INTRODUCTION-1,2

---

1. Use Linux tools like ifconfig, dig, ethtool, route,netstat,nslookup,and ip to understand the networking configuration of the computer that thestudentisworkingon

```
devna@DEVNA: ~
devna@DEVNA:~$ dig

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37206
;; flags: qr rd ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       0       IN      NS      h.root-servers.net.
.                       0       IN      NS      l.root-servers.net.
.                       0       IN      NS      b.root-servers.net.
.                       0       IN      NS      c.root-servers.net.
.                       0       IN      NS      d.root-servers.net.
.                       0       IN      NS      a.root-servers.net.
.                       0       IN      NS      m.root-servers.net.
.                       0       IN      NS      e.root-servers.net.
.                       0       IN      NS      f.root-servers.net.
.                       0       IN      NS      i.root-servers.net.
.                       0       IN      NS      j.root-servers.net.
.                       0       IN      NS      k.root-servers.net.
.                       0       IN      NS      g.root-servers.net.

;; Query time: 40 msec
;; SERVER: 172.25.192.1#53(172.25.192.1) (UDP)
;; WHEN: Sun Mar 16 05:56:56 UTC 2025
```

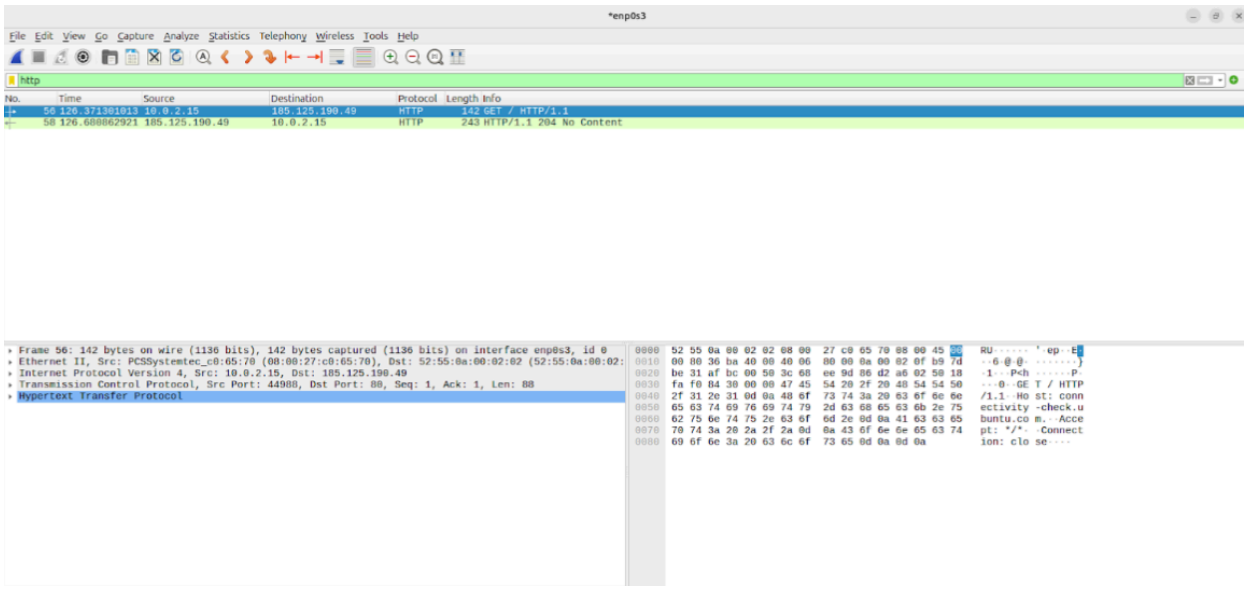
```
devna@DEVNA: ~

                 | -s | --change
                 | -k | --show-features | --show-offload | -K | --features | --offload
                 | --show-priv-flags | --set-priv-flags
                 | -g | --show-ring | -G | --set-ring
                 | -l | --show-channels | -L | --set-channels
                 | -c | --show-coalesce | -C | --coalesce
                 | -a | --show-pause | -A | --pause
                 | --show-eee | --set-eee
                 | --cable-test
                 | --cable-test-tdr
                 | --show-module | --set-module )
               [ DEVNAME | * ]

FLAGS:
        --debug MASK    turn on debugging messages
        --json          enable JSON output format (not supported by all commands)
        -I|--include-statistics         request device statistics related to the command (not supported by all commands)
devna@DEVNA:~$ sudo ethtool eth0
Settings for eth0:
        Supported ports: [ ]
        Supported link modes:   Not reported
        Supported pause frame use: No
        Supports auto-negotiation: No
        Supported FEC modes: Not reported
        Advertised link modes:  Not reported
        Advertised pause frame use: No
        Advertised auto-negotiation: No
        Advertised FEC modes: Not reported
        Speed: 10000Mb/s
        Duplex: Full
        Port: Other
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: off
        Current message level: 0x000000f7 (247)
                               drv probe link ifdown ifup rx_err tx_err
        Link detected: yes
devna@DEVNA:~$
```

```
devna@DEVNA: ~

        Supported pause frame use: No
        Supports auto-negotiation: No
        Supported FEC modes: Not reported
        Advertised link modes:  Not reported
        Advertised pause frame use: No
        Advertised auto-negotiation: No
        Advertised FEC modes: Not reported
        Speed: 10000Mb/s
        Duplex: Full
        Port: Other
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: off
        Current message level: 0x000000f7 (247)
                               drv probe link ifdown ifup rx_err tx_err
        Link detected: yes
devna@DEVNA:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: bond0: <BROADCAST,MULTICAST,MASTER> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 96:d4:28:3c:3f:b1 brd ff:ff:ff:ff:ff:ff
3: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether a6:a5:a6:c1:75:23 brd ff:ff:ff:ff:ff:ff
4: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
5: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
6: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:84:ee:0e brd ff:ff:ff:ff:ff:ff
    inet 172.25.202.131/20 brd 172.25.207.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe84:ee0e/64 scope link
       valid_lft forever preferred_lft forever
devna@DEVNA:~$
```

```
devna@DEVNA:~$ ip route show
default via 172.25.192.1 dev eth0
172.25.192.0/20 dev eth0 proto kernel scope link src 172.25.202.131
devna@DEVNA:~$
```

APPLICATION LAYER-1

1. Use Wireshark packet capture to analyze various header fields and their usage in different application layer protocols likeHTTP,SMTPand FTP



Transport layer-1,2,3,4,5,8

1. Using Wireshark, observe three way handshaking connection establishment, three way handshaking connection termination and Data transfer in client server communication using TCP.

