0.10:80
tcp 50.0.0.10:1026     10.0.0.10:1026     200.0.0.10:80      200.0.0.10:80

R1#
```



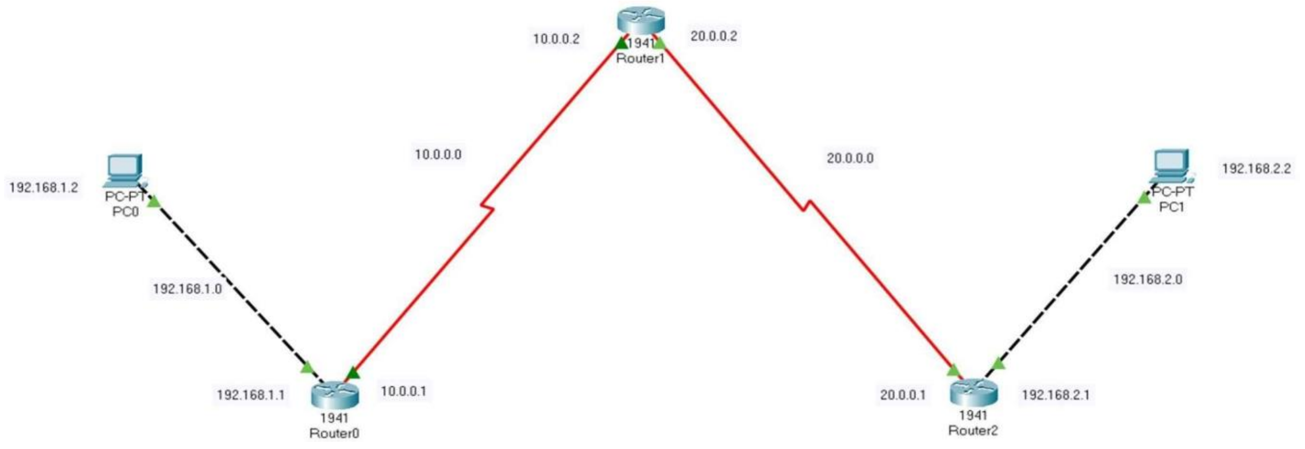
```
R2#show ip nat translation
Pro  Inside global     Inside local       Outside local      Outside global
---  200.0.0.10        192.168.1.10       ---                ---
tcp 200.0.0.10:80      192.168.1.10:80    50.0.0.10:1025     50.0.0.10:1025
tcp 200.0.0.10:80      192.168.1.10:80    50.0.0.10:1026     50.0.0.10:1026

R2#
```

**Result:**

| Ex.No: | |
|---|---|
| Date: | |

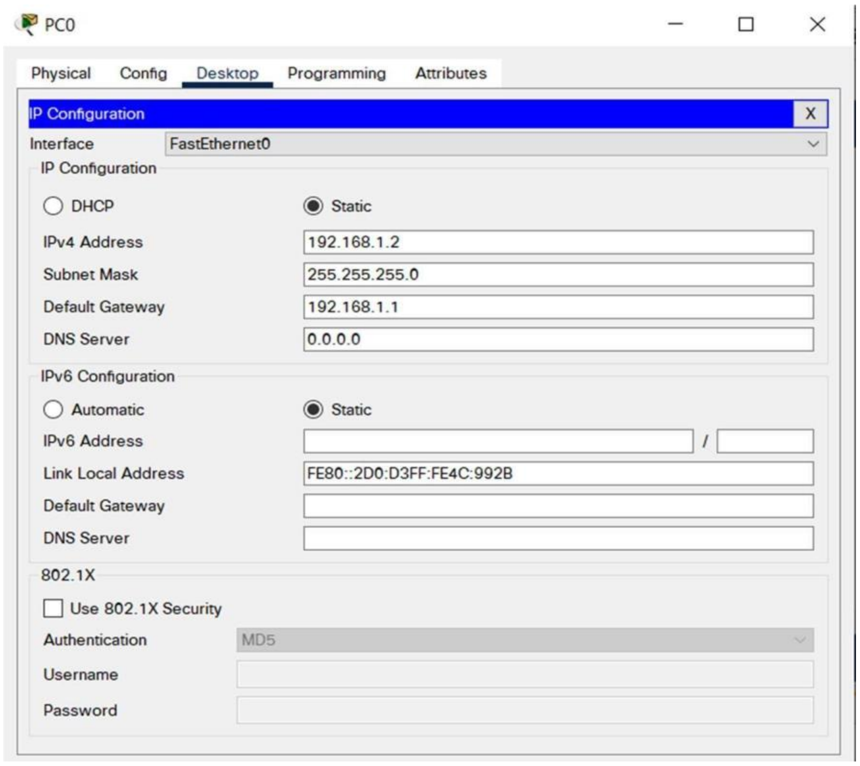
**AIM :**

**PROCEDURE :**



.

| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |



Interface: FastEthernet0

IP Configuration

DHCP [✓ Static]

| | |
|---|---|
| IPv4 Address | 192.168.1.2 |
| Subnet Mask | 255.255.255.0 |
| Default Gateway | 192.168.1.1 |
| DNS Server | 0.0.0.0 |

IPv6 Configuration

Automatic [✓ Static]

| | |
|---|---|
| IPv6 Address | / |
| Link Local Address | FE80::2D0:D3FF:FE4C:992B |
| Default Gateway | |
| DNS Server | |

802.1X

☐ Use 802.1X Security

| | |
|---|---|
| Authentication | MD5 |
| Username | |
| Password | |

```
Router>enable

Router#config t

Router(config)#int gig0/0


Router(config-if)#ip add 192.168.1.1 255.255.255.0

Router(config-if)#no shut


Router(config-if)#exit

Router(config)#int

se0/1/0
```

```
Router>enable

Router#config t

Router(config)#int se0/1/0

Router(config-if)#ip add 10.0.0.2 255.0.0.0

Router(config-if)#no shut

Router(config-if)#exit

Router(config)#int se0/1/1

Router(config-if)#ip add 20.0.0.1 255.0.0.0

Router(config-if)#no shut
```

CONFIGURATION ON ROUTER3:

```
Router>enable

Router#config t

Router(config)#int se0/1/0

Router(config-if)#ip add 20.0.0.2 255.0.0.0

Router(config-if)#no shut

Router(config-if)#exit

Router(config)#int gig0/0

Router(config-if)#ip add 192.168.2.1 255.255.255.0

Router(config-if)#no shut
```

```
Router>enable

Router#config t

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.2

Router(config)#
```

```
Router>enable

Router#config t

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#ip route 0.0.0.0 0.0.0.0 20.0.0.1

Router(config)#
```

```
Router#ping 20.0.0.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 20.0.0.2, timeout is 2 seconds:
```

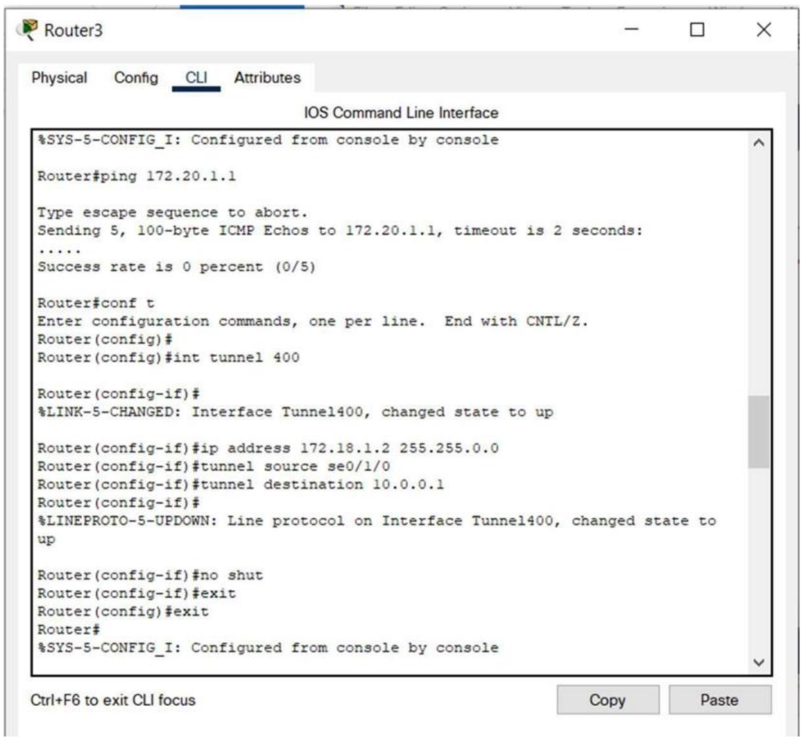
Now we go to Router3 and test the network by pinging Router1 interface.

