**NAME : KHUSHI PANWAR**

**ROLL NO: 2021334**

**COMPUTER NETWORKS PRACTICAL FILE**

1. **Write a program to implement Cyclic Redundancy check (CRC) algorithm for noisy channel :**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int i, j, k, l;

//varaibles defined for loops

int fs; //message frame size sent

cout<<"\n Enter Frame/Message size : ";

cin>>fs;

int f[20]; //message frame sent stored in array

cout<<"\n Enter Frame/Message (in binary) : ";

for (i = 0; i<fs; i++)

{

cin>>f[i];

}

int gs; //generator size decided

cout<<"\n Enter Generator size : ";

cin>>gs;

int g[20]; //generator stored in arra

cout<<"\n Enter Generator : ";

for (i = 0; i<gs; i++)

{

cin>>g[i];

}

//printing the message or data frame sent

cout<<"\n Sender Side : ";

cout<<"\n Message sent/data Frame: ";

for (i = 0; i<fs; i++)

{

cout<<f[i];

}

cout<<"\n Generator : ";

for (i = 0; i<gs; i++)

{

cout<<g[i];

}

//zeros to be added add the end of msg bits (key - 1)

int rs = gs - 1;

cout<<"\n Number of 0's to be appended : "<<rs;

for (i = fs; i<fs + rs; i++)

{

f[i] = 0;

}//add zeros after the data bits

int temp[20];

for (i = 0; i<20; i++)

{

temp[i] = f[i];

} // temp is the duplicate of the data bits array

cout<<"\n Message after appending 0's : ";

for (i = 0; i<fs + rs; i++)

{

cout<<temp[i];

}//printing message after appending zero

for (i = 0; i<fs; i++){

j = 0;

k = i;

{

for (j = 0, k = i; j<gs; j++, k++)

{

if ((temp[k]==1 &&g[j] == 1) || (temp[k] == 0&& g[j] == 0))

{

temp[k]= 0;

}

else

{

temp[k] = 1;

}

}

}

}

int crc[15];

for (i = 0, j = fs; i<rs; i++, j++)

{

crc[i] = temp[j];

}

cout<<"\n CRC bits: ";

for (i = 0; i<rs; i++)

{

cout<<crc[i];

}

cout<<"\n Transmitted Frame: ";

int tf[15];

for (i = 0; i<fs; i++)

{

tf[i] = f[i];

}

for (i = fs, j = 0; i<fs + rs; i++, j++)

{

tf[i] = crc[j];

}

for (i = 0; i<fs + rs; i++)

{

cout<<tf[i];

}

cout<<"\n Receiver side : ";

cout<<"\n Received Frame: ";

for (i = 0; i<fs + rs; i++)

{

cout<<tf[i];

}

for (i = 0; i<fs + rs; i++)

{

temp[i] = tf[i];

}

for (i = 0; i<fs + rs; i++)

{

j = 0;

k = i;

if (temp[k] >= g[j])

{

for (j = 0, k = i; j<gs; j++, k++)

{

if ((temp[k] == 1&&g[j] == 1) || (temp[k] == 0&&

g[j] == 0))

{

temp[k] = 0;

}

else

{

temp[k] = 1;

}

}

}

}

cout<<"\n Remainder: ";

int rrem[15];

for (i = fs, j = 0; i<fs + rs; i++, j++)

{

rrem[j] = temp[i];

}

for (i = 0; i<rs; i++)

{

cout<<rrem[i];

}

int flag = 0;

for (i = 0; i<rs; i++)

{

if (rrem[i] != 0)

{

flag = 1;

}

}

if (flag == 0)

{

cout<<"\n Since Remainder Is 0 Hence Message Transmitted From Sender To Receiver Is Correct";

}

else

{

cout<<"\n Since Remainder Is Not 0 Hence Message Transmitted From Sender To Receiver Contains Error";