3970	104.715860	192.168.70.204	192.168.70.81	TCP	66	53402 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
3971	104.717752	192.168.70.81	192.168.70.204	TCP	66	53 → 53400 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM WS=512
3972	104.717851	192.168.70.204	192.168.70.81	TCP	54	53400 → 53 [ACK] Seq=1 Ack=1 Win=131328 Len=0

2. Write the system calls used for creating sockets and transferring data between two nodes.

Server

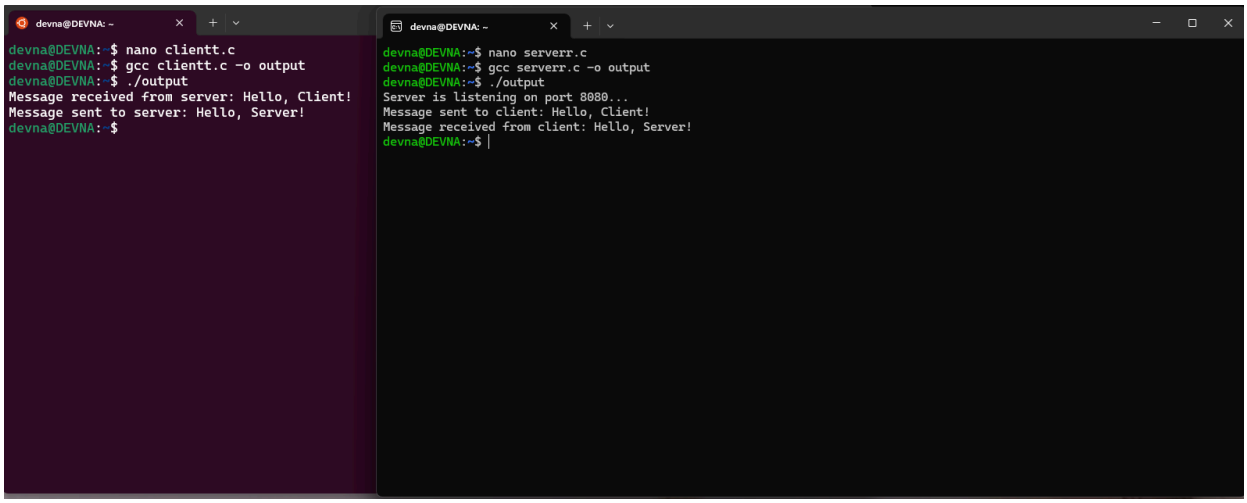
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
   int server_sock, client_sock;
   struct sockaddr_in server_addr, client_addr;
   char buffer[1024];

   server_sock = socket(AF_INET, SOCK_STREAM, 0);

   server_addr.sin_family = AF_INET;
   server_addr.sin_addr.s_addr = INADDR_ANY;
   server_addr.sin_port = htons(8080);

   bind(server_sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
   listen(server_sock, 5);

   int addr_len = sizeof(client_addr);
   client_sock = accept(server_sock, (struct sockaddr*)&client_addr,
&addr_len);

   recv(client_sock, buffer, sizeof(buffer), 0);
   printf("Received from client: %s\n", buffer);

   send(client_sock, "Hello, Client!", 14, 0);

   close(client_sock);
   close(server_sock);

   return 0;
```

```
}
```

Client

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
   int sock;
   struct sockaddr_in server_addr;
   char buffer[1024] = "Hello, Server!";

   sock = socket(AF_INET, SOCK_STREAM, 0);

   server_addr.sin_family = AF_INET;
   server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
   server_addr.sin_port = htons(8080);

   connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));

   send(sock, buffer, strlen(buffer), 0);

   recv(sock, buffer, sizeof(buffer), 0);
   printf("Received from server: %s\n", buffer);

   close(sock);

   return 0;
}
```

OUTPUT



3) Write a program to find the maximum, minimum and average of an array of integers using socket programming.

Server :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>

int main() {
    int server_socket, client_socket, n, arr[100], max, min, sum = 0;
    float avg;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_size;

    // Create socket
```

```
server_socket = socket(AF_INET, SOCK_STREAM, 0);
if (server_socket < 0) {
    perror("Socket Error");
    exit(1);
}
printf("Server Socket Created...\n");

// Server address structure
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(8080);
server_addr.sin_addr.s_addr = INADDR_ANY;

// Bind socket
bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr));
printf("Bind Successful...\n");

// Listen for clients
listen(server_socket, 5);
printf("Waiting for Client...\n");

addr_size = sizeof(client_addr);
client_socket = accept(server_socket, (struct sockaddr*)&client_addr, &addr_size);
printf("Client Connected...\n");

// Receive array size
recv(client_socket, &n, sizeof(n), 0);
recv(client_socket, arr, n * sizeof(int), 0);
printf("Received Array from Client...\n");

// Calculate Max, Min, and Average
max = min = arr[0];
for (int i = 0; i < n; i++) {
    if (arr[i] > max)
        max = arr[i];
    if (arr[i] < min)
        min = arr[i];
    sum += arr[i];
}
avg = (float)sum / n;

