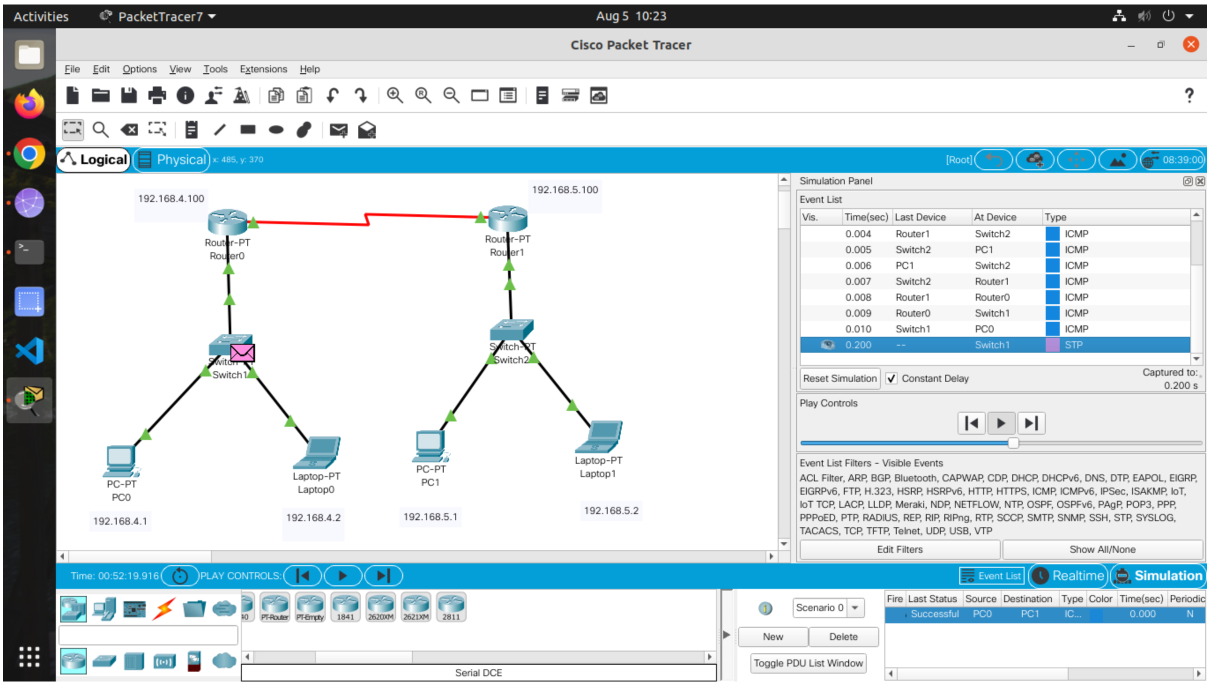
Name: *Tejas Padhiyar*

Roll No.: *31451*

Batch: *M4*

Class: *TE4*

Assignment No. 1 (A2)



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.004	Router1	Switch2	ICMP
	0.005	Switch2	PC1	ICMP
	0.006	PC1	Switch2	ICMP
	0.007	Switch2	Router1	ICMP
	0.008	Router1	Router0	ICMP
	0.009	Router0	Switch1	ICMP
	0.010	Switch1	PC0	ICMP
	0.200	--	Switch1	STP

Reset Simulation ☑ Constant Delay

Captured to: 0.200 s

Play Controls

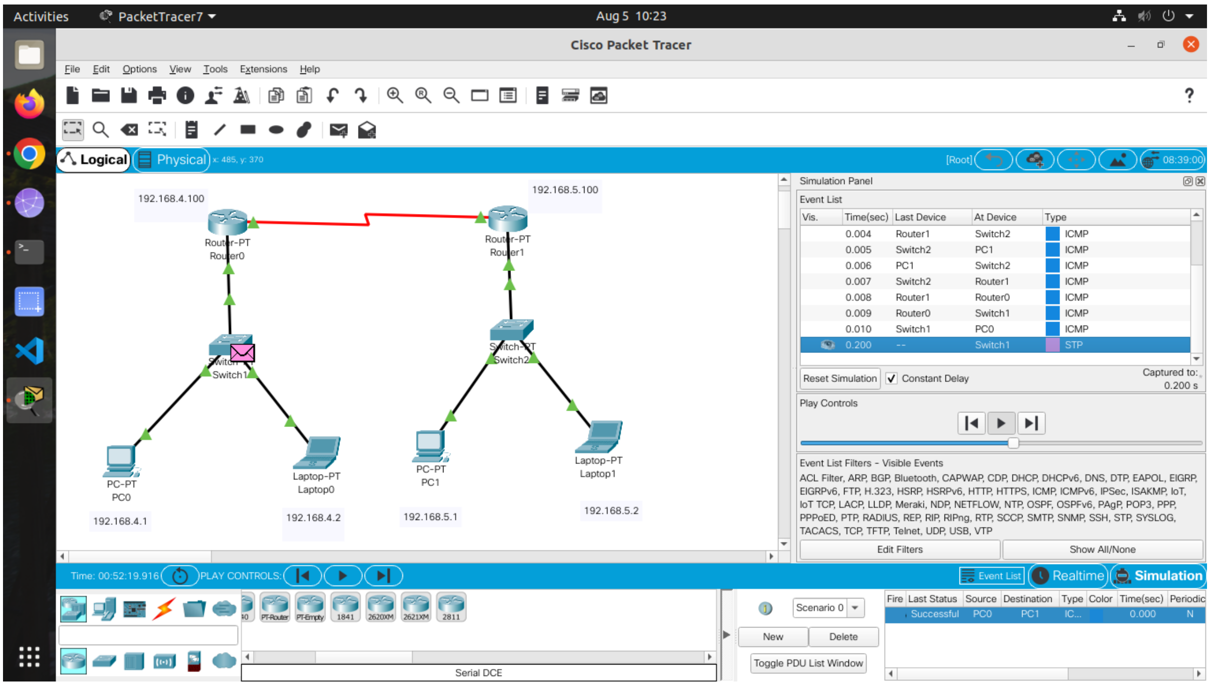
Event List Filters - Visible Events

ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters | Show All/None

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
	Successful	PC0	PC1	IC...		0.000	N

Assignment No. 2 (A2)



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.004	Router1	Switch2	ICMP
	0.005	Switch2	PC1	ICMP
	0.006	PC1	Switch2	ICMP
	0.007	Switch2	Router1	ICMP
	0.008	Router1	Router0	ICMP
	0.009	Router0	Switch1	ICMP
	0.010	Switch1	PC0	ICMP
	0.200	--	Switch1	STP

Reset Simulation ☑ Constant Delay

Captured to: 0.200 s

Play Controls

Event List Filters - Visible Events

ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters | Show All/None

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
	Successful	PC0	PC1	IC...		0.000	N

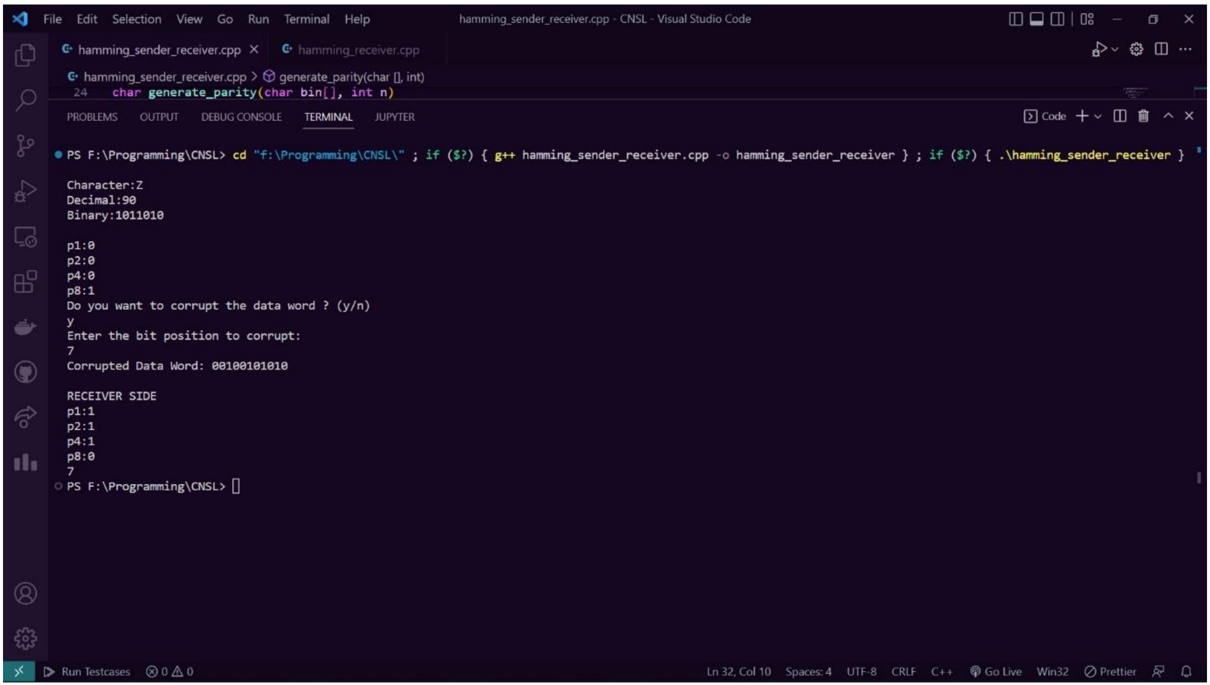
Assignment No. 3 (A3)

```
#include <bits/stdc++.h>
using namespace std;
char generate_parity_1(string bin, int n)
{
    int counter = 0;
    for (int i = 0; i < n; i++)
    {
        if (bin[i] == '1')
        {
            counter++;
        }
    }
    if (counter % 2 == 0)
        return '0';
    else
        return '1';
}

char generate_parity(char bin[], int n)
{
    int counter = 0;
    for (int i = 0; i < n; i++)
    {
        if (bin[i] == '1')
            counter++;
    }
    if (counter % 2 == 0)
        return '0';
    else
        return '1';
}

void reverse(char str[], int length)
{
    int start = 0;
    int end = length -1;
    while (start < end)
    {
        swap(*(str+start), *(str+end));
        start++;
        end--;
    }
}

char* itoa(int num, char* str, int base)
{
    int i = 0;
    bool isNegative = false;
    /* Handle 0 explicitly, otherwise empty string is printed for 0 */
    if (num == 0)
    {
        str[i++] = '0';
        str[i] = '\0';
        return str;
    }
    // In standard itoa(), negative numbers are handled only with
    // base 10. Otherwise numbers are considered unsigned.
    if (num < 0 && base == 10)
    {
        isNegative = true;
        num = -num;
    }
    // Process individual digits
    while (num != 0)
    {
        int rem = num % base;
        str[i++] = (rem > 9)? (rem-10) + 'a' : rem + '0';
        num = num/base;
    }
    // If number is negative, append '-'
    if (isNegative)
        str[i++] = '-';
    str[i] = '\0'; // Append string terminator
    // Reverse the string
    reverse(str, i);
    return str;
}
int main()
{
    string s;
    cout << "Enter a String :" << endl;
    cin >> s;
    char p1, p2, p4, p8, hamming[11], p1_a[5], p2_a[5], p4_a[3], p8_a[3];
    for (int i = 0; i < s.length(); i++)
    {
        int c_ascii = int(s[i]);
        char bin[7];
        char bin_cpy[7];
        cout << "Character:" << s[i] << endl;
        cout << "Decimal:" << c_ascii << endl;
        cout << "Binary:";
        itoa(c_ascii, bin, 2);
        // bin_cpy[0]='0';
        int counter = 0;
        for (int i = 0; i <= 7; i++)
        {
            bin_cpy[i] = bin[counter];
            counter++;
        }
        for (int i = 0; i < 7; i++)
        {
            cout << bin_cpy[i];
        }
        cout << endl;
        for (int i = 0; i < 5; i++)
        {
            if (i == 0)
            {
                p1_a[i] = bin_cpy[0];
                p2_a[i] = bin_cpy[0];
                p4_a[i] = bin_cpy[1];
                p8_a[i] = bin_cpy[4];
            }
            else if (i == 1)
            {
                p1_a[i] = bin_cpy[1];
                p2_a[i] = bin_cpy[2];
                p4_a[i] = bin_cpy[2];
                p8_a[i] = bin_cpy[5];
            }
            else if (i == 2)
            {
                p1_a[i] = bin_cpy[3];
                p2_a[i] = bin_cpy[3];
                p4_a[i] = bin_cpy[3];
                p8_a[i] = bin_cpy[6];
            }
            else if (i == 3)
            {
                p1_a[i] = bin_cpy[4];
                p2_a[i] = bin_cpy[5];
            }
            else
            {
                p1_a[i] = bin_cpy[6];
                p2_a[i] = bin_cpy[6];
            }
        }
        p1 = generate_parity(p1_a, 5);
        p2 = generate_parity(p2_a, 5);
        p4 = generate_parity(p4_a, 3);
        p8 = generate_parity(p8_a, 3);

        hamming[0] = p1;
        hamming[1] = p2;
        int counter1 = 0;
        for (int i = 2; i < 11; i++)
        {
            if (i == 3)
                hamming[i] = p4;
            else if (i == 7)
                hamming[i] = p8;
            else
            {
                hamming[i] = bin_cpy[counter1];
                counter1++;
            }
        }
        cout << endl;
        cout << "p1:" << p1 << endl;
        cout << "p2:" << p2 << endl;
        cout << "p4:" << p4 << endl;
        cout << "p8:" << p8 << endl;
        cout << "Do you want to corrupt the data word ? (y/n)" << endl;
        char ch;
        bool flag;
        cin >> ch;
        if (ch == 'y')
        {
            flag = true;
            cout << "Enter the bit position to corrupt:" << endl;
            int pos;
            cin >> pos;
            if (hamming[pos - 1] == '1')
            {
                hamming[pos - 1] = '0';
            }
            else
            {
                hamming[pos - 1] = '1';
            }
```

```
        cout << "Corrupted Code Word: ";
        for (int i = 0; i < 11; i++)
        {
            cout << hamming[i];
        }
        cout << endl;
    }
    else
    {
        flag = false;
        cout << "Uncorrupted Code Word:";
        for (int i = 0; i < 11; i++)
        {
            cout << hamming[i];
        }
    }
    cout << endl;
    cout << "RECEIVER SIDE" << endl;
    string p1_check = "";
    string p2_check = "";
    string p4_check = "";
    string p8_check = "";
    p1_check = p1_check + hamming[0] + hamming[2] + hamming[4] + hamming[6] + hamming[8] +
hamming[10]; // 0 2 4 6 8 10
    p2_check = p2_check + hamming[1] + hamming[2] + hamming[5] + hamming[6] + hamming[9] +
hamming[10]; // 1 2 5 6 9 10
    p4_check = p4_check + hamming[3] + hamming[4] + hamming[5] + hamming[6]; // 3 4 5 6
    p8_check = p8_check + hamming[7] + hamming[8] + hamming[9] + hamming[10]; // 7 8 9 10
    char p1_rec = generate_parity_1(p1_check, 6);
    char p2_rec = generate_parity_1(p2_check, 6);
    char p4_rec = generate_parity_1(p4_check, 4);
    char p8_rec = generate_parity_1(p8_check, 4);

    cout << "p1:" << p1_rec << endl;
    cout << "p2:" << p2_rec << endl;
    cout << "p4:" << p4_rec << endl;
    cout << "p8:" << p8_rec << endl;

    string pos = "";
    pos = pos + p8_rec + p4_rec + p2_rec + p1_rec;

    int pos_no = stoi(pos, nullptr, 2);
    cout << pos_no << endl;

    if(flag == true)
    {
        if (hamming[pos_no - 1] == '1')
        {
            hamming[pos_no - 1] = '0';
        }
        else
        {
            hamming[pos_no - 1] = '1';
        }
        cout<<"Corrected Code Word is : ";
        for(int k = 0; k <= 11; k++)
        {
            cout<<hamming[k];
        }
    }
```