```
Router#ping 10.0.0.1

Type escape sequence to abort.
```

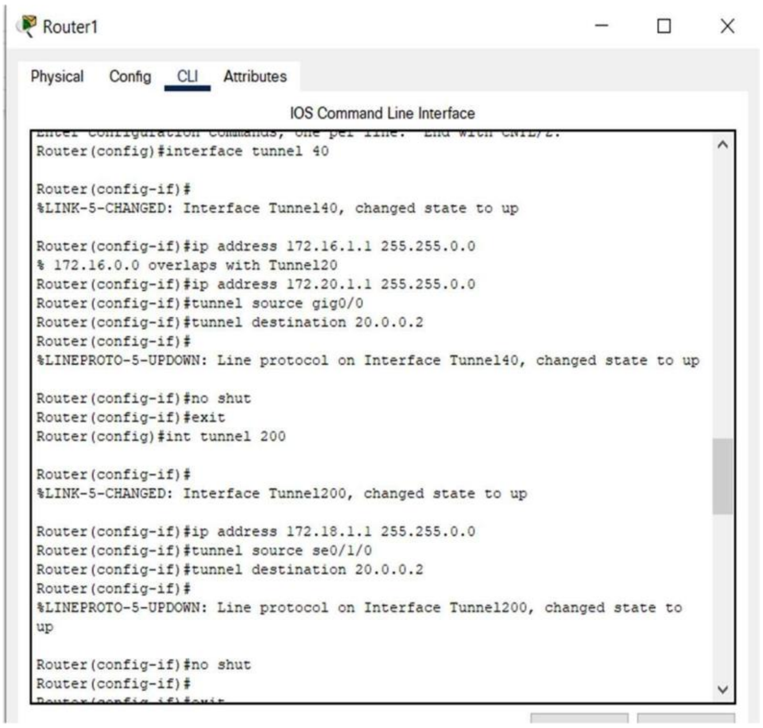
```
Success rate is 100 percent (5/5), round-trip min/avg/max = 25/28/32 ms
```

```
Router#config t

Router(config)#interface tunnel

200

Router(config-if)#ip address 172.18.1.1 255.255.0.0

Router(config-if)#tunnel source se0/1/0

Router(config-if)#tunnel destination 20.0.0.2
```

```
Router#config t

Router(config)#interface tunnel

400

Router(config-if)#ip address 172.18.1.2 255.255.0.0

Router(config-if)#tunnel source se0/1/0

Router(config-if)#tunnel destination 10.0.0.1
```



```
%SYS-5-CONFIG_I: Configured from console by console

Router#ping 172.20.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.20.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
Router(config)#int tunnel 400

Router(config-if)#
%LINK-5-CHANGED: Interface Tunnel400, changed state to up

Router(config-if)#ip address 172.18.1.2 255.255.0.0
Router(config-if)#tunnel source se0/1/0
Router(config-if)#tunnel destination 10.0.0.1
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel400, changed state to
up

Router(config-if)#no shut
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
```



```
Router(config)#interface tunnel 40

Router(config-if)#
%LINK-5-CHANGED: Interface Tunnel40, changed state to up

Router(config-if)#ip address 172.16.1.1 255.255.0.0
% 172.16.0.0 overlaps with Tunnel20
Router(config-if)#ip address 172.20.1.1 255.255.0.0
Router(config-if)#tunnel source gig0/0
Router(config-if)#tunnel destination 20.0.0.2
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel40, changed state to up

Router(config-if)#no shut
Router(config-if)#exit
Router(config)#int tunnel 200

Router(config-if)#
%LINK-5-CHANGED: Interface Tunnel200, changed state to up

Router(config-if)#ip address 172.18.1.1 255.255.0.0
Router(config-if)#tunnel source se0/1/0
Router(config-if)#tunnel destination 20.0.0.2
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel200, changed state to
up

Router(config-if)#no shut
Router(config-if)#
```

```
Router#ping 172.18.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.18.1.2, timeout is 2 seconds:

!!!!!
```

```
Router#ping 172.18.1.1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.18.1.1, timeout is 2 seconds:
```

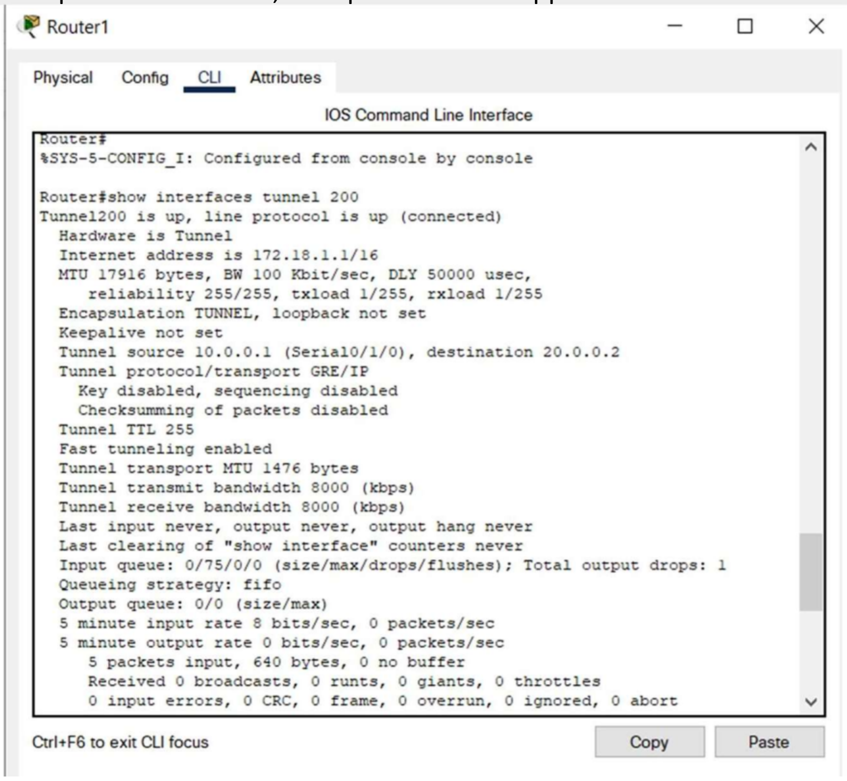
```
Router(config)#ip route 192.168.2.0 255.255.255.0 172.18.1.2

Router(config)#ip route 192.168.1.0 255.255.255.0 172.18.1.1
```

```
Router#show interfaces Tunnel 200

Tunnel200 is up, line protocol is up (connected)