// Send Results
send(client_socket, &max, sizeof(max), 0);
send(client_socket, &min, sizeof(min), 0);
send(client_socket, &avg, sizeof(avg), 0);
```

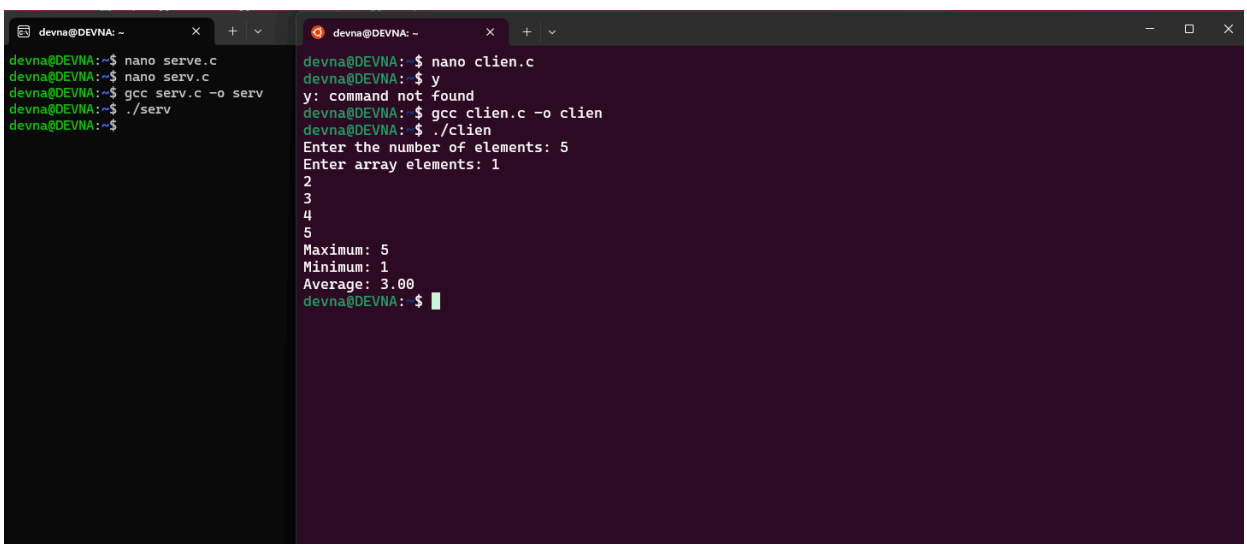
```
    printf("Results Sent to Client...\n");

    close(client_socket);
    close(server_socket);
    return 0;
}
```

Client :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>

int main() {
    int client_socket, n, arr[100], max, min;
    float avg;
    struct sockaddr_in server_addr;

    // Create socket
    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket < 0) {
        perror("Socket Error");
        exit(1);
    }
    printf("Client Socket Created...\n");

    // Server address
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(8080);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Connect to server
    connect(client_socket, (struct sockaddr*)&server_addr, sizeof(server_addr));
    printf("Connected to Server...\n");

    // Input Array
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    printf("Enter array elements: ");
```

```
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Send Array
    send(client_socket, &n, sizeof(n), 0);
    send(client_socket, arr, n * sizeof(int), 0);

    // Receive Results
    recv(client_socket, &max, sizeof(max), 0);
    recv(client_socket, &min, sizeof(min), 0);
    recv(client_socket, &avg, sizeof(avg), 0);

    // Print Results
    printf("Maximum: %d\n", max);
    printf("Minimum: %d\n", min);
    printf("Average: %.2f\n", avg);

    close(client_socket);
    return 0;
}
```

OUTPUT



4. (a)Create three programs, two of which are clients to a single server. Client1 will send a string to the server process using datagram socket and stream socket. The server will reverse the string and send the result to Client2. Client2 prints the reversed string it receives and then all the processes terminate.

Server :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

void reverseString(char *str) {
   int len = strlen(str);
   for (int i = 0; i < len / 2; i++) {
       char temp = str[i];
       str[i] = str[len - i - 1];
       str[len - i - 1] = temp;
   }
}
```

```
int main() {
   int udp_socket, tcp_socket, client2_socket;
   struct sockaddr_in udp_addr, client1_addr, tcp_addr, client2_addr;
   socklen_t client1_len = sizeof(client1_addr);
   char buffer[1024];

   // UDP SOCKET
   udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
   udp_addr.sin_family = AF_INET;
   udp_addr.sin_port = htons(8080);
   udp_addr.sin_addr.s_addr = INADDR_ANY;
   bind(udp_socket, (struct sockaddr*)&udp_addr, sizeof(udp_addr));

   printf("Server is waiting for Client1 (UDP)...\n");
   recvfrom(udp_socket, buffer, sizeof(buffer), 0, (struct
sockaddr*)&client1_addr, &client1_len);
   printf("Received from Client1: %s\n", buffer);

   reverseString(buffer);
   printf("Reversed String: %s\n", buffer);

   // TCP SOCKET
   tcp_socket = socket(AF_INET, SOCK_STREAM, 0);
   tcp_addr.sin_family = AF_INET;
   tcp_addr.sin_port = htons(9090);
   tcp_addr.sin_addr.s_addr = INADDR_ANY;
   bind(tcp_socket, (struct sockaddr*)&tcp_addr, sizeof(tcp_addr));
   listen(tcp_socket, 1);

   printf("Waiting for Client2 (TCP)...\n");
   client2_socket = accept(tcp_socket, (struct sockaddr*)&client2_addr,
&client1_len);
   printf("Client2 Connected!\n");

   send(client2_socket, buffer, strlen(buffer), 0);
   printf("Reversed String sent to Client2!\n");

   close(udp_socket);
   close(tcp_socket);
```

```
    close(client2_socket);