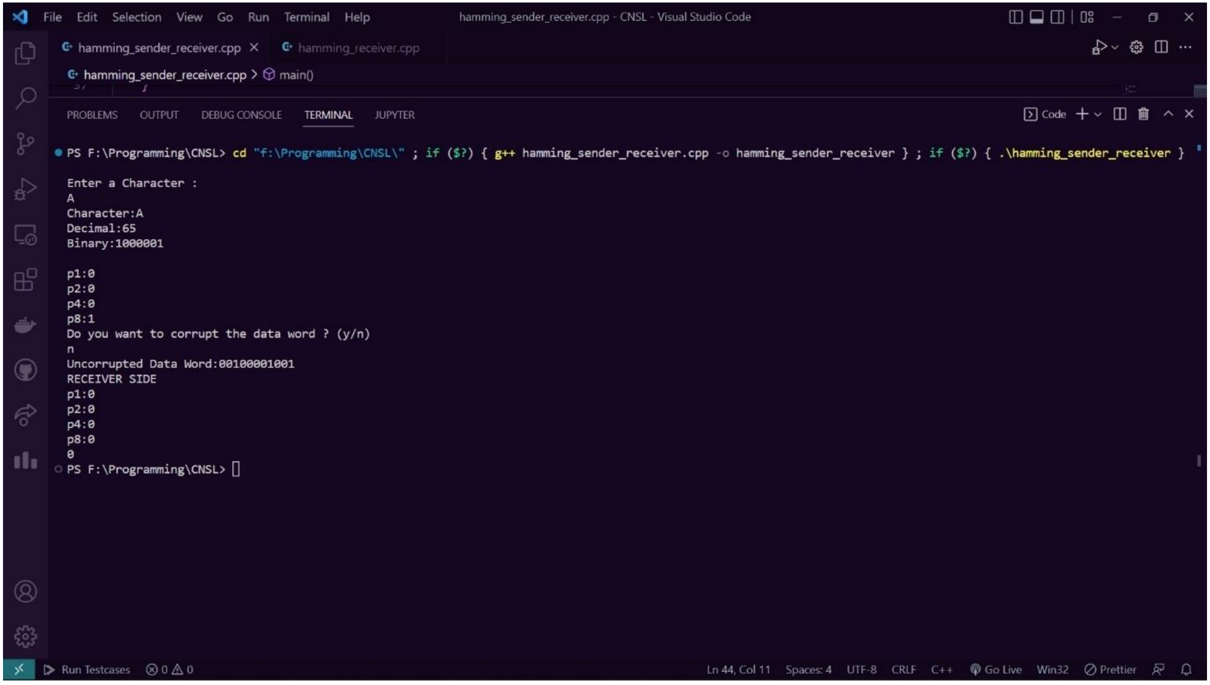
```
        cout<<endl<<"Data Word : ";
        for(int k = 0; k <= i; k++)
        {
            cout<<s[k];
        }
        cout<<endl<<"========================="<<endl;
    }

    return 0;
}
```

Output:


File Edit Selection View Go Run Terminal Help
hamming_sender_receiver.cpp - CNSL - Visual Studio Code
hamming_sender_receiver.cpp
hamming_receiver.cpp
hamming_sender_receiver.cpp > generate_parity(char [], int)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Code
Run Testcases ⊗ 0 ⚠ 0
Ln 32, Col 10 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

```
24    char generate_parity(char bin[], int n)
```

```
PS F:\Programming\CNSL> cd "f:\Programming\CNSL\" ; if ($?) { g++ hamming_sender_receiver.cpp -o hamming_sender_receiver } ; if ($?) { .\hamming_sender_receiver }

Character:Z
Decimal:90
Binary:1011010

p1:0
p2:0
p4:0
p8:1
Do you want to corrupt the data word ? (y/n)
y
Enter the bit position to corrupt:
7
Corrupted Data Word: 00100101010

RECEIVER SIDE
p1:1
p2:1
p4:1
p8:0
7
PS F:\Programming\CNSL> 
```


File Edit Selection View Go Run Terminal Help
hamming_sender_receiver.cpp - CNSL - Visual Studio Code
hamming_sender_receiver.cpp
hamming_receiver.cpp
hamming_sender_receiver.cpp > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Code
Run Testcases ⊗ 0 ⚠ 0
Ln 44, Col 11 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

```
PS F:\Programming\CNSL> cd "f:\Programming\CNSL\" ; if ($?) { g++ hamming_sender_receiver.cpp -o hamming_sender_receiver } ; if ($?) { .\hamming_sender_receiver }

Enter a Character :
A
Character:A
Decimal:65
Binary:1000001

p1:0
p2:0
p4:0
p8:1
Do you want to corrupt the data word ? (y/n)
n
Uncorrupted Data Word:00100001001
RECEIVER SIDE
p1:0
p2:0
p4:0
p8:0
0
PS F:\Programming\CNSL> 
```

Assignment No. 4 (A4)