Hardware is Tunnel
```

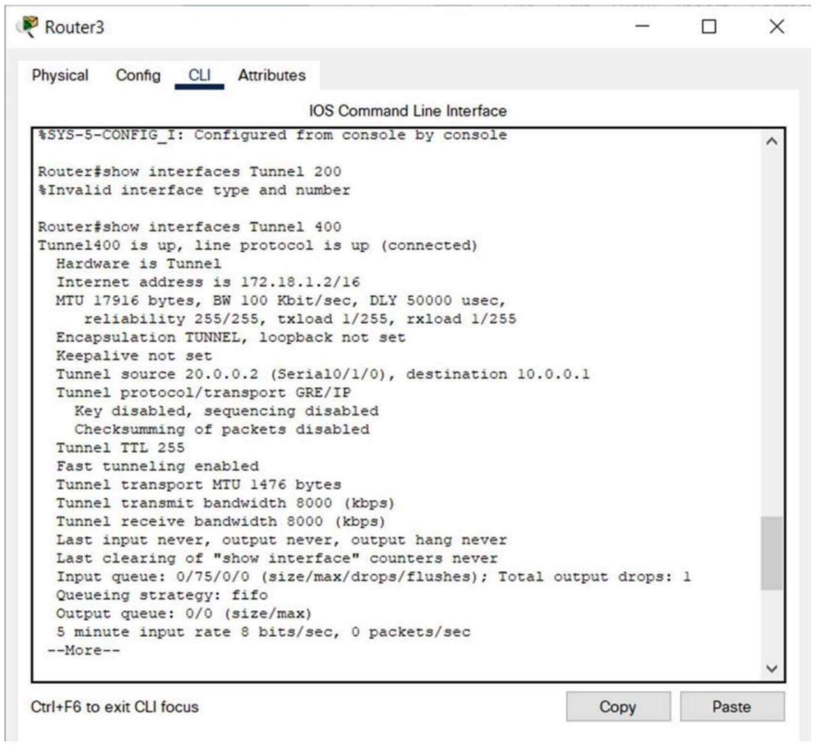
```
MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation TUNNEL, loopback not set

Keepalive not set

Tunnel source 10.0.0.1 (FastEthernet0/1), destination 20.0.0.2

Tunnel protocol/transport GRE/IP

Key disabled, sequencing disabled

Checksumming of packets disabled

Tunnel TTL 255

Fast tunneling enabled

Tunnel transport MTU 1476 bytes

Tunnel transmit bandwidth 8000 (kbps)

Tunnel receive bandwidth 8000 (kbps)

Last input never, output never, output hang never

Last clearing of "show interface" counters never

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1

Queueing strategy: fifo

Output queue: 0/0 (size/max)

5 minute input rate 32 bits/sec, 0 packets/sec

5 minute output rate 32 bits/sec, 0 packets/sec

52 packets input, 3508 bytes, 0 no buffer

Received 0 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

0 input packets with dribble condition detected

52 packets output, 3424 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets
```

0 unknown protocol drops

0 output buffer failures, 0 output buffers swapped out



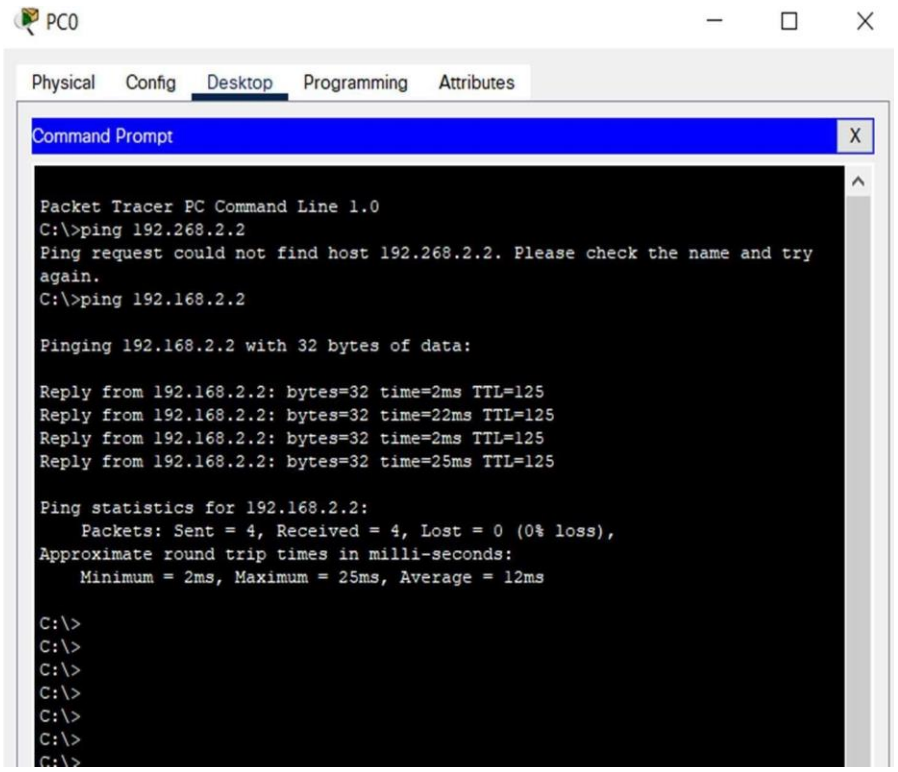
```
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#show interfaces tunnel 200
Tunnel200 is up, line protocol is up (connected)
  Hardware is Tunnel
  Internet address is 172.18.1.1/16
  MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.0.0.1 (Serial0/1/0), destination 20.0.0.2
  Tunnel protocol/transport GRE/IP
    Key disabled, sequencing disabled
    Checksumming of packets disabled
  Tunnel TTL 255
  Fast tunneling enabled
  Tunnel transport MTU 1476 bytes
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 8 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     5 packets input, 640 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
```



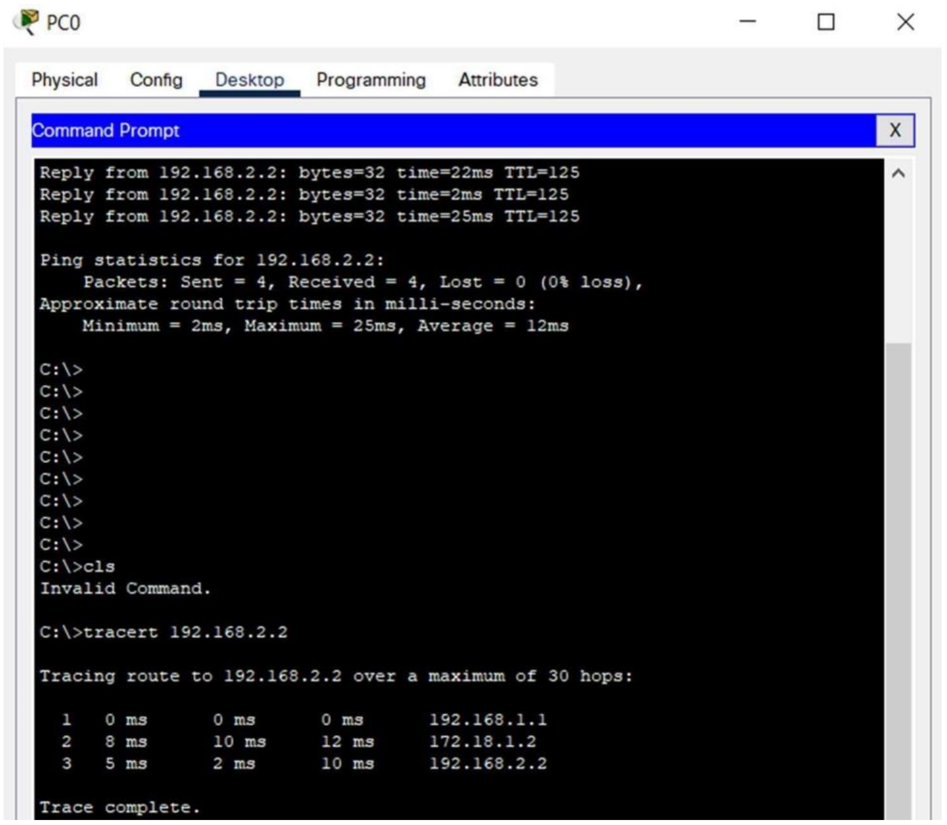
```
%SYS-5-CONFIG_I: Configured from console by console

Router#show interfaces Tunnel 200
%Invalid interface type and number

Router#show interfaces Tunnel 400
Tunnel400 is up, line protocol is up (connected)
  Hardware is Tunnel
  Internet address is 172.18.1.2/16
  MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 20.0.0.2 (Serial0/1/0), destination 10.0.0.1
  Tunnel protocol/transport GRE/IP
    Key disabled, sequencing disabled
    Checksumming of packets disabled
  Tunnel TTL 255
  Fast tunneling enabled
  Tunnel transport MTU 1476 bytes
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 8 bits/sec, 0 packets/sec
 --More--
```