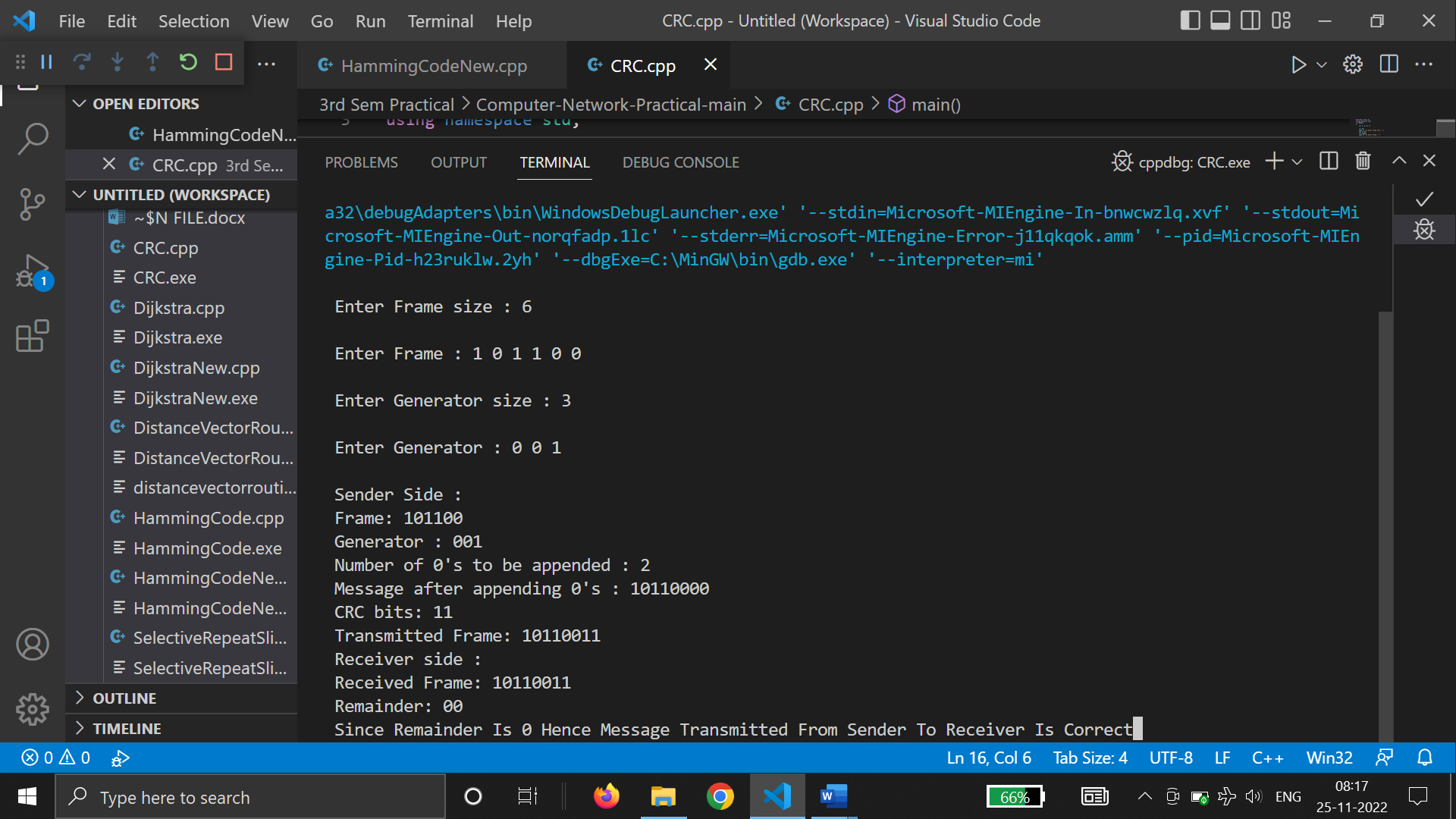
}

getch();

//hold the output window until user presses a key to escape

return 0;

}



1. **Simulate and implement the stop and wait protocol for noisy channel :**

#include<iostream>

#include<time.h>

#include<cstdlib>

#include<ctime>

#include<unistd.h>

usingnamespacestd;

classtimer

{

private:

    unsignedlongbegTime;

public:

    voidstart()

    {

        begTime = clock();

    }

    unsignedlongelapsedTime()

    {

        return ((unsignedlong)clock() - begTime) / CLOCKS\_PER\_SEC;

    }

    boolisTimeout(unsignedlongseconds)

    {

        returnseconds>= elapsedTime();

    }

};

intmain()

{

    cout<<endl<<"\t \*\* PROGRAM TO IMPLEMENT STOP AND WAIT PROTOCOL \*\* "<<endl<<endl;

    intfcount;

    cout<<"-> Enter the number of frames : ";

    cin>>fcount;

    cout<<"-> Enter the frames : ";

    intframes[10];

    for (inti=0 ; i<fcount; i++) {

        cin>>frames[i];

    }

    Unsigned long seconds = 5;

    srand(time(NULL));

    timert;

    cout<<"Sender has to send frames : ";

    for (inti = 0; i<fcount; i++)

        cout<<frames[i] <<" ";

    cout<<endl;

    int count = 0;

    bool delay = false;

    cout<<endl<<"Sender\t\t\t\t\tReceiver"<<endl;

do

    {

        Bool timeout = false;

        cout<<"Sending Frame : "<<frames[count];

        cout<<"\t\t";

        t.start();

        if (rand() % 2)

        {

            intto = 24600 + rand() % (64000 - 24600) + 1;

            for (int i = 0; i<64000; i++)

                for (intj = 0; j<to; j++)

                {

                }

        }

        if (t.elapsedTime() <= seconds)

        {

            cout<<"Received Frame : "<<frames[count] <<" ";

            if (delay)

            {

                cout<<"Duplicate";

                delay = false;

            }

            cout<<endl;

            count++;

        }

        else

        {

            cout<<"---"<<endl;

            cout<<"Timeout"<<endl;

            timeout = true;

        }

        t.start();

        if (rand() % 2 || !timeout)

        {

            intto = 24600 + rand() % (64000 - 24600) + 1;

            for (int i = 0; i<64000; i++)

                for (intj = 0; j<to; j++)

                {

                }

            if (t.elapsedTime() >seconds)