Server.cpp

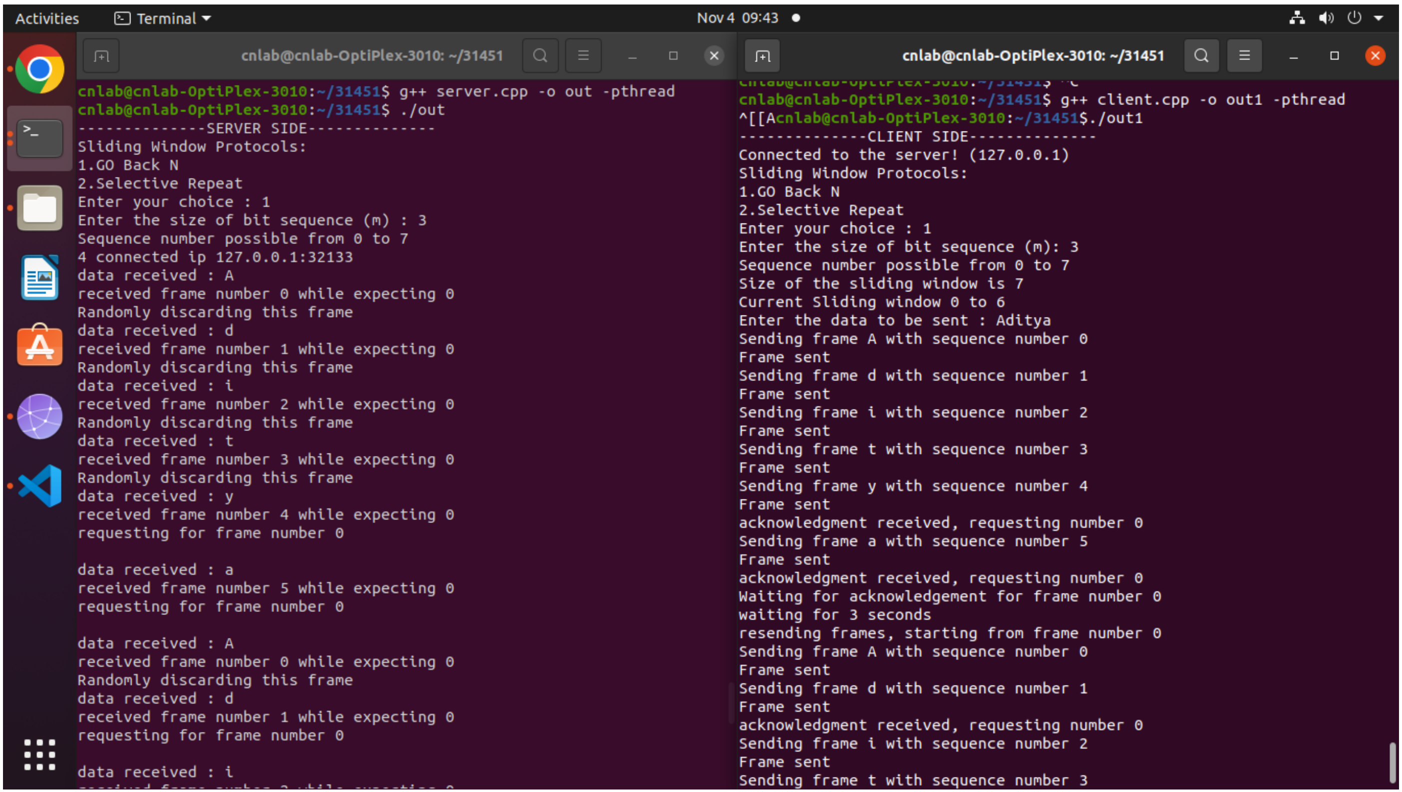
```
#include <bits/stdc++.h>
#include <sys/socket.h>
#include <cstring>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <thread>
#include <chrono>
#include <iostream>

using namespace std;

void gbn(int);
void sr(int);

int m;
int min_seq_num = 0;
int max_seq_num;

int current_sequence_number = min_seq_num;
int acknowledgement_remaining = min_seq_num;
int size_of_sliding_window;
int maximum_sequence_number;

struct msg {
        char data;
        int sequence_number;
};

struct rmsg {
    bool isAck;
    int sequence_number;
};

int main() {
    cout<<"--------------SERVER SIDE--------------"<<endl;
    int sfd,cfd;
    sfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sfd == -1) {
        cout << "socket not created" << endl;;
        exit(1);
    }
    struct sockaddr_in my_addr,peer_addr;
    memset(&my_addr, 0, sizeof(struct sockaddr_in));
    my_addr.sin_family = AF_INET;
    my_addr.sin_port = htons(8080);
    inet_aton("0.0.0.0",&my_addr.sin_addr);
    if (bind(sfd, (struct sockaddr *) &my_addr, sizeof(struct sockaddr_in)) == -1) {
        cout << "error in binding" << endl;
        exit(1);
    }
    if (listen(sfd,50) == -1) {
        cout << "error in listening" << endl;
        exit(1);
    }
    socklen_t peer_addr_size;
    peer_addr_size = sizeof(struct sockaddr_in);
    int choice = 0;
    cout << "Sliding Window Protocols: \n1.GO Back N\n2.Selective Repeat"<<endl;
        cout << "Enter your choice : ";
        cin >> choice;
        if (choice == 1) {
        cout << "Enter the size of bit sequence (m) : " ;
            cin >> m;
        max_seq_num = pow(2,m) - 1;
        cout << "Sequence number possible from " << min_seq_num << " to " << max_seq_num << endl;
        while (true) {
            cfd = accept(sfd, (struct sockaddr *) &peer_addr, &peer_addr_size);
            cout << cfd <<  " connected ip " << inet_ntoa(peer_addr.sin_addr) << ":" << peer_addr.sin_port <<
endl;
                    gbn(cfd);
        }

        }
        else if (choice == 2) {
        cout << "Enter the size of bit sequence (m) : ";
        cin >> m;
        max_seq_num = pow(2,m) - 1;
        size_of_sliding_window = pow(2, m - 1);
            maximum_sequence_number = current_sequence_number + size_of_sliding_window - 1;
        cout << "Sequence number possible from " << min_seq_num << " to " << max_seq_num << endl;
        cout << "Receiver sliding window size : " << size_of_sliding_window << endl;
        cout << "From " << min_seq_num << " to " << maximum_sequence_number << endl;
        while (true) {
            cfd = accept(sfd, (struct sockaddr *) &peer_addr, &peer_addr_size);
            cout << cfd <<  " connected ip " << inet_ntoa(peer_addr.sin_addr) << ":" << peer_addr.sin_port <<
endl;
                    sr(cfd);
        }
        }

    return 0;

}

void gbn(int cfd) {
    int expected_sequence_number = min_seq_num;
        std::random_device dev;

        std::mt19937 rng(dev());
        std::uniform_int_distribution<std::mt19937::result_type> distBin(0,1);
    while (true) {
        msg m1;
        int received_sequence_number;
        int res = recv(cfd, &m1, sizeof(m1), 0);
        if (res == 0) {
            break;
            return;
        }
        received_sequence_number = m1.sequence_number;
        cout << "data received : " << m1.data << endl;
        cout << "received frame number " << received_sequence_number << " while expecting " <<
expected_sequence_number << endl;
        if (distBin(rng)) {
            cout << "Randomly discarding this frame" << endl;
            continue;
        }
        if (received_sequence_number == expected_sequence_number) {
            cout << "frame received correctly" << endl;
            expected_sequence_number = (expected_sequence_number + 1) % (max_seq_num + 1);
        };
        cout << "requesting for frame number " << expected_sequence_number << endl;
        cout<<endl;
        // this_thread::sleep_for(chrono::seconds(2));
        send(cfd, &expected_sequence_number, sizeof(int), 0);
    }
}

void sr(int cfd) {
    vector<pair<int,bool>> backlog;
    bool nakSent = false;
    for (int i = min_seq_num; i <= maximum_sequence_number; i++) {
        backlog.push_back({i,false});
    }
    std::random_device dev;
        std::mt19937 rng(dev());
        std::uniform_int_distribution<std::mt19937::result_type> distBin(0,1);
    while (true) {
        msg m1;
        int received_sequence_number;
        for (int i = 0; i < backlog.size(); i++) {
            cout << backlog[i].first << " ";
        }
        cout << endl;

        int res = recv(cfd, &m1, sizeof(m1), 0);
        if (res == 0) {
            break;
            return;
        }
        received_sequence_number = m1.sequence_number;
        cout << "data received : " << m1.data << endl;
        cout << "received frame number " << received_sequence_number << " while expecting " <<
backlog.begin()->first << endl;
        if (distBin(rng)) {
            cout << "Randomly discarding this frame" << endl;
            continue;
        }
        for (int i = 0; i < backlog.size(); i++) {
            if (backlog[i].first == received_sequence_number) {
                backlog[i].second = true;
                break;
            }
        }
        if (!nakSent && !backlog.begin()->second) {
            cout << "sending Negative Acknowledgement" << endl;
            int nak = backlog.begin()->first;
            rmsg m1;
            m1.sequence_number = nak;
            m1.isAck = false;
            send(cfd, &m1, sizeof(m1), 0);
            nakSent = true;
        }
        int i = 0;
        while (backlog[i].second) {
            i = (i + 1) % (max_seq_num+1);
        }
        if (backlog[((i-1) + (max_seq_num + 1)) % (max_seq_num+1)].first == received_sequence_number) {
            cout << "sending Acknowledgement" << endl;
            int ack = (backlog[((i-1) + (max_seq_num + 1)) % (max_seq_num+1)].first + 1) % (max_seq_num
+ 1);
            rmsg m1;
            m1.sequence_number = ack;
            m1.isAck = true;
            send(cfd, &m1, sizeof(m1), 0);
            for (int i = 0; i < backlog.size(); i++) {
                if (backlog[i].second) {
                    backlog.erase(backlog.begin() + i);
                    backlog.push_back({(backlog.back().first+1) % (max_seq_num + 1), false});
                    i--;
                }
            }
            cout<<endl;
        }
    }
}
```

Client.cpp

```
#include <bits/stdc++.h>
#include <sys/socket.h>
#include <cstring>
#include <netinet/in.h>
#include <netinet/ip.h> /* superset of previous */
#include <arpa/inet.h>
#include <chrono>
#include <thread>
#include <iostream>

using namespace std;

void gbn(int);
void sr(int);

struct msg {
    char data;
    int sequence_number;
};

struct rmsg {
    bool isAck;
    int sequence_number;
};

int main() {
    cout<<"--------------CLIENT SIDE--------------"<<endl;
    int cfd;
    cfd = socket(AF_INET,SOCK_STREAM, 0);
    if (cfd == -1) {
        cout << "socket not created" << endl;;
        exit(1);
    }

    struct sockaddr_in my_addr,peer_addr;

    memset(&my_addr, 0, sizeof(struct sockaddr_in));
    my_addr.sin_family = AF_INET;
    my_addr.sin_port = htons(0);
    inet_aton("127.0.0.1",&my_addr.sin_addr);
    memset(&peer_addr, 0, sizeof(struct sockaddr_in));
    peer_addr.sin_family = AF_INET;
    peer_addr.sin_port = htons(8080);
    inet_aton("127.0.0.1", &peer_addr.sin_addr);

    // my_addr.sin_addr=(in_addr)INADDR_LOOPBACK;
    if (bind(cfd, (struct sockaddr *) &my_addr, sizeof(struct sockaddr_in)) == -1) {
        cout << "error in binding" << endl;
        exit(errno);
    }

    if (connect(cfd, (struct sockaddr *) &peer_addr, sizeof(peer_addr)) == -1) {
        cout << "error in connecting" << endl;
        exit(errno);
    }

    cout << "Connected to the server! (127.0.0.1)" << endl;
    int choice = 0;
    cout << "Sliding Window Protocols: \n1.GO Back N\n2.Selective Repeat"<<endl;
    cout << "Enter your choice : ";
    cin >> choice;
    if (choice == 1) {
        gbn(cfd);
    }
    else if (choice == 2) {
        sr(cfd);
    }

    return 0;
}

int m;
int min_seq_num = 0;
int max_seq_num;
int current_sequence_number = min_seq_num;
int acknowledgement_remaining = min_seq_num;
int size_of_sliding_window;
int maximum_sequence_number;
int acknowledgedDataIndex = -1;
mutex m1;

void sendFramesGBN(int cfd, string data) {
    unique_lock<mutex> l(m1,defer_lock);
    int dataIndex = 0;
    bool flagDataSent = false;
    while (true) {
        for (int i = 0; i < size_of_sliding_window - 1 ; i++) {
            if (dataIndex >= data.size()) {
                flagDataSent = true;
                break;
            }
            cout << "Sending frame " << data[dataIndex] << " with sequence number " <<
current_sequence_number << endl;
            l.lock();
            msg m1 = msg {data[dataIndex++], current_sequence_number};
            send(cfd, &m1, sizeof(m1), 0);
            current_sequence_number = (current_sequence_number + 1) % (size_of_sliding_window + 1);
            l.unlock();
            cout << "Frame sent" << endl;
            this_thread::sleep_for(chrono::seconds(1));
        }
        l.lock();
        if (acknowledgement_remaining != current_sequence_number) {
            cout << "Waiting for acknowledgement for frame number " << acknowledgement_remaining <<
endl;
            cout << "waiting for 3 seconds" << endl;
            this_thread::sleep_for(chrono::seconds(3));
            if (acknowledgement_remaining != current_sequence_number) {
                cout << "resending frames, starting from frame number " << acknowledgement_remaining <<
endl;
                dataIndex = acknowledgedDataIndex+1;
                flagDataSent = false;
                current_sequence_number = acknowledgement_remaining;
            }
        } else if (flagDataSent) {
            // close(cfd);
            exit(0);
        }
        l.unlock();
    }
    cout<<endl;
}

bool check(int a, int b, int c) {
    if (a < b) {
        if (a < c && c < b) return true;
        else return false;
    } else {
         if (b < c && c < a) return false;
      else return true;
    }
}

void recvAcksGBN(int cfd) {
    unique_lock<mutex> l(m1,defer_lock);
    struct timeval tv;
    fd_set cfds;
    FD_ZERO(&cfds);
    FD_SET(cfd, &cfds);
    tv.tv_sec = 1;

    while (true) {
        int ack;
        recv(cfd, &ack, sizeof(int), 0);
        cout << "acknowledgment received, requesting number " << ack << endl;
        //cout << "->" << min_seq_num << " " << maximum_sequence_number << endl;
        if (check(min_seq_num, maximum_sequence_number, ack)) {
            l.lock();
            acknowledgedDataIndex++;
            int number_of_frames_acknowledged = abs(ack - acknowledgement_remaining) %
(size_of_sliding_window-1);
            acknowledgement_remaining = ack;
            //cout << "acknowledgement_remaining changed to " << acknowledgement_remaining << endl;
            min_seq_num = ack;
            maximum_sequence_number = (maximum_sequence_number +
number_of_frames_acknowledged) % (size_of_sliding_window + 1);
            l.unlock();
        }
    }
    cout<<endl;
}
```