    return 0;
}
```

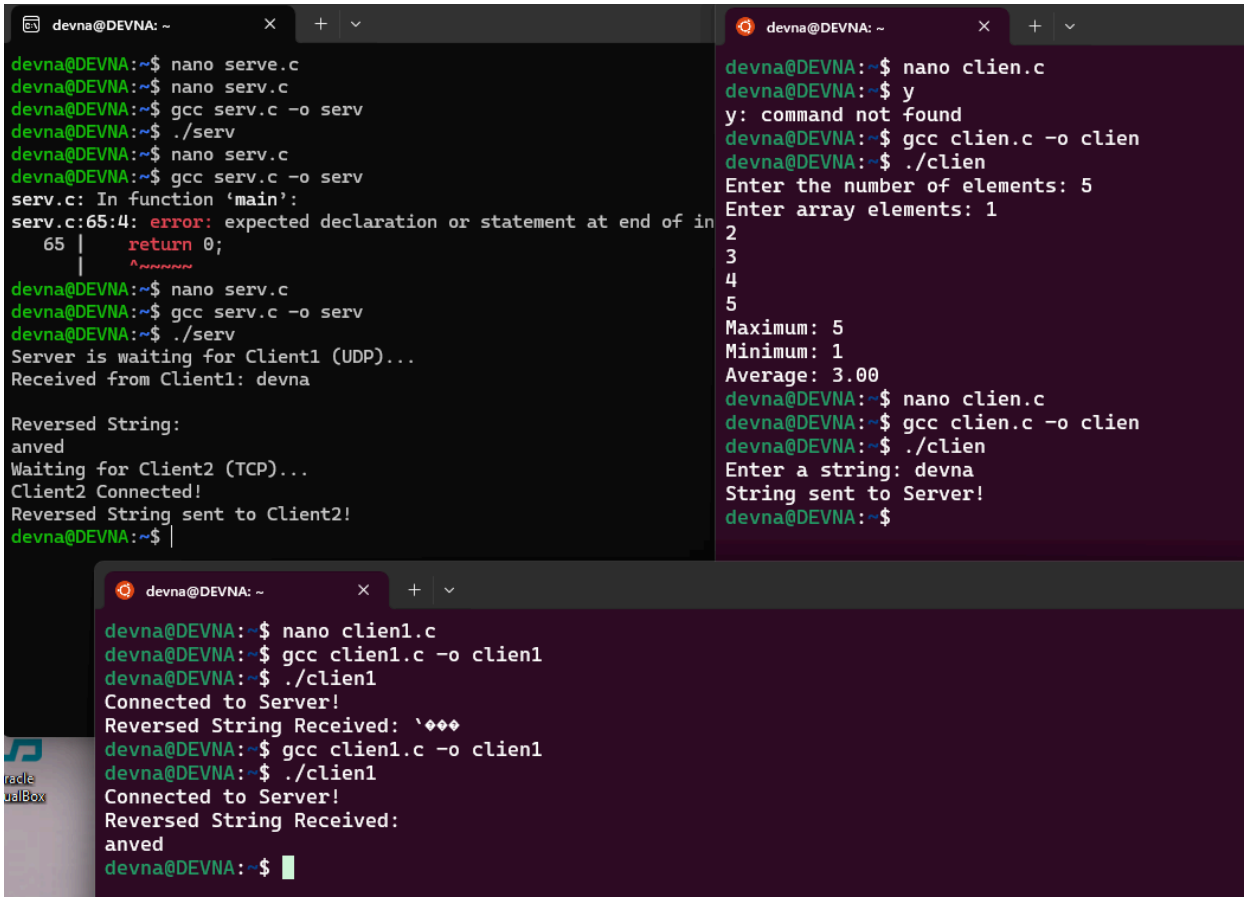
Client1 :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

int main() {
    int udp_socket;
    struct sockaddr_in server_addr;
    char buffer[1024];

    udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(8080);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    printf("Enter a string: ");
    fgets(buffer, sizeof(buffer), stdin);

    sendto(udp_socket, buffer, strlen(buffer), 0, (struct
sockaddr*)&server_addr, sizeof(server_addr));
    printf("String sent to Server!\n");

    close(udp_socket);

    return 0;
}
```

Client 2 :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

int main() {
   int tcp_socket;
   struct sockaddr_in server_addr;
   char buffer[1024];

   tcp_socket = socket(AF_INET, SOCK_STREAM, 0);
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(9090);
   server_addr.sin_addr.s_addr = INADDR_ANY;

   connect(tcp_socket, (struct sockaddr*)&server_addr,
sizeof(server_addr));
   printf("Connected to Server!\n");

   recv(tcp_socket, buffer, sizeof(buffer), 0);
   printf("Reversed String Received: %s\n", buffer);

   close(tcp_socket);

   return 0;
}
```

OUTPUT



(b) Follow the same procedure as in part a except that the data type of the message should be integer and the server should square the integer before transmitting it to Client2

Server :

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

```c
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

int main() {
   int udp_socket, tcp_socket, client2_socket, number, squared;
   struct sockaddr_in udp_addr, client1_addr, tcp_addr, client2_addr;
   socklen_t client1_len = sizeof(client1_addr);

   // UDP Socket
   udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
   udp_addr.sin_family = AF_INET;
   udp_addr.sin_port = htons(8080);
   udp_addr.sin_addr.s_addr = INADDR_ANY;
   bind(udp_socket, (struct sockaddr*)&udp_addr, sizeof(udp_addr));

   printf("Server waiting for Client1 (UDP)...\n");
   recvfrom(udp_socket, &number, sizeof(number), 0, (struct
sockaddr*)&client1_addr, &client1_len);
   printf("Received from Client1: %d\n", number);

   squared = number * number;
   printf("Squared Value: %d\n", squared);

   // TCP Socket
   tcp_socket = socket(AF_INET, SOCK_STREAM, 0);
   tcp_addr.sin_family = AF_INET;
   tcp_addr.sin_port = htons(9090);
   tcp_addr.sin_addr.s_addr = INADDR_ANY;
   bind(tcp_socket, (struct sockaddr*)&tcp_addr, sizeof(tcp_addr));
   listen(tcp_socket, 1);

   printf("Waiting for Client2 (TCP)...\n");
   client2_socket = accept(tcp_socket, (struct sockaddr*)&client2_addr,
&client1_len);
   printf("Client2 Connected!\n");

   send(client2_socket, &squared, sizeof(squared), 0);
   printf("Squared Value sent to Client2!\n");
```

```
    close(udp_socket);
    close(tcp_socket);
    close(client2_socket);

    return 0;
}
```

Client 1 :

```
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

int main() {
    int udp_socket, number;
    struct sockaddr_in server_addr;

    udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(8080);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    printf("Enter an integer: ");
    scanf("%d", &number);

    sendto(udp_socket, &number, sizeof(number), 0, (struct
sockaddr*)&server_addr, sizeof(server_addr));
    printf("Integer sent to Server!\n");

    close(udp_socket);

    return 0;
}
```

Client 2 :

```c
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

int main() {
   int tcp_socket, squared;
   struct sockaddr_in server_addr;

   tcp_socket = socket(AF_INET, SOCK_STREAM, 0);
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(9090);
   server_addr.sin_addr.s_addr = INADDR_ANY;

   connect(tcp_socket, (struct sockaddr*)&server_addr,
sizeof(server_addr));
   printf("Connected to Server!\n");

   recv(tcp_socket, &squared, sizeof(squared), 0);
   printf("Squared Value Received: %d\n", squared);

   close(tcp_socket);

   return 0;
```

OUTPUT

devna@DEVNA: ~

```
devna@DEVNA:~$ nano serv.c
devna@DEVNA:~$ gcc serv.c -o serv
devna@DEVNA:~$ ./serv
devna@DEVNA:~$ nano serv.c
devna@DEVNA:~$ gcc serv.c -o serv
serv.c: In function 'main':
serv.c:65:4: error: expected declaration
   65 |     return 0;
      |     ^~~~~~
devna@DEVNA:~$ nano serv.c
devna@DEVNA:~$ gcc serv.c -o serv
devna@DEVNA:~$ ./serv
Server is waiting for Client1 (UDP)...
Received from Client1: devna