```
Router #show interface Tunnel 400

Tunnel400 is up, line protocol is up (connected)

Hardware is Tunnel

Internet address is 172.18.1.2/16

MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation TUNNEL, loopback not set

Keepalive not set

Tunnel source 20.0.0.2 (FastEthernet0/0), destination 10.0.0.1

Tunnel protocol/transport GRE/IP

Key disabled, sequencing disabled

Checksumming of packets disabled

Tunnel TTL 255

Fast tunneling enabled

Tunnel transport MTU 1476 bytes

Tunnel transmit bandwidth 8000 (kbps)

Tunnel receive bandwidth 8000 (kbps)

Last input never, output never, output hang never

Last clearing of "show interface" counters never

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1

Queueing strategy: fifo


Output queue: 0/0 (size/max)

5 minute input rate 32 bits/sec, 0 packets/sec
```

```
5 minute output rate 32 bits/sec, 0 packets/sec


52 packets input, 3424 bytes, 0 no buffer


Received 0 broadcasts, 0 runts, 0 giants, 0 throttles


0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

0 input packets with dribble condition detected


53 packets output, 3536 bytes, 0 underruns
```



```
Packet Tracer PC Command Line 1.0
C:\>ping 192.268.2.2
Ping request could not find host 192.268.2.2. Please check the name and try
again.
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=22ms TTL=125
Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=25ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 25ms, Average = 12ms

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
```



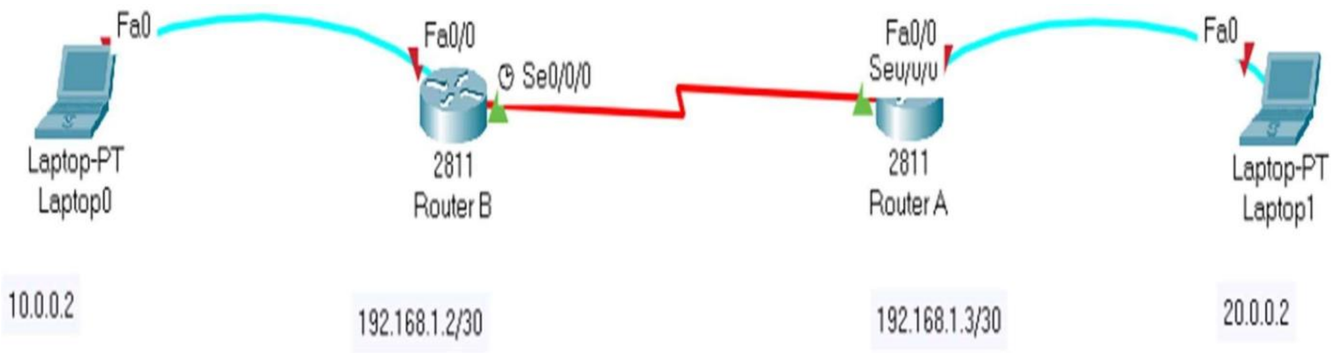
```
Reply from 192.168.2.2: bytes=32 time=22ms TTL=125
Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=25ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 25ms, Average = 12ms

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>cls
Invalid Command.

C:\>tracert 192.168.2.2

Tracing route to 192.168.2.2 over a maximum of 30 hops:

  1   0 ms      0 ms      0 ms      192.168.1.1
  2   8 ms      10 ms     12 ms     172.18.1.2
  3   5 ms      2 ms      10 ms     192.168.2.2

Trace complete.
```

**RESULT:**

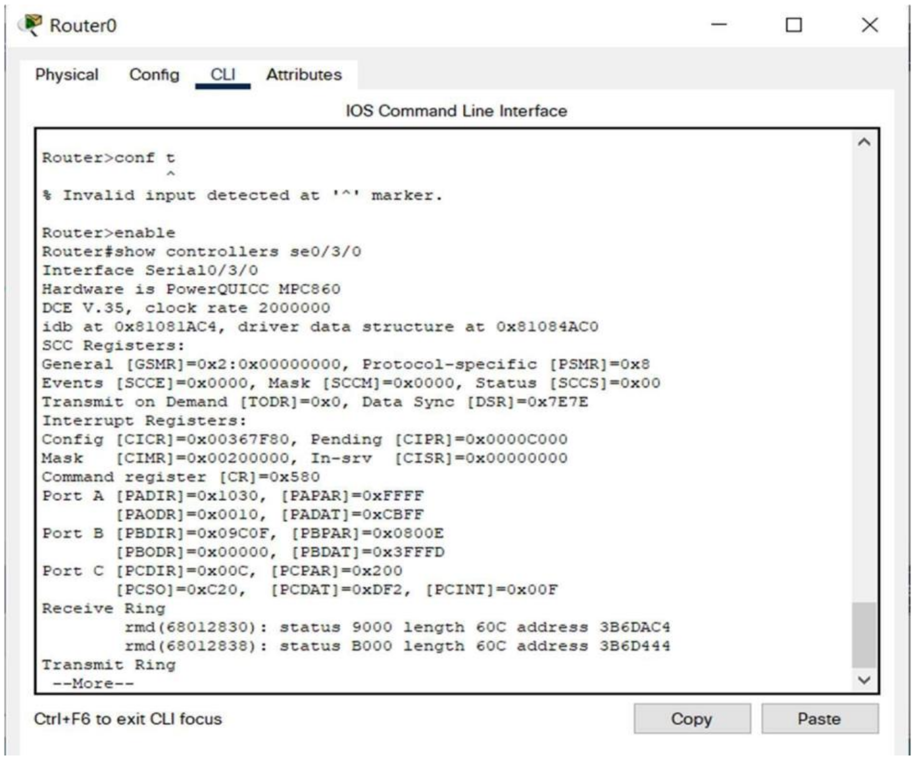
| | |
|---|---|
| **Ex.No:** | |
| **Date:** | |

**AIM:**

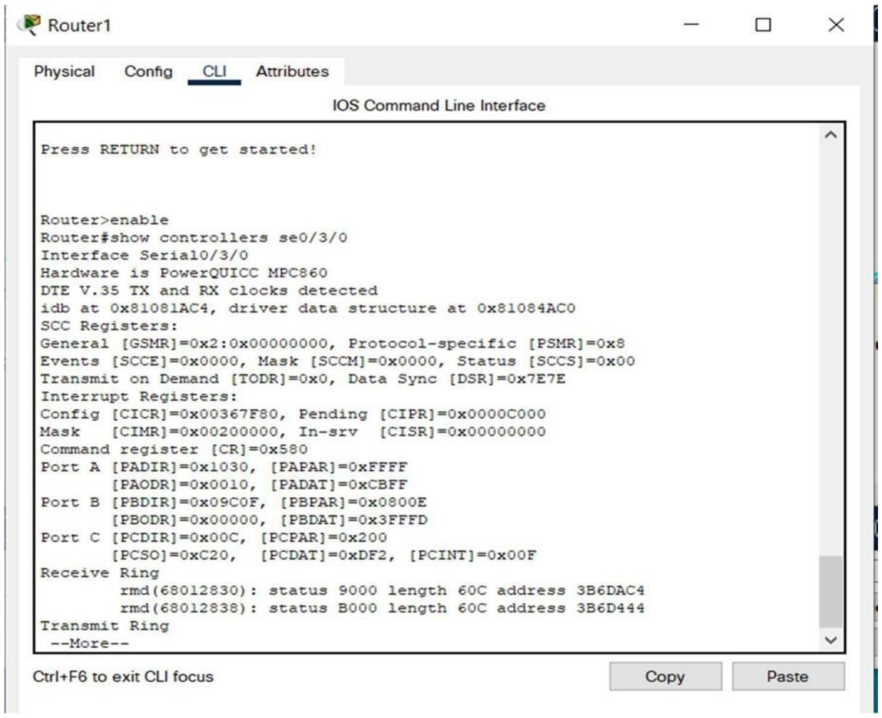
**PROCEDURE:**



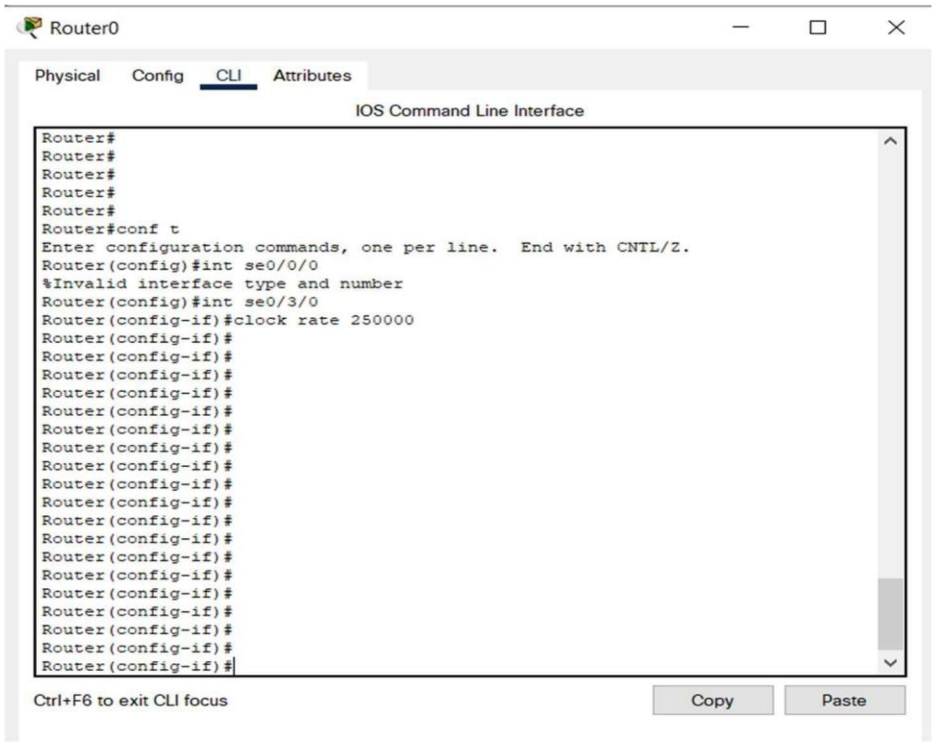
| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |



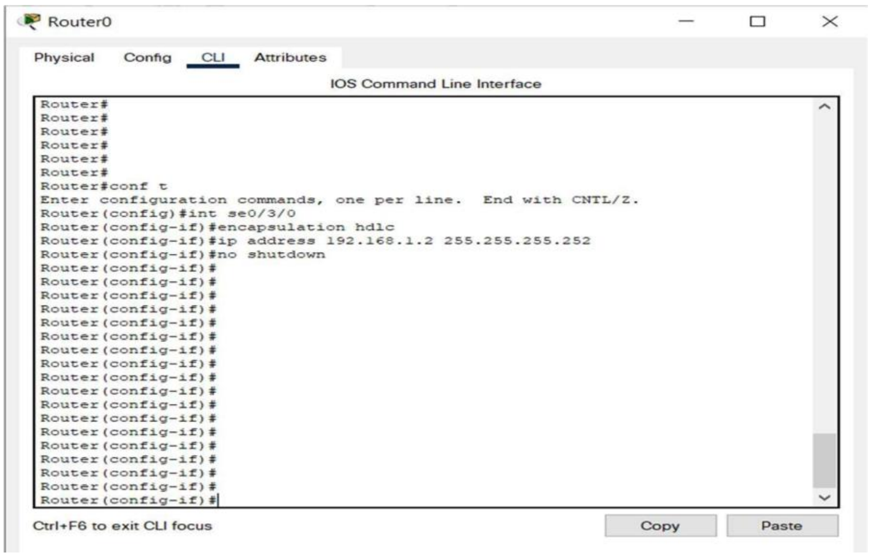
```
Router>conf t
            ^
% Invalid input detected at '^' marker.

Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DCE V.35, clock rate 2000000
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask   [CIMR]=0x00200000, In-srv  [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
       [PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
       [PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
       [PCSO]=0xC20,  [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
        rmd(68012830): status 9000 length 60C address 3B6DAC4
        rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
 --More--
```



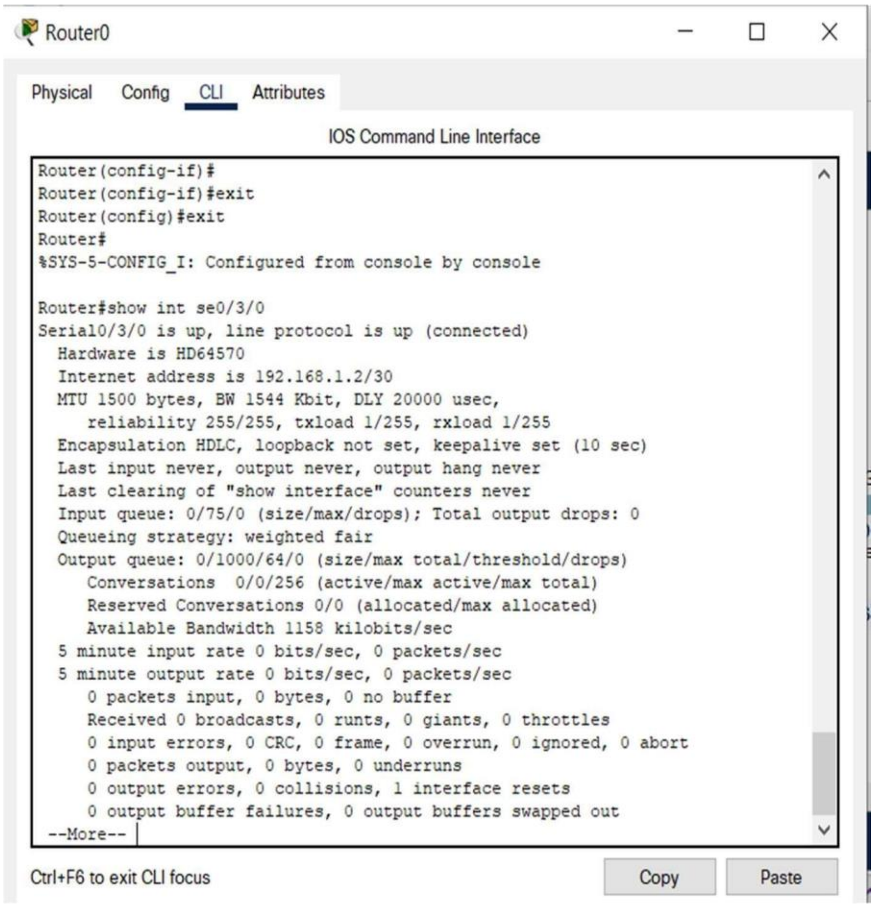
```
Press RETURN to get started!



Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DTE V.35 TX and RX clocks detected
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask   [CIMR]=0x00200000, In-srv  [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
       [PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
       [PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
       [PCSO]=0xC20,  [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
        rmd(68012830): status 9000 length 60C address 3B6DAC4
        rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
 --More--
```



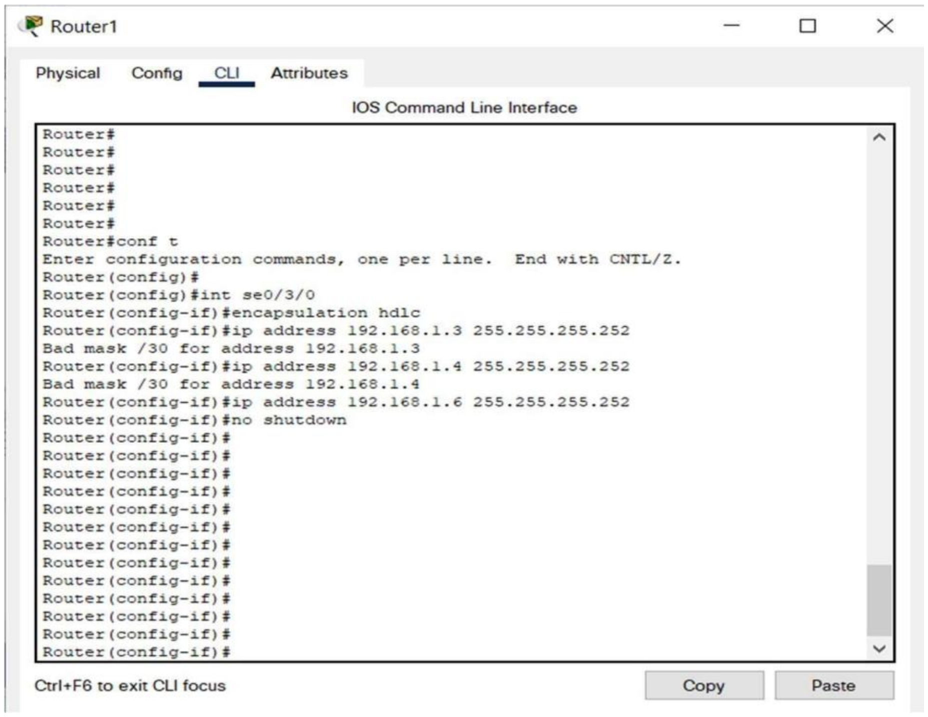
```
Router#
Router#
Router#
Router#
Router#
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int se0/0/0
%Invalid interface type and number
Router(config)#int se0/3/0
Router(config-if)#clock rate 250000
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
```



```
Router#
Router#
Router#
Router#
Router#
Router#
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int se0/3/0
Router(config-if)#encapsulation hdlc
Router(config-if)#ip address 192.168.1.2 255.255.255.252
Router(config-if)#no shutdown
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
```



```
Router(config-if)#
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#show int se0/3/0
Serial0/3/0 is up, line protocol is up (connected)
  Hardware is HD64570
  Internet address is 192.168.1.2/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
 --More--
```



```
Router#
Router#
Router#
Router#
Router#
Router#
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
Router(config)#int se0/3/0
Router(config-if)#encapsulation hdlc
Router(config-if)#ip address 192.168.1.3 255.255.255.252
Bad mask /30 for address 192.168.1.3
Router(config-if)#ip address 192.168.1.4 255.255.255.252
Bad mask /30 for address 192.168.1.4
Router(config-if)#ip address 192.168.1.6 255.255.255.252
Router(config-if)#no shutdown
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
```

```
Router#ping 192.168.1.6

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 198.168.1.6, timeout is 2 seconds:
```

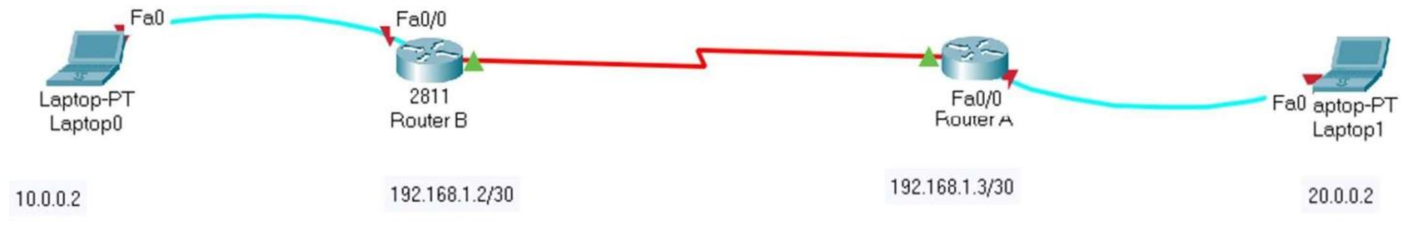
```
Router#ping 192.168.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
```

**RESULT** :

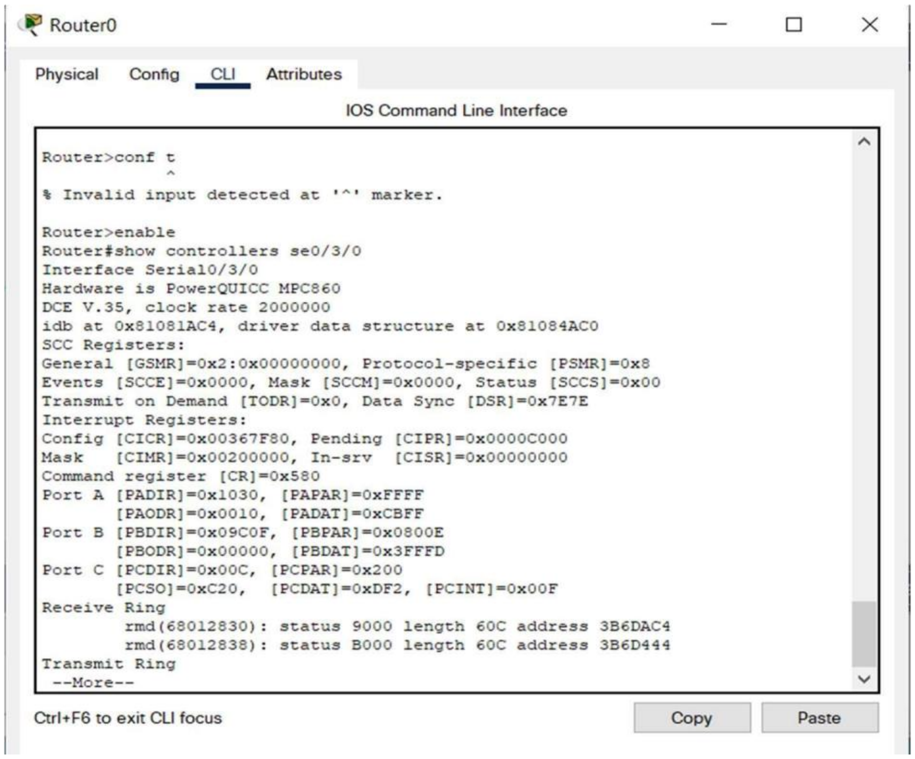
| | |
|---|---|
| **Ex.No:** | |
| **Date:** | |

**AIM:**

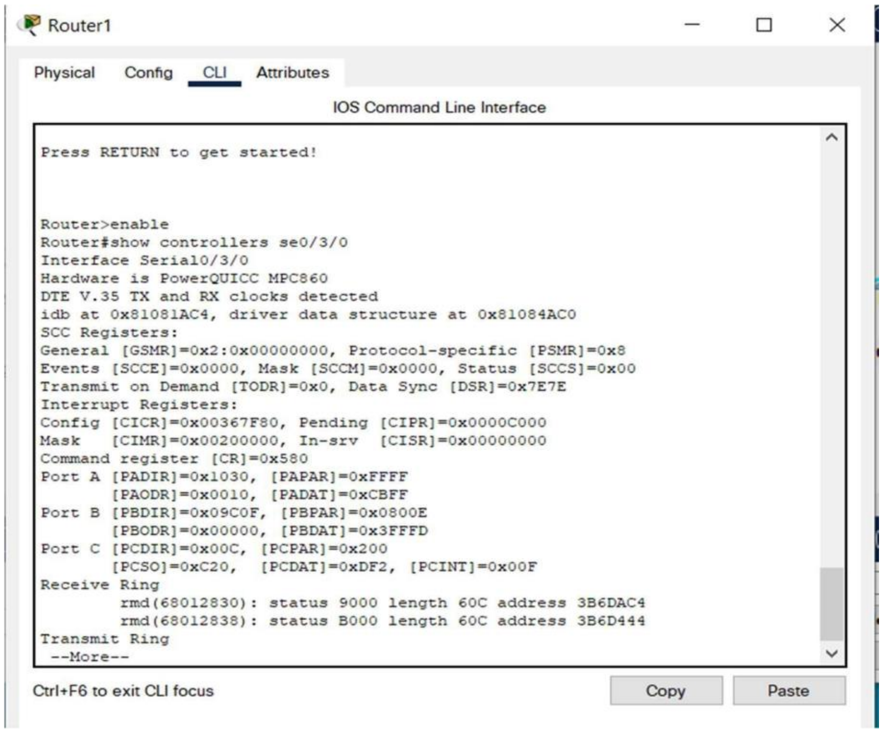
**PROCEDURE:**



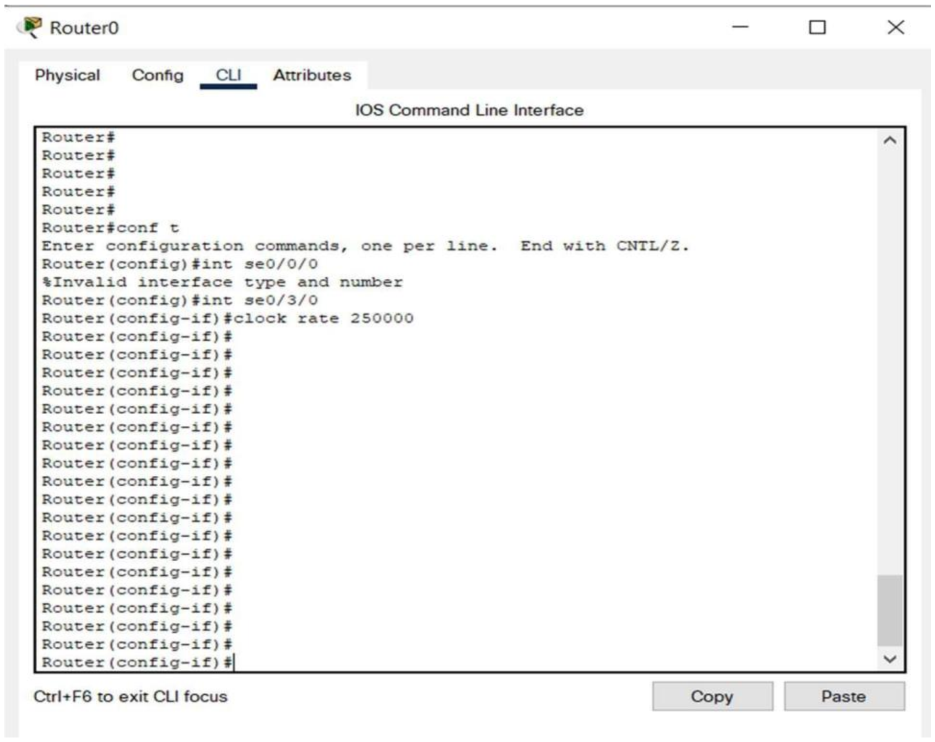
| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |



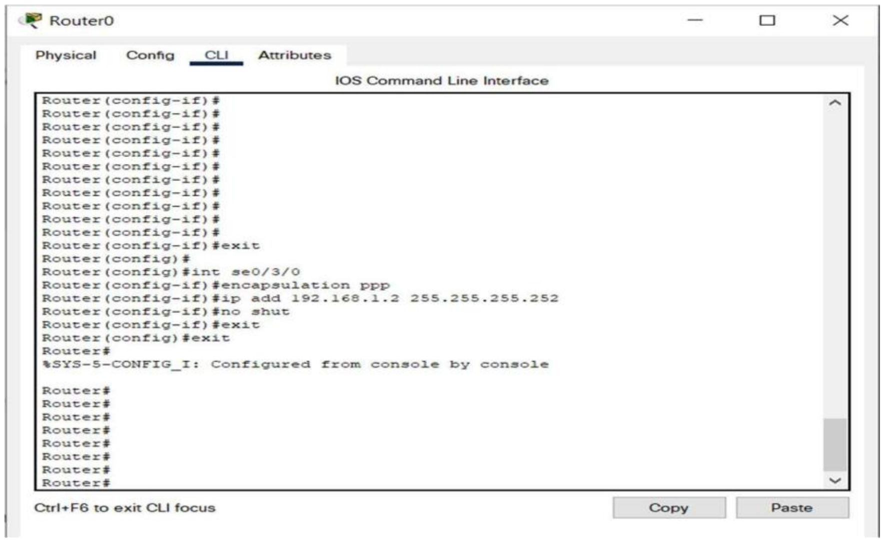
```
Router>conf t
           ^
% Invalid input detected at '^' marker.

Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DCE V.35, clock rate 2000000
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask   [CIMR]=0x00200000, In-srv  [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
       [PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
       [PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
       [PCSO]=0xC20,  [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
        rmd(68012830): status 9000 length 60C address 3B6DAC4
        rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
 --More--
```



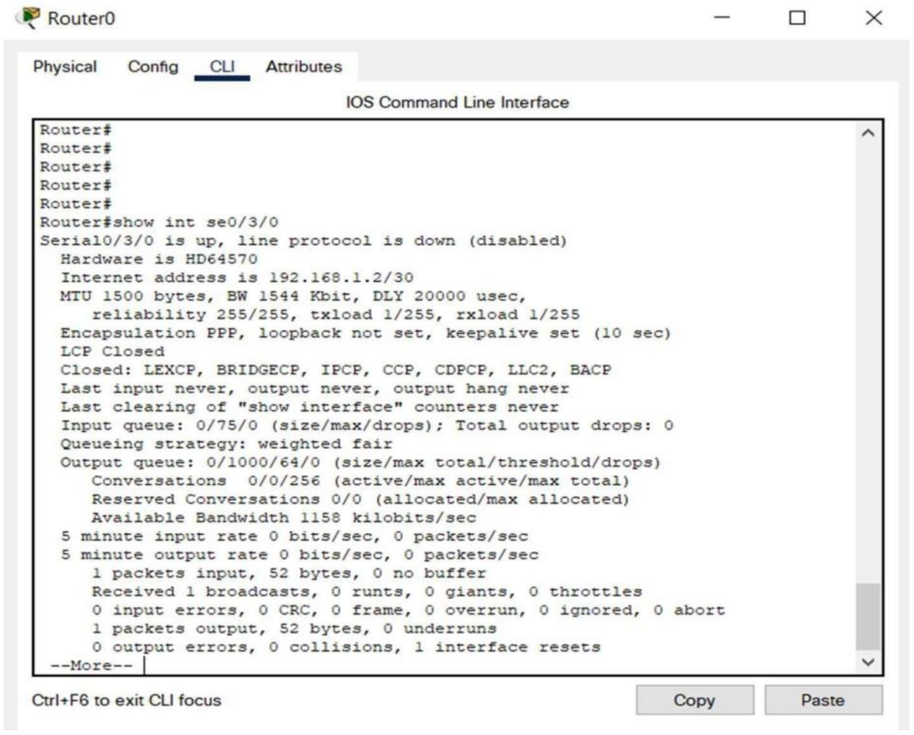
```
Press RETURN to get started!