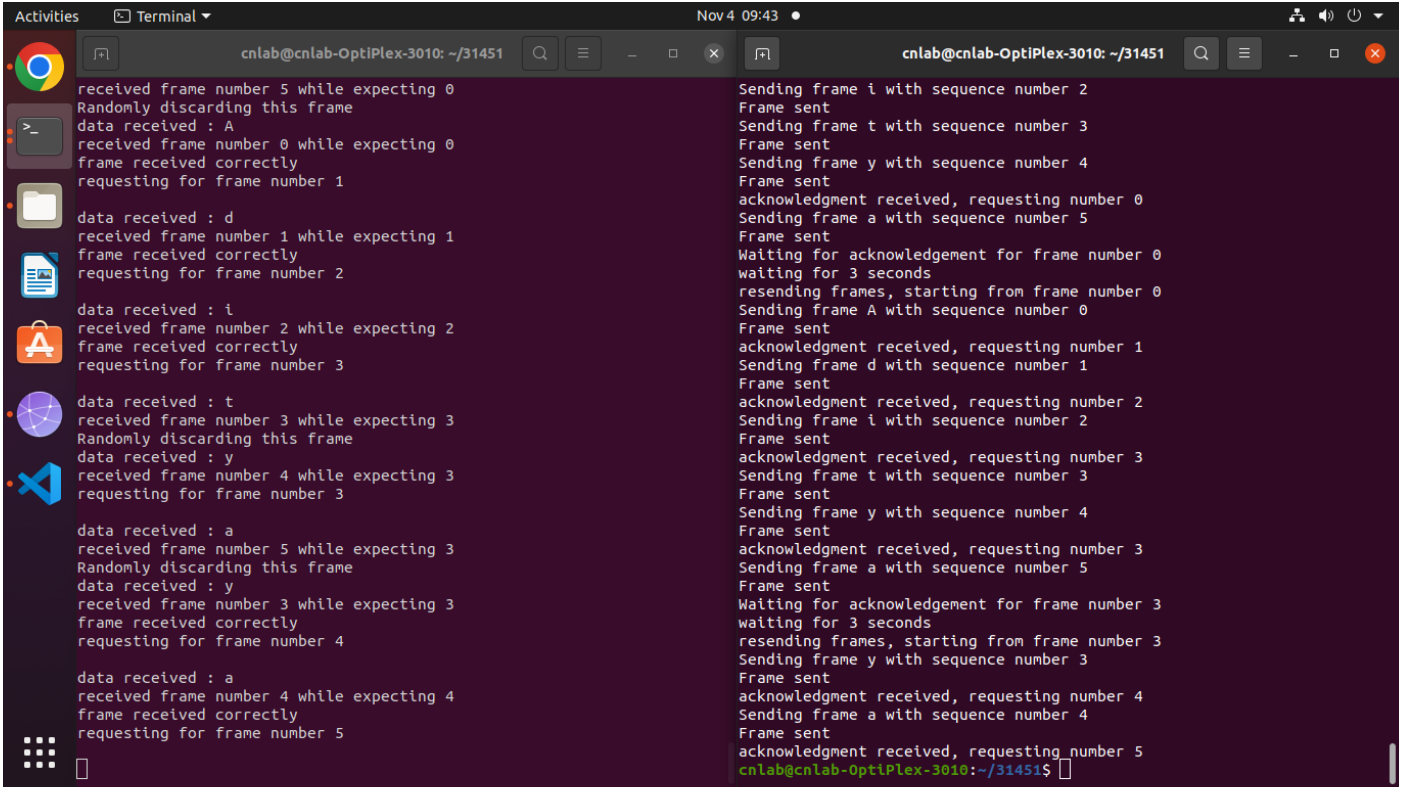
```
void gbn(int cfd) {
    cout << "Enter the size of bit sequence (m): " ;
    cin >> m;
    max_seq_num = pow(2,m) - 1;
    size_of_sliding_window = pow(2, m) - 1;
    maximum_sequence_number = current_sequence_number + size_of_sliding_window - 1;
    cout << "Sequence number possible from " << min_seq_num << " to " << max_seq_num << endl;
    cout << "Size of the sliding window is " << size_of_sliding_window << endl;
    cout << "Current Sliding window " << acknowledgement_remaining << " to " <<
maximum_sequence_number << endl;
    cout << "Enter the data to be sent : ";
    string data;
    cin >> data;
    std::random_device dev;

    std::mt19937 rng(dev());
    std::uniform_int_distribution<std::mt19937::result_type> distBin(0,1);
    // std::cout << distBin(rng) << std::endl;
    thread t2(sendFramesGBN, cfd, data);
    thread t1(recvAcksGBN, cfd);
    t2.join();
    t1.join();
}

vector<pair<int,bool>> receivedAcknowledgments;
vector<char> chars;
int dataIndex = 0;
int dataIndexTemp = 0;

void sendFramesSR(int cfd, string data) {
    unique_lock<mutex> l(m1,defer_lock);
    bool flagDataSent = false;
    while (true) {
        for (int i = 0; i < size_of_sliding_window ; i++) {
            if (!receivedAcknowledgments[i].second) {
                cout << "Sending frame " << chars[current_sequence_number] << " with sequence number " <<
current_sequence_number << endl;
                l.lock();
                msg m1 = msg {chars[current_sequence_number], current_sequence_number};
                send(cfd, &m1, sizeof(m1), 0);
                current_sequence_number = (current_sequence_number + 1) % (max_seq_num + 1);
                l.unlock();
                cout << "Frame sent" << endl;
                this_thread::sleep_for(chrono::seconds(1));
                if (dataIndexTemp == data.size()+size_of_sliding_window) {
                    flagDataSent = true;
                    break;
                }
            }
        }

        if (flagDataSent) {
            exit(0);
        }
        bool allNotReceived = true;
        for (int i = 0; i < receivedAcknowledgments.size(); i++) {
            if (receivedAcknowledgments[i].second) {
                allNotReceived = false;
                break;
            }
        }
        l.lock();
        if (!allNotReceived) {
            for (int i = 0; i < receivedAcknowledgments.size(); i++) {
                if (!receivedAcknowledgments[i].second) {
                    cout << "Waiting for acknowledgement for frame having sequence number " <<
receivedAcknowledgments[i].first << endl;
                    cout << "waiting for 3 seconds" << endl;
                    this_thread::sleep_for(chrono::seconds(3));
                    if (!receivedAcknowledgments[i].second) {
                        cout << "resending frames, starting from frame number " <<
receivedAcknowledgments[i].first << endl;
                        dataIndex = acknowledgedDataIndex+1;
                        flagDataSent = false;
                        current_sequence_number = receivedAcknowledgments[i].first;
                        break;
                    }
                }
            }
        }
        cout<<endl;
        l.unlock();

    }
}

void recvAcksSR(int cfd, string data) {
    unique_lock<mutex> l(m1,defer_lock);
    struct timeval tv;
    fd_set cfds;
    FD_ZERO(&cfds);
    FD_SET(cfd, &cfds);
    tv.tv_sec = 1;

    while (true) {
        rmsg ack;
        recv(cfd, &ack, sizeof(ack), 0);
        if (ack.isAck) {
            cout << "Acknowledgement Received " << ack.sequence_number << endl;
            l.lock();
            for (int i = 0; i < receivedAcknowledgments.size(); i++) {
                if (receivedAcknowledgments[i].first == ack.sequence_number) {
                    break;
                }
                receivedAcknowledgments.erase(receivedAcknowledgments.begin() + i);
                maximum_sequence_number = (maximum_sequence_number + 1) % (max_seq_num + 1);
                if (dataIndex+1 <= data.size()) {
                    chars[maximum_sequence_number] = data[dataIndex++];
                }
                dataIndexTemp++;
                if (dataIndex == data.size()+size_of_sliding_window) {
                    dataIndex = dataIndex-1;
                }
                receivedAcknowledgments.push_back({maximum_sequence_number, false});
                acknowledgedDataIndex++;
                i--;
            }
            cout << "sliding window shifted : " << endl;
            for (int i = 0; i < receivedAcknowledgments.size(); i++) {
                cout << receivedAcknowledgments[i].first << " ";
            }
            cout << endl;
            l.unlock();
        } else {
            cout << "Negaive Acknowledgement Received" << endl;
            l.lock();
            for (int i = 0; i < receivedAcknowledgments.size(); i++) {
                if (ack.sequence_number == receivedAcknowledgments[i].first) {
                    receivedAcknowledgments[i].second = false;
                }
            }
            l.unlock();
        }
    }
    cout<<endl;
}
void sr(int cfd) {
    cout << "Enter the size of bit sequence (m): ";
    cin >> m;
    max_seq_num = pow(2, m) - 1;
    size_of_sliding_window = pow(2, m - 1);
    maximum_sequence_number = current_sequence_number + size_of_sliding_window - 1;
    cout << "Sequence number possible from " << min_seq_num << " to " << max_seq_num << endl;
    cout << "Size of the sliding window is " << size_of_sliding_window << endl;
    cout << "Current Sliding window " << acknowledgement_remaining << " to " <<
maximum_sequence_number << endl;
    cout << "Enter the data you want to send : ";
    string data;
    vector<char> temp(max_seq_num + 1);
    chars = temp;
    cin >> data;
    dataIndex = 0;
    for (int i = min_seq_num; i <= maximum_sequence_number; i++) {
        chars[i] = data[dataIndex++];
        dataIndexTemp++;
        receivedAcknowledgments.push_back({i,false});
    }
    thread t2(sendFramesSR, cfd, data);
    thread t1(recvAcksSR, cfd, data);
    t2.join();
    t1.join();
}
```

Output:


Activities
Terminal
Nov 4 09:43
cnlab@cnlab-OptiPlex-3010: ~/31451
cnlab@cnlab-OptiPlex-3010: ~/31451

```
cnlab@cnlab-OptiPlex-3010:~/31451$ g++ server.cpp -o out -pthread
cnlab@cnlab-OptiPlex-3010:~/31451$ ./out
--------------SERVER SIDE--------------
Sliding Window Protocols:
1.GO Back N
2.Selective Repeat
Enter your choice : 1
Enter the size of bit sequence (m) : 3
Sequence number possible from 0 to 7
4 connected ip 127.0.0.1:32133
data received : A
received frame number 0 while expecting 0
Randomly discarding this frame
data received : d
received frame number 1 while expecting 0
Randomly discarding this frame
data received : i
received frame number 2 while expecting 0
Randomly discarding this frame
data received : t
received frame number 3 while expecting 0
Randomly discarding this frame
data received : y
received frame number 4 while expecting 0
requesting for frame number 0

data received : a
received frame number 5 while expecting 0
requesting for frame number 0

data received : A
received frame number 0 while expecting 0
Randomly discarding this frame
data received : d
received frame number 1 while expecting 0
requesting for frame number 0

data received : i
```

```
cnlab@cnlab-OptiPlex-3010:~/31451$ ^C
cnlab@cnlab-OptiPlex-3010:~/31451$ g++ client.cpp -o out1 -pthread
^[[Acnlab@cnlab-OptiPlex-3010:~/31451$./out1
--------------CLIENT SIDE--------------
Connected to the server! (127.0.0.1)
Sliding Window Protocols:
1.GO Back N
2.Selective Repeat
Enter your choice : 1
Enter the size of bit sequence (m): 3
Sequence number possible from 0 to 7
Size of the sliding window is 7
Current Sliding window 0 to 6
Enter the data to be sent : Aditya
Sending frame A with sequence number 0
Frame sent
Sending frame d with sequence number 1
Frame sent
Sending frame i with sequence number 2
Frame sent
Sending frame t with sequence number 3
Frame sent
Sending frame y with sequence number 4
Frame sent
acknowledgment received, requesting number 0
Sending frame a with sequence number 5
Frame sent
acknowledgment received, requesting number 0
Waiting for acknowledgement for frame number 0
waiting for 3 seconds
resending frames, starting from frame number 0
Sending frame A with sequence number 0
Frame sent
Sending frame d with sequence number 1
Frame sent
acknowledgment received, requesting number 0
Sending frame i with sequence number 2
Frame sent
Sending frame t with sequence number 3
```


Activities
Terminal
Nov 4 09:43
cnlab@cnlab-OptiPlex-3010: ~/31451
cnlab@cnlab-OptiPlex-3010: ~/31451

```
received frame number 5 while expecting 0
Randomly discarding this frame
data received : A
received frame number 0 while expecting 0
frame received correctly
requesting for frame number 1

data received : d
received frame number 1 while expecting 1
frame received correctly
requesting for frame number 2

data received : i
received frame number 2 while expecting 2
frame received correctly
requesting for frame number 3

data received : t
received frame number 3 while expecting 3
Randomly discarding this frame
data received : y
received frame number 4 while expecting 3
requesting for frame number 3

data received : a
received frame number 5 while expecting 3
Randomly discarding this frame
data received : y
received frame number 3 while expecting 3
frame received correctly
requesting for frame number 4

data received : a
received frame number 4 while expecting 4
frame received correctly
requesting for frame number 5
```

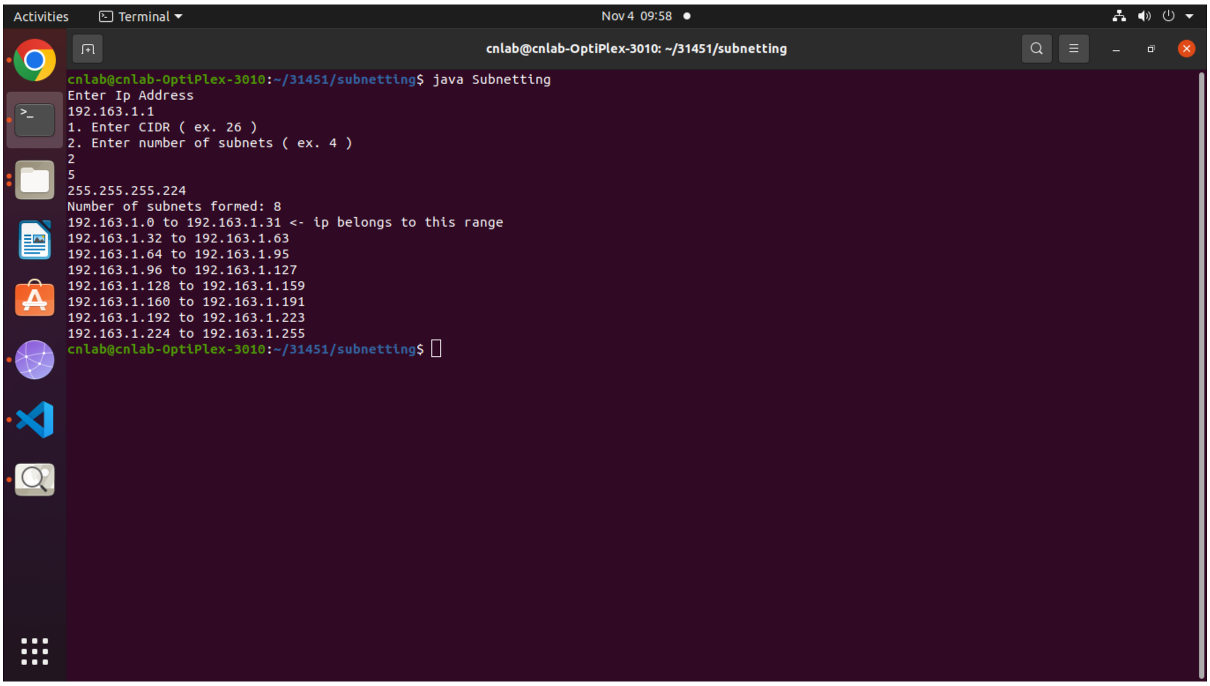
```
Sending frame i with sequence number 2
Frame sent
Sending frame t with sequence number 3
Frame sent
Sending frame y with sequence number 4
Frame sent
acknowledgment received, requesting number 0
Sending frame a with sequence number 5
Frame sent
Waiting for acknowledgement for frame number 0
waiting for 3 seconds
resending frames, starting from frame number 0
Sending frame A with sequence number 0
Frame sent
acknowledgment received, requesting number 1
Sending frame d with sequence number 1
Frame sent
acknowledgment received, requesting number 2
Sending frame i with sequence number 2
Frame sent
acknowledgment received, requesting number 3
Sending frame t with sequence number 3
Frame sent
Sending frame y with sequence number 4
Frame sent
acknowledgment received, requesting number 3
Sending frame a with sequence number 5
Frame sent
Waiting for acknowledgement for frame number 3
waiting for 3 seconds
resending frames, starting from frame number 3
Sending frame y with sequence number 3
Frame sent
acknowledgment received, requesting number 4
Sending frame a with sequence number 4
Frame sent
acknowledgment received, requesting number 5
cnlab@cnlab-OptiPlex-3010:~/31451$
```

Assignment No. 5 (B1)

```
import java.util.*;
import java.lang.Math;

public class Subnetting
{
    // 8+8+8+x
    // powerNumber = 2^x
    int powerNumber;

    private int getPowerNumber()
    {
        return powerNumber;
    }

    private void setPowerNumberFromNoOfSubnets( int nSubnets )
    {
        while( 256%nSubnets != 0 )
        {
            nSubnets++;
        }
        // powerNumber = (int)Math.pow(2,nSubnets);
        powerNumber = 256/nSubnets;
    }

    private void setPowerNumberFromCIDR( int cidr )
    {
        // finding 8 + 8 + ? + ?
        int mod = cidr%8;
        powerNumber = (int)Math.pow(2,8-mod);
    }

    private int getNumberOfSubnets()
    {
        return (256/powerNumber);
    }

    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        String ip,subnetMask = "255.255.";
        int choice,cidr,nSubnets;
        boolean isSupernetting = false;
        Subnetting subnetting = new Subnetting(); // created object because main() is static. Either do
this or create another class especially for main()

        System.out.println("Enter Ip Address");
        ip = sc.next();
        String[] test = ip.split("\\.",5);
        for ( String str : test )
        {
            int x = Integer.valueOf(str);
            if( x < 0 || x > 255 )
            {
                System.out.println("Invalid IP");
                System.exit(1);
            }
        }

        System.out.println("1. Enter CIDR ( ex. 26 )");
System.out.println("2. Enter number of subnets ( ex. 4 )");
choice = sc.nextInt();

if( choice!=1 && choice !=2)
{
    System.out.println("Invalid Input");
    sc.close();
    System.exit(1);
}


if( choice == 1 )
{
    cidr = sc.nextInt();
    if( cidr < 16 || cidr > 31)
    {
        System.out.println("CIDR Does not fit into subnetting or supernetting");
        System.exit(1);
    }
    // finding if supernetting or subnetting
    if( Integer.valueOf(cidr / 8) < 3 )
        isSupernetting = true;

    subnetting.setPowerNumberFromCIDR(cidr);
}
else if ( choice == 2 )
{
    nSubnets = sc.nextInt();
    subnetting.setPowerNumberFromNoOfSubnets(nSubnets);
}

int host = 256 - subnetting.getPowerNumber();

if( isSupernetting )
    subnetMask += host + ".0";
else
    subnetMask += "255." + host;
System.out.println(subnetMask);

if(!isSupernetting)
    System.out.println("Number of subnets formed: " + subnetting.getNumberOfSubnets());
else
    System.out.println("Number of supernets formed: " + subnetting.getNumberOfSubnets());

// removing last element from
ArrayList<String> test2 = new ArrayList<>(Arrays.asList(test));
int lastIpBits;
if( isSupernetting )
{
    test2.remove(2);
    test2.remove(2);
    lastIpBits = Integer.valueOf(test[2]);
}
else
{
    test2.remove(3);
    lastIpBits = Integer.valueOf(test[3]);
}

// converting array back to string
// half ip will be first 3 ip bits e.g. 192.168.13. ( for printing range )
```

```
        String halfIp = "";
        for( String str : test2 )
        {
            halfIp = halfIp + str + ".";
        }

        // finding range
        int pow = subnetting.getPowerNumber();
        int maxLimit = pow;
        int minLimit = 0;
        while( 256 >= maxLimit )
        {
            if( !isSupernetting )
                System.out.print( halfIp + minLimit + " to " + halfIp + (maxLimit-1) );
            else
                System.out.print( halfIp + minLimit + ".0" + " to " + halfIp + (maxLimit-1) + ".0");

            if( minLimit < lastIpBits && maxLimit > lastIpBits )
                System.out.print(" <- ip belongs to this range\n");
            else
                System.out.println();
            minLimit = maxLimit;
            maxLimit += pow;
        }

        sc.close();
    }
}
```

Output:

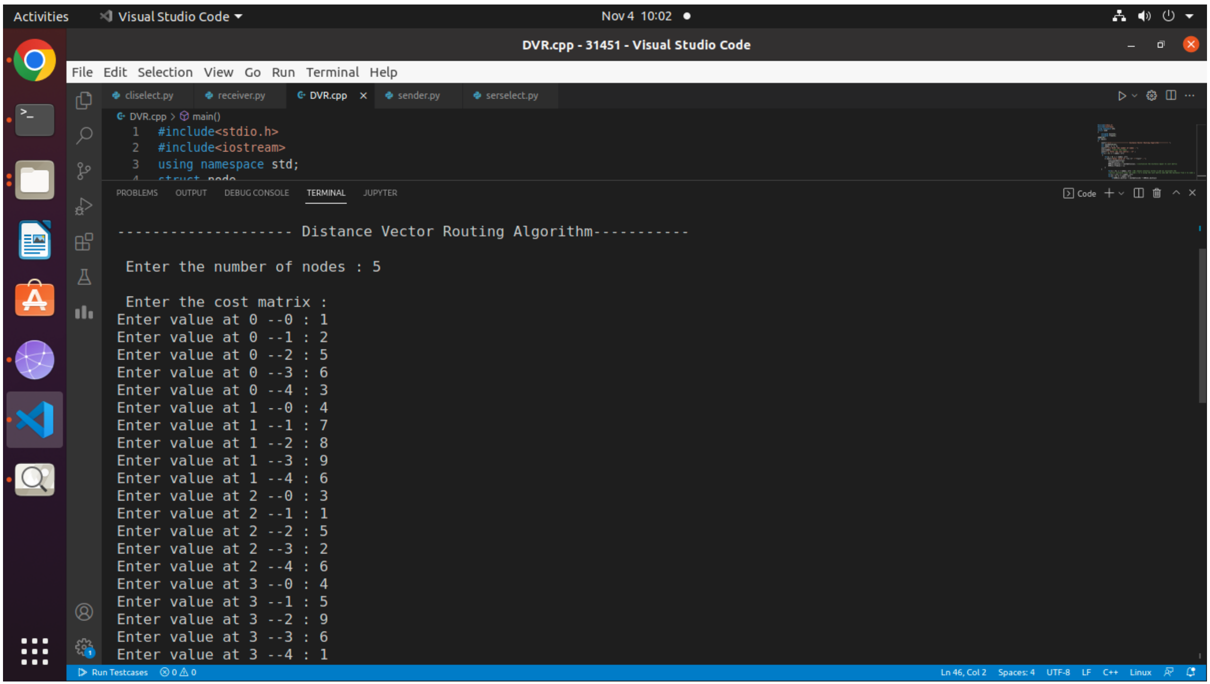


```
cnlab@cnlab-OptiPlex-3010:~/31451/subnetting$ java Subnetting
Enter Ip Address
192.163.1.1
1. Enter CIDR ( ex. 26 )
2. Enter number of subnets ( ex. 4 )
2
5
255.255.255.224
Number of subnets formed: 8
192.163.1.0 to 192.163.1.31 <- ip belongs to this range
192.163.1.32 to 192.163.1.63
192.163.1.64 to 192.163.1.95
192.163.1.96 to 192.163.1.127
192.163.1.128 to 192.163.1.159
192.163.1.160 to 192.163.1.191
192.163.1.192 to 192.163.1.223
192.163.1.224 to 192.163.1.255
cnlab@cnlab-OptiPlex-3010:~/31451/subnetting$ 
```