Reversed String:
anved
Waiting for Client2 (TCP)...
Client2 Connected!
Reversed String sent to Client2!
devna@DEVNA:~$ nano serv.c
devna@DEVNA:~$ gcc serv.c -o serv
devna@DEVNA:~$ ./serv
Server waiting for Client1 (UDP)...
Received from Client1: 45
Squared Value: 2025
Waiting for Client2 (TCP)...
Client2 Connected!
Squared Value sent to Client2!
devna@DEVNA:~$
```

devna@DEVNA: ~

```
devna@DEVNA:~$ nano clien.c
devna@DEVNA:~$ y
y: command not found
devna@DEVNA:~$ gcc clien.c -o clien
devna@DEVNA:~$ ./clien
Enter the number of elements: 5
Enter array elements: 1
2
3
4
5
Maximum: 5
Minimum: 1
Average: 3.00
devna@DEVNA:~$ nano clien.c
devna@DEVNA:~$ gcc clien.c -o clien
devna@DEVNA:~$ ./clien
Enter a string: devna
String sent to Server!
devna@DEVNA:~$ nano clien.c
devna@DEVNA:~$ gcc clien.c -o clien
devna@DEVNA:~$ ./clien
Enter an integer: 45
Integer sent to Server!
devna@DEVNA:~$
```

devna@DEVNA: ~

```
devna@DEVNA:~$ nano clien1.c
devna@DEVNA:~$ gcc clien1.c -o clien1
devna@DEVNA:~$ ./clien1
Connected to Server!
Reversed String Received: `◆◆◆
devna@DEVNA:~$ gcc clien1.c -o clien1
devna@DEVNA:~$ ./clien1
Connected to Server!
Reversed String Received:
anved
devna@DEVNA:~$ nano clien1.c
devna@DEVNA:~$ gcc clien1.c -o clien1
devna@DEVNA:~$ ./clien1
Connected to Server!
Squared Value Received: 2025
devna@DEVNA:~$
```

(c) Write a socket program to enable Client1 to send a float value to the server. The server process should increase the value of the number it receives by a power of 1.5. The server should print both the value it receives and the value that it sends. Client2 should print the value it receives from the server

Server : //compile like: gcc 3cserver.c -o 3cserver -lm

```
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<math.h>

int main() {
   int udp_socket, tcp_socket, client2_socket;
   struct sockaddr_in udp_addr, client1_addr, tcp_addr, client2_addr;
   socklen_t client1_len = sizeof(client1_addr);
   float number, result;

   // UDP Socket
   udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
   udp_addr.sin_family = AF_INET;
   udp_addr.sin_port = htons(8080);
   udp_addr.sin_addr.s_addr = INADDR_ANY;
   bind(udp_socket, (struct sockaddr*)&udp_addr, sizeof(udp_addr));

   printf("Server waiting for Client1 (UDP)...\n");
   recvfrom(udp_socket, &number, sizeof(number), 0, (struct
sockaddr*)&client1_addr, &client1_len);
   printf("Received Float from Client1: %.2f\n", number);

   result = pow(number, 1.5);
   printf("Processed Value (Power 1.5): %.2f\n", result);
```

```
   // TCP Socket
   tcp_socket = socket(AF_INET, SOCK_STREAM, 0);
   tcp_addr.sin_family = AF_INET;
   tcp_addr.sin_port = htons(9090);
   tcp_addr.sin_addr.s_addr = INADDR_ANY;
   bind(tcp_socket, (struct sockaddr*)&tcp_addr, sizeof(tcp_addr));
   listen(tcp_socket, 1);

   printf("Waiting for Client2 (TCP)...\n");
   client2_socket = accept(tcp_socket, (struct sockaddr*)&client2_addr,
&client1_len);
   printf("Client2 Connected!\n");

   send(client2_socket, &result, sizeof(result), 0);
   printf("Processed Value sent to Client2!\n");

   close(udp_socket);
   close(tcp_socket);
   close(client2_socket);

   return 0;
}
```

Client 1:

```
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

int main() {
   int udp_socket;
   struct sockaddr_in server_addr;
   float number;

   udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
```

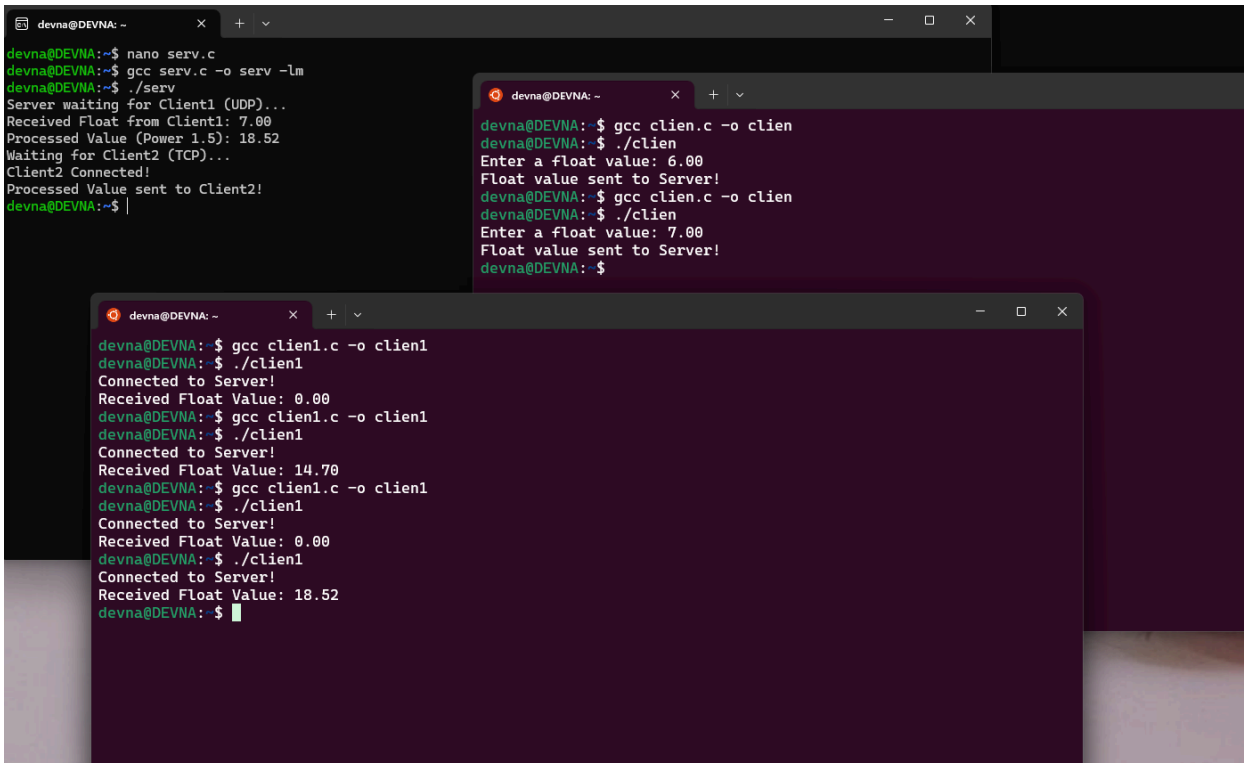
```
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(8080);
   server_addr.sin_addr.s_addr = INADDR_ANY;