Router>enable
Router#show controllers se0/3/0
Interface Serial0/3/0
Hardware is PowerQUICC MPC860
DTE V.35 TX and RX clocks detected
idb at 0x81081AC4, driver data structure at 0x81084AC0
SCC Registers:
General [GSMR]=0x2:0x00000000, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x0000, Status [SCCS]=0x00
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x0000C000
Mask   [CIMR]=0x00200000, In-srv  [CISR]=0x00000000
Command register [CR]=0x580
Port A [PADIR]=0x1030, [PAPAR]=0xFFFF
       [PAODR]=0x0010, [PADAT]=0xCBFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
       [PBODR]=0x00000, [PBDAT]=0x3FFFD
Port C [PCDIR]=0x00C, [PCPAR]=0x200
       [PCSO]=0xC20,  [PCDAT]=0xDF2, [PCINT]=0x00F
Receive Ring
        rmd(68012830): status 9000 length 60C address 3B6DAC4
        rmd(68012838): status B000 length 60C address 3B6D444
Transmit Ring
 --More--
```



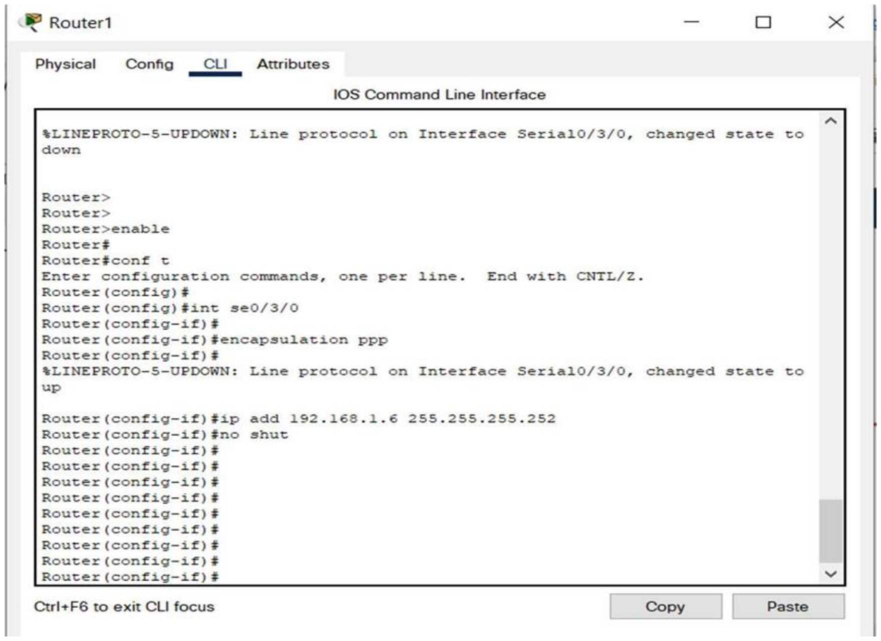
```
Router#
Router#
Router#
Router#
Router#
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int se0/0/0
%Invalid interface type and number
Router(config)#int se0/3/0
Router(config-if)#clock rate 250000
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
```



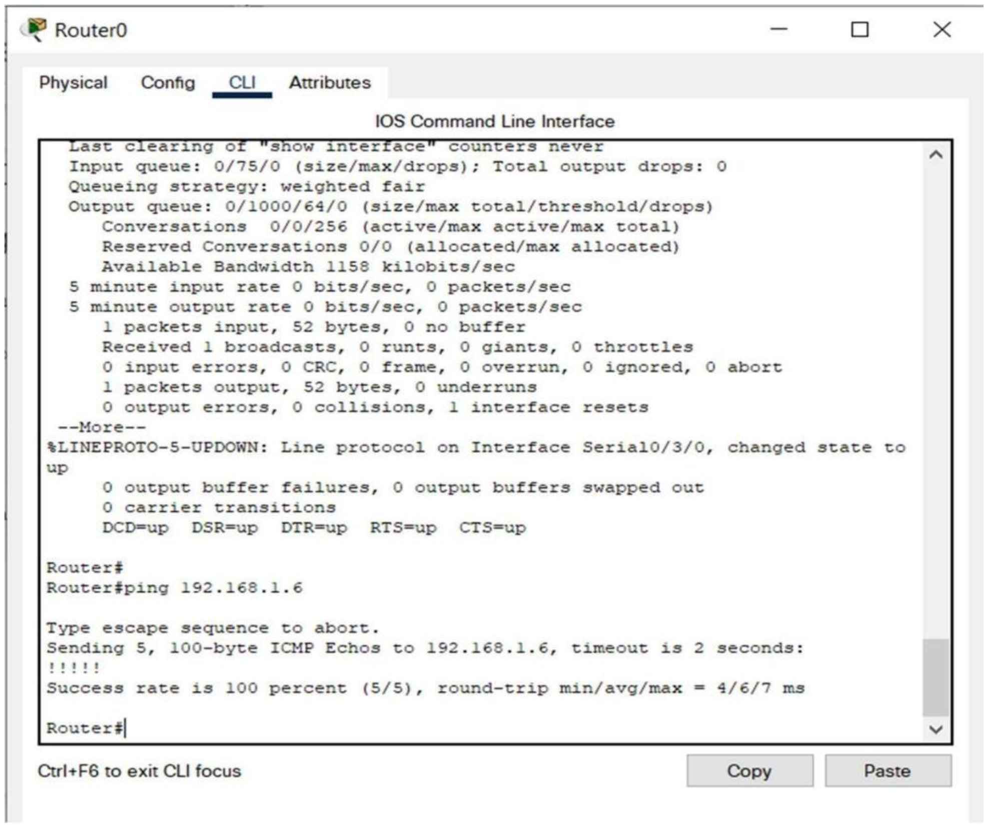
```
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#exit
Router(config)#
Router(config)#int se0/3/0
Router(config-if)#encapsulation ppp
Router(config-if)#ip add 192.168.1.2 255.255.255.252
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#
Router#
Router#
Router#
Router#
Router#
Router#
Router#
```



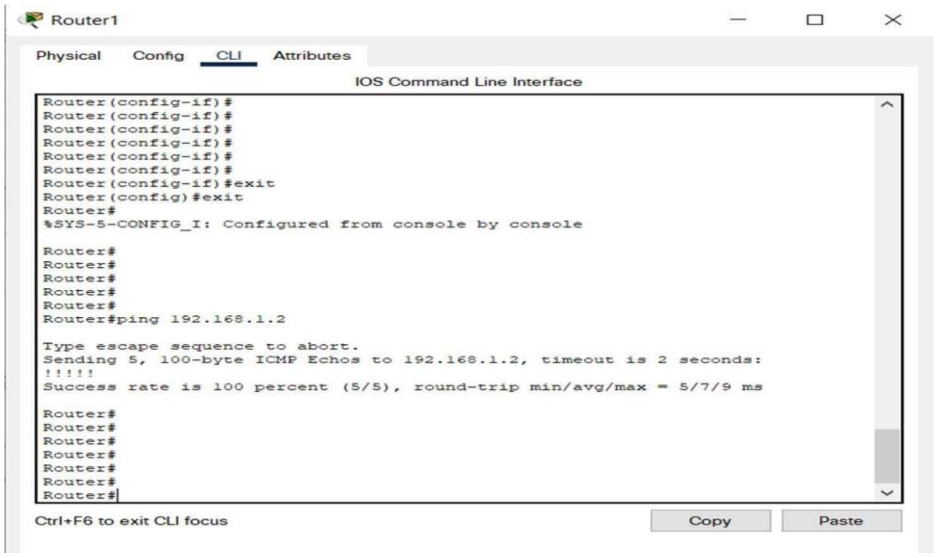
```
Router#
Router#
Router#
Router#
Router#
Router#show int se0/3/0
Serial0/3/0 is up, line protocol is down (disabled)
  Hardware is HD64570
  Internet address is 192.168.1.2/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, loopback not set, keepalive set (10 sec)
  LCP Closed
  Closed: LEXCP, BRIDGECP, IPCP, CCP, CDPCP, LLC2, BACP
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     1 packets input, 52 bytes, 0 no buffer
     Received 1 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1 packets output, 52 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
 --More--
```



```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/3/0, changed state to
down


Router>
Router>
Router>enable
Router#
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
Router(config)#int se0/3/0
Router(config-if)#
Router(config-if)#encapsulation ppp
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/3/0, changed state to
up

Router(config-if)#ip add 192.168.1.6 255.255.255.252
Router(config-if)#no shut
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
```



```
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     1 packets input, 52 bytes, 0 no buffer
     Received 1 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1 packets output, 52 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
 --More--
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/3/0, changed state to
up
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

Router#
Router#ping 192.168.1.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/6/7 ms

Router#
```



```
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#
Router#
Router#
Router#
Router#
Router#ping 192.168.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 5/7/9 ms

Router#
Router#
Router#
Router#
Router#
Router#
Router#
```

**RESULT:**