Assignment No. 6 (B2)

```
#include<stdio.h>
#include<iostream>
using namespace std;
struct node
{
    unsigned dist[6];
    unsigned from[6];
}DVR[10];
int main()
{
    cout<<"\n\n-------------------- Distance Vector Routing Algorithm----------- ";
    int costmat[6][6];
    int nodes, i, j, k;
    cout<<"\n\n Enter the number of nodes : ";
    cin>>nodes; //Enter the nodes
    cout<<"\n Enter the cost matrix : \n" ;
    for(i = 0; i < nodes; i++)
     {
        for(j = 0; j < nodes; j++)
        { cout<<"Enter value at "<<i <<" --"<<j<<" : ";
            cin>>costmat[i][j];
            costmat[i][i] = 0;
            DVR[i].dist[j] = costmat[i][j]; //initialise the distance equal to cost matrix
            DVR[i].from[j] = j;
        }
    }
            for(i = 0; i < nodes; i++) //We choose arbitary vertex k and we calculate the
            //direct distance from the node i to k using the cost matrix and add the distance from k to
node j
            for(j = i+1; j < nodes; j++)
            for(k = 0; k < nodes; k++)
                if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])
                { //We calculate the minimum distance
                    DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];
                    DVR[j].dist[i] = DVR[i].dist[j];
                    DVR[i].from[j] = k;
                    DVR[j].from[i] = k;
                }
        for(i = 0; i < nodes; i++)
        {
            cout<<"\n\n For router: "<<i+1;
            for(j = 0; j < nodes; j++)
                cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Distance "<<DVR[i].dist[j];
        }
    cout<<" \n\n ";
    return 0;
}
```

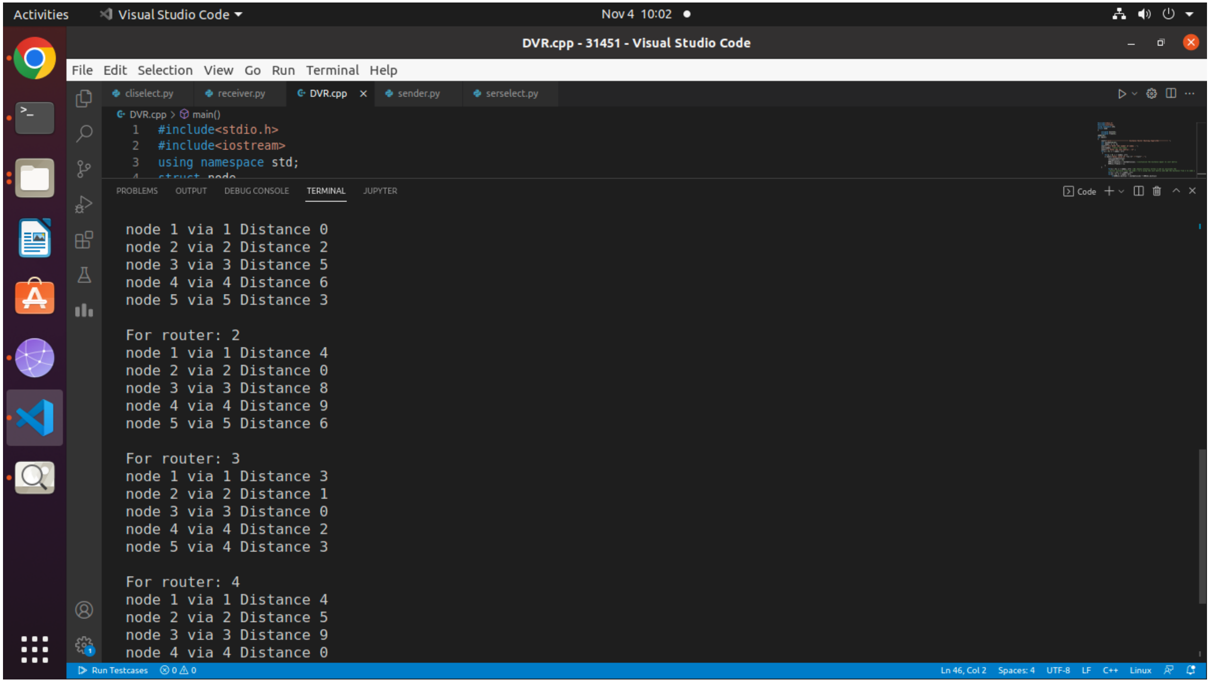
Output:



Activities
Visual Studio Code
Nov 4 10:02
DVR.cpp - 31451 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
cliselect.py
receiver.py
DVR.cpp
sender.py
serselect.py
DVR.cpp > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Code
Run Testcases
Ln 46, Col 2 Spaces: 4 UTF-8 LF C++ Linux

```
#include<stdio.h>
#include<iostream>
using namespace std;
struct node
```

```
------------------- Distance Vector Routing Algorithm-----------

 Enter the number of nodes : 5

 Enter the cost matrix :
Enter value at 0 --0 : 1
Enter value at 0 --1 : 2
Enter value at 0 --2 : 5
Enter value at 0 --3 : 6
Enter value at 0 --4 : 3
Enter value at 1 --0 : 4
Enter value at 1 --1 : 7
Enter value at 1 --2 : 8
Enter value at 1 --3 : 9
Enter value at 1 --4 : 6
Enter value at 2 --0 : 3
Enter value at 2 --1 : 1
Enter value at 2 --2 : 5
Enter value at 2 --3 : 2
Enter value at 2 --4 : 6
Enter value at 3 --0 : 4
Enter value at 3 --1 : 5
Enter value at 3 --2 : 9
Enter value at 3 --3 : 6
Enter value at 3 --4 : 1
```

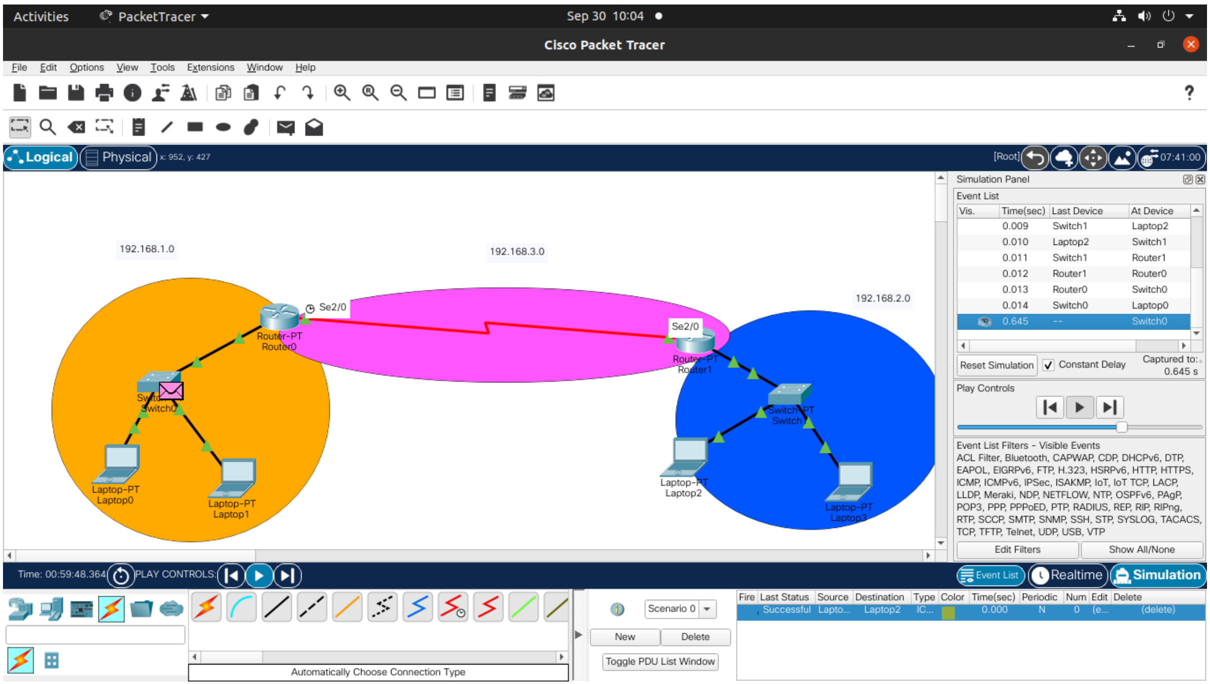


Activities
Visual Studio Code
Nov 4 10:02
DVR.cpp - 31451 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
cliselect.py
receiver.py
DVR.cpp
sender.py
serselect.py
DVR.cpp > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Code
Run Testcases
Ln 46, Col 2 Spaces: 4 UTF-8 LF C++ Linux

```
#include<stdio.h>
#include<iostream>
using namespace std;
struct node
```

```
 node 1 via 1 Distance 0
 node 2 via 2 Distance 2
 node 3 via 3 Distance 5
 node 4 via 4 Distance 6
 node 5 via 5 Distance 3

 For router: 2
 node 1 via 1 Distance 4
 node 2 via 2 Distance 0
 node 3 via 3 Distance 8
 node 4 via 4 Distance 9
 node 5 via 5 Distance 6

 For router: 3
 node 1 via 1 Distance 3
 node 2 via 2 Distance 1
 node 3 via 3 Distance 0
 node 4 via 4 Distance 2
 node 5 via 4 Distance 3

 For router: 4
 node 1 via 1 Distance 4
 node 2 via 2 Distance 5
 node 3 via 3 Distance 9
 node 4 via 4 Distance 0
```

Assignment No. 7 (B3)



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device
	0.009	Switch1	Laptop2
	0.010	Laptop2	Switch1
	0.011	Switch1	Router1
	0.012	Router1	Router0
	0.013	Router0	Switch0
	0.014	Switch0	Laptop0
👁	0.645	--	Switch0