   printf("Enter a float value: ");
   scanf("%f", &number);

   sendto(udp_socket, &number, sizeof(number), 0, (struct
sockaddr*)&server_addr, sizeof(server_addr));
   printf("Float value sent to Server!\n");

   close(udp_socket);

   return 0;
}
```

Client 2 :

```
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<unistd.h>

int main() {
   int tcp_socket;
   struct sockaddr_in server_addr;
   float result;

   tcp_socket = socket(AF_INET, SOCK_STREAM, 0);
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(9090);
   server_addr.sin_addr.s_addr = INADDR_ANY;

   connect(tcp_socket, (struct sockaddr*)&server_addr,
sizeof(server_addr));
   printf("Connected to Server!\n");
```

```
recv(tcp_socket, &result, sizeof(result), 0);
printf("Received Float Value: %.2f\n", result);

close(tcp_socket);

Ret}
```

OUTPUT



5. Implement a multi-user chat server using TCP as transport layer protocol.

Server :

```c
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<string.h>
#include<unistd.h>
#include<pthread.h>

#define MAX_CLIENTS 10
int clients[MAX_CLIENTS];
int count = 0;

void *client_handler(void *socket) {
   int client_socket = *(int*)socket;
   char msg[1024];

   while(1) {
       bzero(msg, sizeof(msg));
       recv(client_socket, msg, sizeof(msg), 0);
       if(strcmp(msg, "exit") == 0) {
           printf("Client Disconnected!\n");
           close(client_socket);
           break;
       }

       printf("Received: %s\n", msg);

       // Broadcasting to all clients
       for(int i = 0; i < count; i++) {
           if(clients[i] != client_socket) {
               send(clients[i], msg, sizeof(msg), 0);
           }
       }
   }
   pthread_exit(NULL);
}
```

```
int main() {
   int server_socket, client_socket;
   struct sockaddr_in server_addr, client_addr;
   socklen_t addr_size;
   pthread_t thread;

   server_socket = socket(AF_INET, SOCK_STREAM, 0);
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(8080);
   server_addr.sin_addr.s_addr = INADDR_ANY;

   bind(server_socket, (struct sockaddr*)&server_addr,
sizeof(server_addr));
   listen(server_socket, MAX_CLIENTS);
   printf("Server Waiting for Clients...\n");

   while(1) {
       addr_size = sizeof(client_addr);
       client_socket = accept(server_socket, (struct
sockaddr*)&client_addr, &addr_size);
       printf("Client Connected!\n");

       clients[count++] = client_socket;
       pthread_create(&thread, NULL, client_handler,
(void*)&client_socket);
   }
   close(server_socket);
   return 0;
}
```

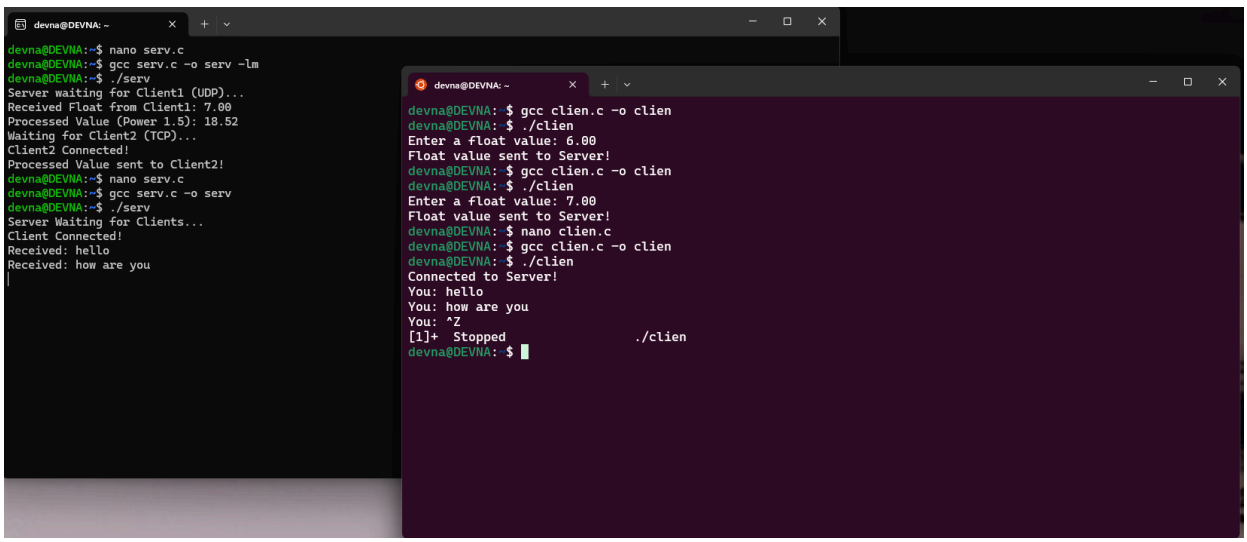
Client :

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