Reset Simulation ☑ Constant Delay Captured to: 0.645 s

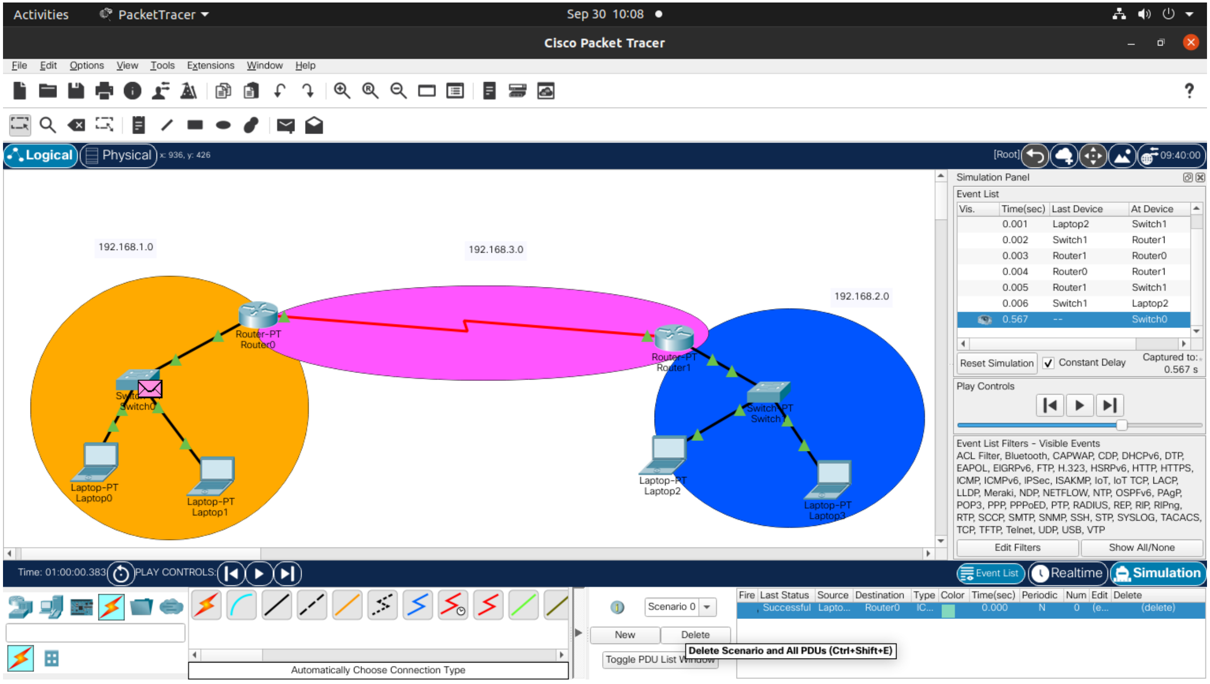
Play Controls

Event List Filters - Visible Events

ACL Filter, Bluetooth, CAPWAP, CDP, DHCPv6, DTP, EAPOL, EIGRPv6, FTP, H.323, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters | Show All/None

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	Lapto...	Laptop2	IC...	🟩	0.000	N	0	(e...	(delete)



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device
	0.001	Laptop2	Switch1
	0.002	Switch1	Router1
	0.003	Router1	Router0
	0.004	Router0	Router1
	0.005	Router1	Switch1
	0.006	Switch1	Laptop2
👁	0.567	--	Switch0

Reset Simulation ☑ Constant Delay Captured to: 0.567 s

Play Controls

Event List Filters - Visible Events

ACL Filter, Bluetooth, CAPWAP, CDP, DHCPv6, DTP, EAPOL, EIGRPv6, FTP, H.323, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters | Show All/None

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	Lapto...	Router0	IC...	🟩	0.000	N	0	(e...	(delete)

Assignment No. 8 (B4)

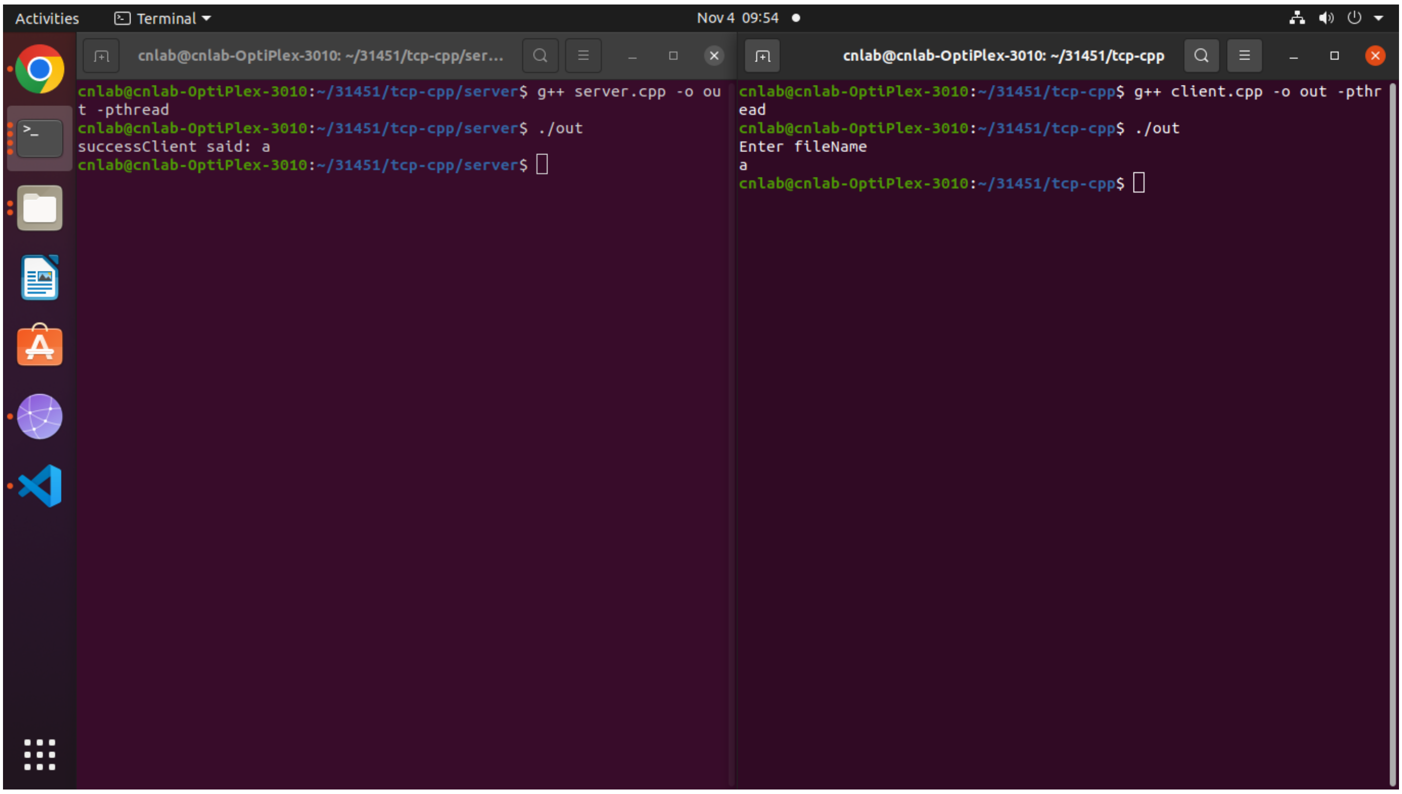
Server.cpp

```
#include<iostream>
#include<unistd.h>
#include<netinet/in.h>
#include<string.h>
#include<string>
#include<fstream>
#include <arpa/inet.h>
#include<sys/socket.h>

#define PORT 6511
#define MAXLINES 1024
using namespace std;
int main()
{
    int sockfd,connfd;
    sockaddr_in server,client;
    char buffer[MAXLINES];
    char fileBuffer[MAXLINES];
    sockfd = socket(AF_INET,SOCK_STREAM,0);
    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);
    server.sin_addr.s_addr = INADDR_ANY;
    bind(sockfd,(const sockaddr*) &server,sizeof(server));
    listen(sockfd,5);
    socklen_t len;
    connfd = accept(sockfd, (sockaddr *)&client, &len);
     if (connfd < 0)
         cout << "failse";
     else
         cout << "success";
    int n = read(connfd,buffer,sizeof(buffer));
    buffer[n] = '\0';
    cout << "Client said: " << buffer << endl;
    ifstream ifs(buffer,ios::in|ios::ate);
    int size = ifs.tellg();
    ifs.seekg(ios::beg);
    ifs.read(fileBuffer,size);
    send(connfd,fileBuffer,size,0);
    return 0;
}
```

Client.cpp

```
#include<iostream>
#include<unistd.h>
#include<netinet/in.h>
#include<string.h>
#include<string>
#include<fstream>
#include <arpa/inet.h>
#include<sys/socket.h>

#define PORT 6511
#define MAXLINES 1024
using namespace std;
int main()
{
    int sockfd;
    sockaddr_in server;
    char buffer[MAXLINES];
    string fileName;

    if( (sockfd = socket(AF_INET,SOCK_STREAM,0) ) < 0 )
    {
        cout << "err";
    }
    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);
    server.sin_addr.s_addr = inet_addr("127.0.0.1");

    if(connect(sockfd,(const sockaddr *) &server,sizeof(server)) < 0 )
        cout << "error";

    cout << "Enter fileName" << endl;
    cin >> fileName;
    send(sockfd,fileName.c_str(),fileName.length(),0);

    int n = read(sockfd,buffer,MAXLINES);

    ofstream ofs(fileName,ios::out);
    ofs.write(buffer,n);
    ofs.close();

    return 0;
}
```

Output:


Activities
Terminal
Nov 4 09:54
cnlab@cnlab-OptiPlex-3010: ~/31451/tcp-cpp/ser...
cnlab@cnlab-OptiPlex-3010: ~/31451/tcp-cpp

```
cnlab@cnlab-OptiPlex-3010:~/31451/tcp-cpp/server$ g++ server.cpp -o ou
t -pthread
cnlab@cnlab-OptiPlex-3010:~/31451/tcp-cpp/server$ ./out
successClient said: a
cnlab@cnlab-OptiPlex-3010:~/31451/tcp-cpp/server$ 
```

```
cnlab@cnlab-OptiPlex-3010:~/31451/tcp-cpp$ g++ client.cpp -o out -pthr
ead
cnlab@cnlab-OptiPlex-3010:~/31451/tcp-cpp$ ./out
Enter fileName
a
cnlab@cnlab-OptiPlex-3010:~/31451/tcp-cpp$ 
```

Assignment No. 9 (B5)