```c
#include<arpa/inet.h>
#include<string.h>
#include<unistd.h>
#include<pthread.h>

void *receive_msg(void *socket) {
   int client_socket = *(int*)socket;
   char msg[1024];

   while(1) {
       bzero(msg, sizeof(msg));
       recv(client_socket, msg, sizeof(msg), 0);
       printf("Message: %s\n", msg);
   }
}

int main() {
   int client_socket;
   struct sockaddr_in server_addr;
   pthread_t thread;
   char msg[1024];

   client_socket = socket(AF_INET, SOCK_STREAM, 0);
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(8080);
   server_addr.sin_addr.s_addr = INADDR_ANY;

   connect(client_socket, (struct sockaddr*)&server_addr,
sizeof(server_addr));
   printf("Connected to Server!\n");

   pthread_create(&thread, NULL, receive_msg, (void*)&client_socket);

   while(1) {
       bzero(msg, sizeof(msg));
       printf("You: ");
       fgets(msg, sizeof(msg), stdin);
       msg[strlen(msg) - 1] = '\0';

       send(client_socket, msg, sizeof(msg), 0);
```

```
        if(strcmp(msg, "exit") == 0) {
            printf("Disconnected from Server!\n");
            close(client_socket);
            break;
        }
    }
    return 0;
}
```

OUTPUT



8. Implement leaky bucket algorithm for congestion control.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUCKET_SIZE 10
#define OUTPUT_RATE 3

void leaky_bucket(int packets[], int n) {
   int bucket = 0;

   for (int i = 0; i < n; i++) {
       printf("\nIncoming packet size: %d", packets[i]);

       if (packets[i] > BUCKET_SIZE) {
           printf(" -> Packet discarded (too large!)\n");
           continue;
       }

       if (bucket + packets[i] > BUCKET_SIZE) {
           printf(" -> Packet discarded (Bucket Overflow!)\n");
       } else {
           bucket += packets[i];
           printf(" -> Packet added to bucket (Current bucket size:
%d)\n", bucket);
       }


       bucket -= OUTPUT_RATE;
       if (bucket < 0) bucket = 0;

       printf(" -> After leaking %d packets, bucket size: %d\n",
OUTPUT_RATE, bucket);
       sleep(1);
   }


   while (bucket > 0) {
       printf("\nLeaking %d packets...", OUTPUT_RATE);
       bucket -= OUTPUT_RATE;
```

```c
        if (bucket < 0) bucket = 0;
        printf(" Bucket size now: %d\n", bucket);
        sleep(1);
    }
    printf("\nBucket is empty.\n");
}

int main() {
    int packets[] = {4, 8, 15, 6, 3, 12};
    int n = sizeof(packets) / sizeof(packets[0]);

    printf("Leaky Bucket Algorithm Simulation\n");
    leaky_bucket(packets, n);

    return 0;
}
```

OUTPUT

```
devna@DEVNA: ~
devna@DEVNA:~$ nano lba.c
devna@DEVNA:~$ gcc lba.c -o lba
devna@DEVNA:~$ ./lba
Leaky Bucket Algorithm Simulation

Incoming packet size: 4 -> Packet added to bucket (Current bucket size: 4)
 -> After leaking 3 packets, bucket size: 1

Incoming packet size: 8 -> Packet added to bucket (Current bucket size: 9)
 -> After leaking 3 packets, bucket size: 6

Incoming packet size: 15 -> Packet discarded (too large!)

Incoming packet size: 6 -> Packet discarded (Bucket Overflow!)
 -> After leaking 3 packets, bucket size: 3

Incoming packet size: 3 -> Packet added to bucket (Current bucket size: 6)
 -> After leaking 3 packets, bucket size: 3

Incoming packet size: 12 -> Packet discarded (too large!)

Leaking 3 packets... Bucket size now: 0

Bucket is empty.
devna@DEVNA:~$
```

NETWORK LAYER-1,2,4,5

---

1. Use tools like ping and traceroute to explore various Internet paths to popular servers.

OUTPUT

```
devna@DEVNA: ~
devna@DEVNA:~$ ping google.com
PING google.com (142.250.182.46) 56(84) bytes of data.
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=1 ttl=117 time=158 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=2 ttl=117 time=176 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=3 ttl=117 time=215 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=4 ttl=117 time=30.3 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=5 ttl=117 time=21.3 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=6 ttl=117 time=166 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=7 ttl=117 time=21.5 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=8 ttl=117 time=226 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=9 ttl=117 time=39.8 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=10 ttl=117 time=155 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=11 ttl=117 time=20.8 ms
64 bytes from maa05s19-in-f14.1e100.net (142.250.182.46): icmp_seq=12 ttl=117 time=195 ms
^Z
[3]+  Stopped                 ping google.com
devna@DEVNA:~$ traceroute google.com
Command 'traceroute' not found, but can be installed with:
sudo apt install inetutils-traceroute  # version 2:2.4-3ubuntu1, or
sudo apt install traceroute            # version 1:2.1.5-1
devna@DEVNA:~$ sudo apt install inetutils.traceroute
[sudo] password for devna:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
devna@DEVNA:~$ traceroute google.com
traceroute to google.com (142.250.182.46), 64 hops max
  1   172.25.192.1  0.561ms  0.569ms  0.491ms
  2   192.168.1.1  2.757ms  2.310ms  94.938ms
  3   100.84.0.1  8.171ms  5.197ms  6.437ms
  4   103.153.93.61  4.503ms  4.758ms  4.591ms
  5   10.1.4.21  268.721ms  97.446ms  108.637ms
  6   72.14.205.178  197.811ms  101.904ms  204.646ms
  7   *  *  *
  8   142.251.55.30  19.014ms  18.658ms  167.239ms
  9   142.250.239.56  22.445ms  181.766ms  20.520ms
 10   142.250.182.46  22.697ms  135.465ms  19.441ms
devna@DEVNA:~$
```

2. Use web-based tools like the whois utility to query Internet registries, and understand which IP addresses are allocated to the student's network. Find out which are the major ISPs, and which is the ISP of the student's network.

```
devna@DEVNA:~$ whois 23.192.228.80

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2025, American Registry for Internet Numbers, Ltd.
#


NetRange:       23.192.0.0 - 23.223.255.255
CIDR:           23.192.0.0/11
NetName:        AKAMAI
NetHandle:      NET-23-192-0-0-1
Parent:         NET23 (NET-23-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Akamai Technologies, Inc. (AKAMAI)
RegDate:        2013-07-12
Updated:        2013-08-09
Ref:            https://rdap.arin.net/registry/ip/23.192.0.0



OrgName:        Akamai Technologies, Inc.
OrgId:          AKAMAI
Address:        145 Broadway
City:           Cambridge
StateProv:      MA
PostalCode:     02142
Country:        US
RegDate:        1999-01-21
Updated:        2023-10-24
Ref:            https://rdap.arin.net/registry/entity/AKAMAI


OrgTechHandle: SJS98-ARIN
OrgTechName:   Schecter, Steven Jay
OrgTechPhone:  +1-617-274-7134
OrgTechEmail:  ip-admin@akamai.com
OrgTechRef:    https://rdap.arin.net/registry/entity/SJS98-ARIN

OrgTechHandle: IPADM11-ARIN
OrgTechName:   ipadmin
OrgTechPhone:  +1-617-444-0017
OrgTechEmail:  ip-admin@akamai.com
OrgTechRef:    https://rdap.arin.net/registry/entity/IPADM11-ARIN
```

```
OrgTechHandle: SJS98-ARIN
OrgTechName:   Schecter, Steven Jay
OrgTechPhone:  +1-617-274-7134
OrgTechEmail:  ip-admin@akamai.com
OrgTechRef:    https://rdap.arin.net/registry/entity/SJS98-ARIN

OrgTechHandle: IPADM11-ARIN
OrgTechName:   ipadmin
OrgTechPhone:  +1-617-444-0017
OrgTechEmail:  ip-admin@akamai.com
OrgTechRef:    https://rdap.arin.net/registry/entity/IPADM11-ARIN

OrgAbuseHandle: NUS-ARIN
OrgAbuseName:   NOC United States
OrgAbusePhone:  +1-617-444-2535
OrgAbuseEmail:  abuse@akamai.com
OrgAbuseRef:    https://rdap.arin.net/registry/entity/NUS-ARIN


#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2025, American Registry for Internet Numbers, Ltd.
#

devna@DEVNA:~$
```

4. Implement Distance Vector Routing algorithm and Link State Routing algorithm.

DVR

```c
#include <stdio.h>

#define MAX_NODES 10
#define INF 9999

typedef struct {
   int distance[MAX_NODES];
   int next_hop[MAX_NODES];
} RoutingTable;

int main() {
   int n, i, j, k;
   int cost[MAX_NODES][MAX_NODES];
   RoutingTable rt[MAX_NODES];

   printf("Enter the number of routers: ");
   scanf("%d", &n);

   printf("Enter the cost adjacency matrix (9999 for no direct link):\n");
   for (i = 0; i < n; i++) {
       for (j = 0; j < n; j++) {
           scanf("%d", &cost[i][j]);
           if (cost[i][j] == 0 && i != j)
               cost[i][j] = INF;
       }
   }

   // Initialize routing tables
   for (i = 0; i < n; i++) {
       for (j = 0; j < n; j++) {
           rt[i].distance[j] = cost[i][j];
           rt[i].next_hop[j] = (cost[i][j] != INF && i != j) ? j : -1;
       }
   }

   // Distance Vector Routing Algorithm
   int updated;
   do {
       updated = 0;
```

```
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                for (k = 0; k < n; k++) {
                    if (rt[i].distance[k] + rt[k].distance[j] <
rt[i].distance[j]) {
                        rt[i].distance[j] = rt[i].distance[k] +
rt[k].distance[j];
                        rt[i].next_hop[j] = k;
                        updated = 1;
                    }
                }
            }
        }
    } while (updated);

    // Print final routing tables
    for (i = 0; i < n; i++) {
        printf("\nRouting table for Router %d:\n", i);
        printf("Destination\tNext Hop\tDistance\n");
        for (j = 0; j < n; j++) {
            printf("%d\t\t%d\t\t%d\n", j, rt[i].next_hop[j],
rt[i].distance[j]);
        }
    }

    return 0;
}
```

OUTPUT

```
devna@DEVNA: ~
devna@DEVNA:~$ nano dvr.c
devna@DEVNA:~$ gcc dvr.c -o dvr
devna@DEVNA:~$ ./dvr
Enter the number of routers: 3
Enter the cost adjacency matrix (9999 for no direct link):
0
2
7
2
0
1
7
0
1

Routing table for Router 0:
Destination     Next Hop        Distance
0               -1              0
1               1               2
2               1               3

Routing table for Router 1:
Destination     Next Hop        Distance
0               0               2
1               -1              0
2               2               1

Routing table for Router 2:
Destination     Next Hop        Distance
0               0               7
1               0               9
2               -1              1
devna@DEVNA:~$
```

LSR

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

#define MAX_NODES 10
#define INF 9999

void dijkstra(int graph[MAX_NODES][MAX_NODES], int n, int start) {
   int distance[MAX_NODES], visited[MAX_NODES], parent[MAX_NODES];
   int i, j, min, next_node;

   // Initialize distances and visited nodes
   for (i = 0; i < n; i++) {
       distance[i] = INF;
       visited[i] = 0;
       parent[i] = -1;
```

```
    }
    distance[start] = 0;

    for (i = 0; i < n - 1; i++) {
        min = INF;
        next_node = -1;

        // Find the node with the minimum distance
        for (j = 0; j < n; j++) {
            if (!visited[j] && distance[j] < min) {
                min = distance[j];
                next_node = j;
            }
        }

        if (next_node == -1) break; // No more reachable nodes
        visited[next_node] = 1;

        // Update distances
        for (j = 0; j < n; j++) {
            if (!visited[j] && graph[next_node][j] != INF &&
                distance[next_node] + graph[next_node][j] < distance[j]) {
                distance[j] = distance[next_node] + graph[next_node][j];
                parent[j] = next_node;
            }
        }
    }

    // Print shortest paths
    printf("\nShortest paths from Router %d:\n", start);
    for (i = 0; i < n; i++) {
        printf("To %d: Distance = %d, Path = %d", i, distance[i], i);
        j = i;
        while (parent[j] != -1) {
            printf(" <- %d", parent[j]);
            j = parent[j];
        }
        printf("\n");
    }
}
```

```c
int main() {
    int n, i, j, start;
    int graph[MAX_NODES][MAX_NODES];

    printf("Enter the number of routers: ");
    scanf("%d", &n);

    printf("Enter the cost adjacency matrix (9999 for no direct link):\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
            if (graph[i][j] == 0 && i != j)
                graph[i][j] = INF;
        }
    }

    printf("Enter the starting router: ");
    scanf("%d", &start);

    dijkstra(graph, n, start);

    return 0;
}
```

OUTPUT

```
devna@DEVNA: ~
devna@DEVNA:~$ nano lsr.c
devna@DEVNA:~$ gcc lsr.c -o lsr
devna@DEVNA:~$ ./lsr
Enter the number of routers: 3
Enter the cost adjacency matrix (9999 for no direct link):
1 2 3
0 2 1
2 0 4
Enter the starting router: 1

Shortest paths from Router 1:
To 0: Distance = 3, Path = 0 <- 2 <- 1
To 1: Distance = 0, Path = 1
To 2: Distance = 1, Path = 2 <- 1
devna@DEVNA:~$
```