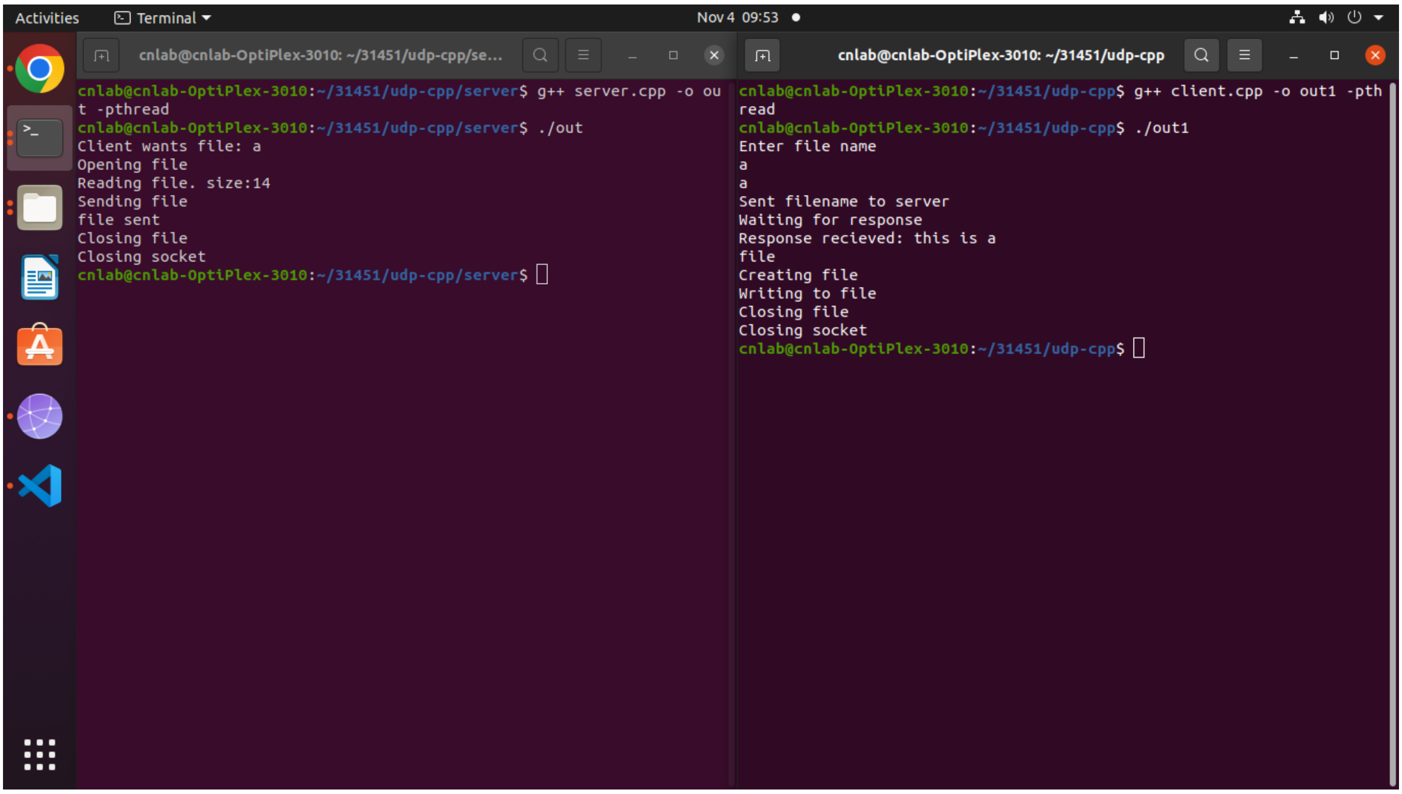
Server.cpp

```
#include<iostream>
#include<stdlib.h>
#include<netinet/in.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<stdio.h>
#include<fstream>
#include<string>

#define PORT 7512
#define MAXLINES 1024

using namespace std;

int main()
{
    int sockfd;
    char* fileName = new char[1];
    char buffer[MAXLINES];
    struct sockaddr_in server,client;

    sockfd = socket(AF_INET,SOCK_DGRAM,0);
    memset(&server,0,sizeof(server));
    memset(&client,0,sizeof(client));

    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(PORT);

    bind(sockfd,(const struct sockaddr*) &server,sizeof(server));

    socklen_t len;
    int n = recvfrom(sockfd,fileName,MAXLINES,0,( struct sockaddr *)&client,&len);
    fileName[n]='\0';
    cout << "Client wants file: " << fileName << endl;

    cout << "Opening file" << endl;
    std::ifstream ifs(fileName,ios::ate);
    if(!ifs)
    {
        cout << "file not present";
    }
    else
    {
        int size = ifs.tellg();
        ifs.seekg(ios::beg);

        cout << "Reading file. size:"<<size << endl;
        ifs.read(buffer,size);

        cout << "Sending file" << endl;

        sendto(sockfd,(const char *)buffer,size,0,(const struct sockaddr *) &client,sizeof(client));
        cout << "file sent " << endl;
        cout << "Closing file" << endl;
        ifs.close();
        cout << "Closing socket" << endl;
        close(sockfd);
    }

    return 0;
}
```

Client.cpp

```
#include<iostream>
#include<fstream>
#include<stdlib.h>
#include<netinet/in.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<stdio.h>
#include<string>

#define PORT 7512
#define MAXLINES 1024

using namespace std;

int main()
{
    int sockfd;
    string fileName;
    char buffer[MAXLINES];
    struct sockaddr_in server;

    sockfd = socket(AF_INET,SOCK_DGRAM,0);
    memset(&server,0,sizeof(server));

    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(PORT);

    cout << "Enter file name" << endl;
    cin >> fileName;

    cout << fileName.c_str() << endl;
    sendto(sockfd, (const char *)fileName.c_str(), fileName.length(),0,(const struct sockaddr *)&server,
sizeof(server));
    cout << "Sent filename to server" << endl;

    cout << "Waiting for response" << endl;
    socklen_t len;
    int n = recvfrom(sockfd,(char *)buffer,sizeof(buffer),0,(struct sockaddr *)&server,&len);
    buffer[n] = '\0';
    cout << "Response recieved: " << buffer << endl;
    ofstream ofs;
    cout << "Creating file" << endl;
    ofs.open(fileName,ios::out);
    cout << "Writing to file" << endl;
    ofs.write(buffer,n);
    cout << "Closing file" << endl;
    ofs.close();
    cout << "Closing socket"<<endl;
    close(sockfd);
    return 0;
}
```

Output:



Activities
Terminal
Nov 4 09:53
cnlab@cnlab-OptiPlex-3010: ~/31451/udp-cpp/se...
cnlab@cnlab-OptiPlex-3010: ~/31451/udp-cpp

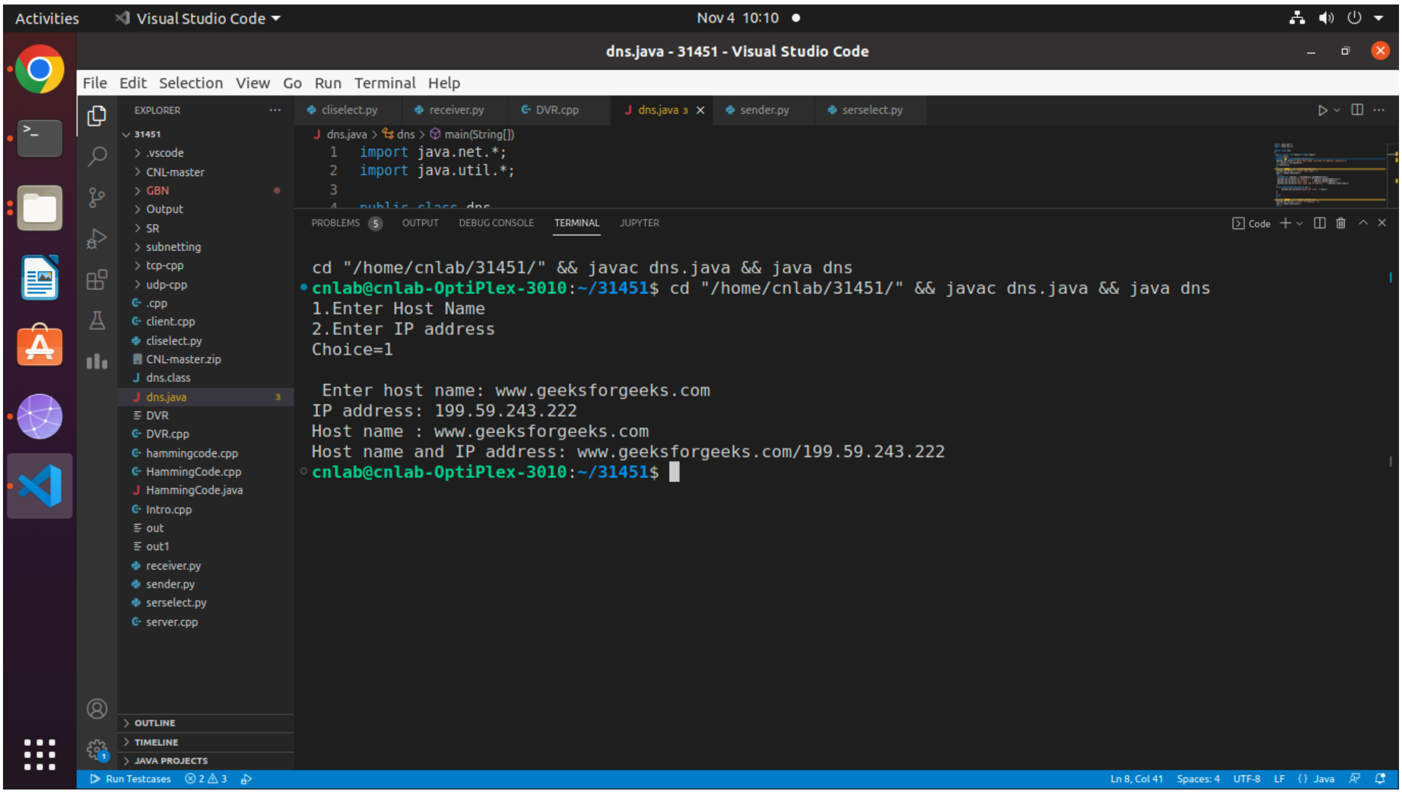
```
cnlab@cnlab-OptiPlex-3010:~/31451/udp-cpp/server$ g++ server.cpp -o ou
t -pthread
cnlab@cnlab-OptiPlex-3010:~/31451/udp-cpp/server$ ./out
Client wants file: a
Opening file
Reading file. size:14
Sending file
file sent
Closing file
Closing socket
cnlab@cnlab-OptiPlex-3010:~/31451/udp-cpp/server$
```

```
cnlab@cnlab-OptiPlex-3010:~/31451/udp-cpp$ g++ client.cpp -o out1 -pth
read
cnlab@cnlab-OptiPlex-3010:~/31451/udp-cpp$ ./out1
Enter file name
a
a
Sent filename to server
Waiting for response
Response recieved: this is a
file
Creating file
Writing to file
Closing file
Closing socket
cnlab@cnlab-OptiPlex-3010:~/31451/udp-cpp$
```

Assignment No. 10 (C1)

```
import java.net.*;
import java.util.*;

public class dns
{
 public static void main(String[] args){
  String host;
  Scanner ch = new Scanner(System.in);
  System.out.print("1.Enter Host Name \n2.Enter IP address \nChoice=");
  int choice = ch.nextInt();
  if(choice==1)
  {
  Scanner input = new Scanner(System.in);
  System.out.print("\n Enter host name: ");
  host = input.nextLine();
  try {
   InetAddress address = InetAddress.getByName(host);
   System.out.println("IP address: " + address.getHostAddress());
   System.out.println("Host name : " + address.getHostName());
   System.out.println("Host name and IP address: " + address.toString());
  }
  catch (UnknownHostException ex) {
       System.out.println("Could not find " + host);
  }
  }
  else
  {
  Scanner input = new Scanner(System.in);
  System.out.print("\n Enter IP address: ");
  host = input.nextLine();
  try {
   InetAddress address = InetAddress.getByName(host);
   System.out.println("Host name : " + address.getHostName());
   System.out.println("IP address: " + address.getHostAddress());
   System.out.println("Host name and IP address: " + address.toString());

  }
  catch (UnknownHostException ex) {
       System.out.println("Could not find " + host);
  }
  }

 }
}
```

Output:



```java
import java.net.*;
import java.util.*;
```

```
cd "/home/cnlab/31451/" && javac dns.java && java dns
cnlab@cnlab-OptiPlex-3010:~/31451$ cd "/home/cnlab/31451/" && javac dns.java && java dns
1.Enter Host Name
2.Enter IP address
Choice=1

 Enter host name: www.geeksforgeeks.com
IP address: 199.59.243.222
Host name : www.geeksforgeeks.com
Host name and IP address: www.geeksforgeeks.com/199.59.243.222
cnlab@cnlab-OptiPlex-3010:~/31451$ 
```