            {

                cout<<"Delayed Ack"<<endl;

                count--;

                delay = true;

            }

            elseif (!timeout)

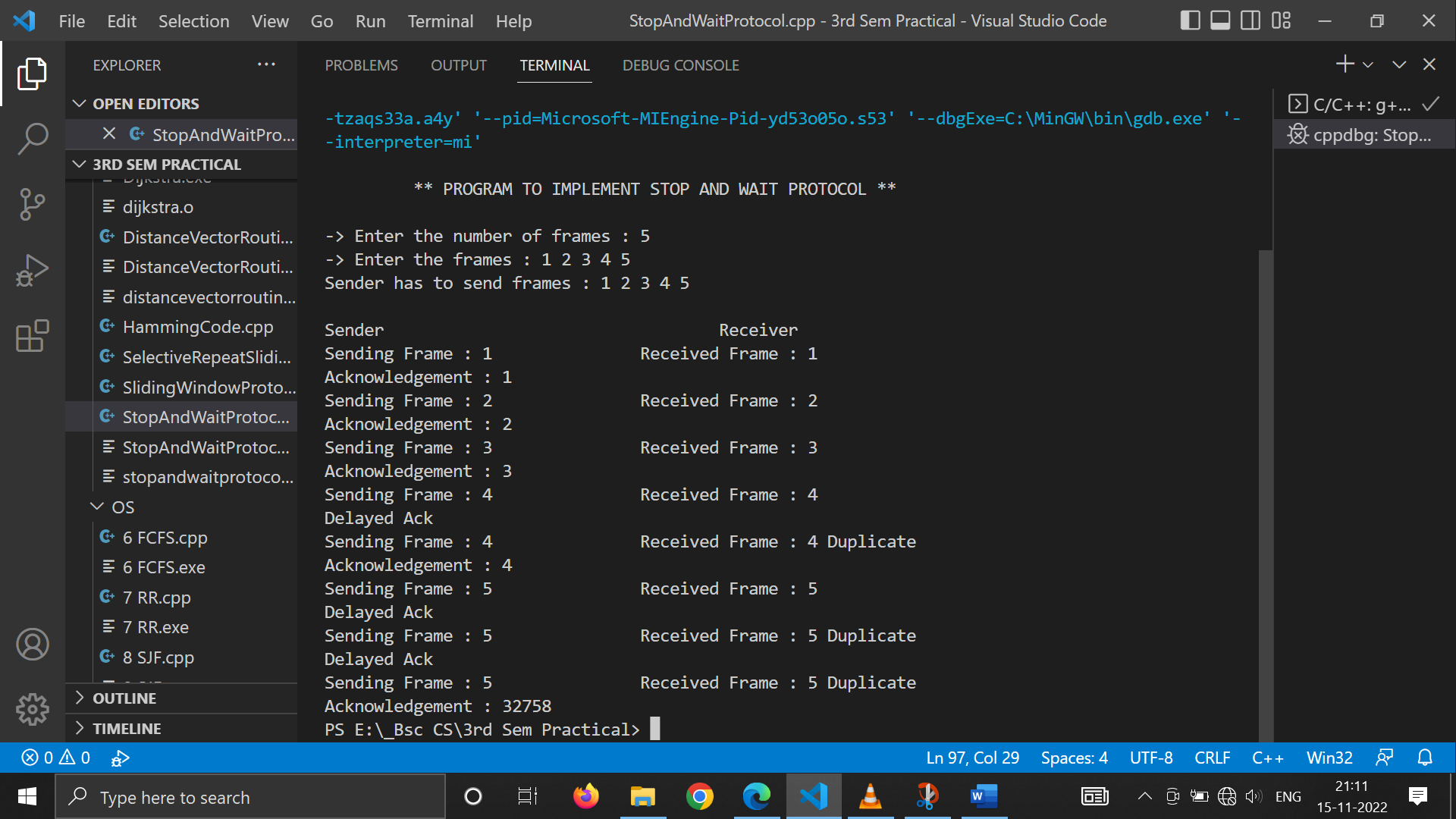
                cout<<"Acknowledgement : "<<frames[count] - 1<<endl;

        }

    } while (count != fcount);

    return0;

}



1. **Simulate and implement the Go back and sliding window protocol :**

#include<iostream>

usingnamespacestd;

intmain()

{

     cout<<endl<<"\t \*\* PROGRAM TO IMPLEMENT SLIDING WINDOW PROTOCOL \*\* "<<endl<<endl;

    intw,i,f,frames[50];

    cout<<"Enter window size: ";

    cin>>w;

    cout<<"\nEnter number of frames to transmit: ";

    cin>>f;

    cout<<"\nEnter "<<f<<" frames: ";

    for(i=1;i<=f;i++)

        cin>>frames[i];

    cout<<"\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n";

    cout<<"After sending "<<w<<" frames at each stage sender waits for acknowledgement sent by the receiver\n\n";

    for(i=1;i<=f;i++)

    {

        if(i%w==0)

        {

            cout<<frames[i]<<"\n";

            cout<<"Acknowledgement of above frames sent is received by sender\n\n";

        }

        else

            cout<<frames[i]<<" ";

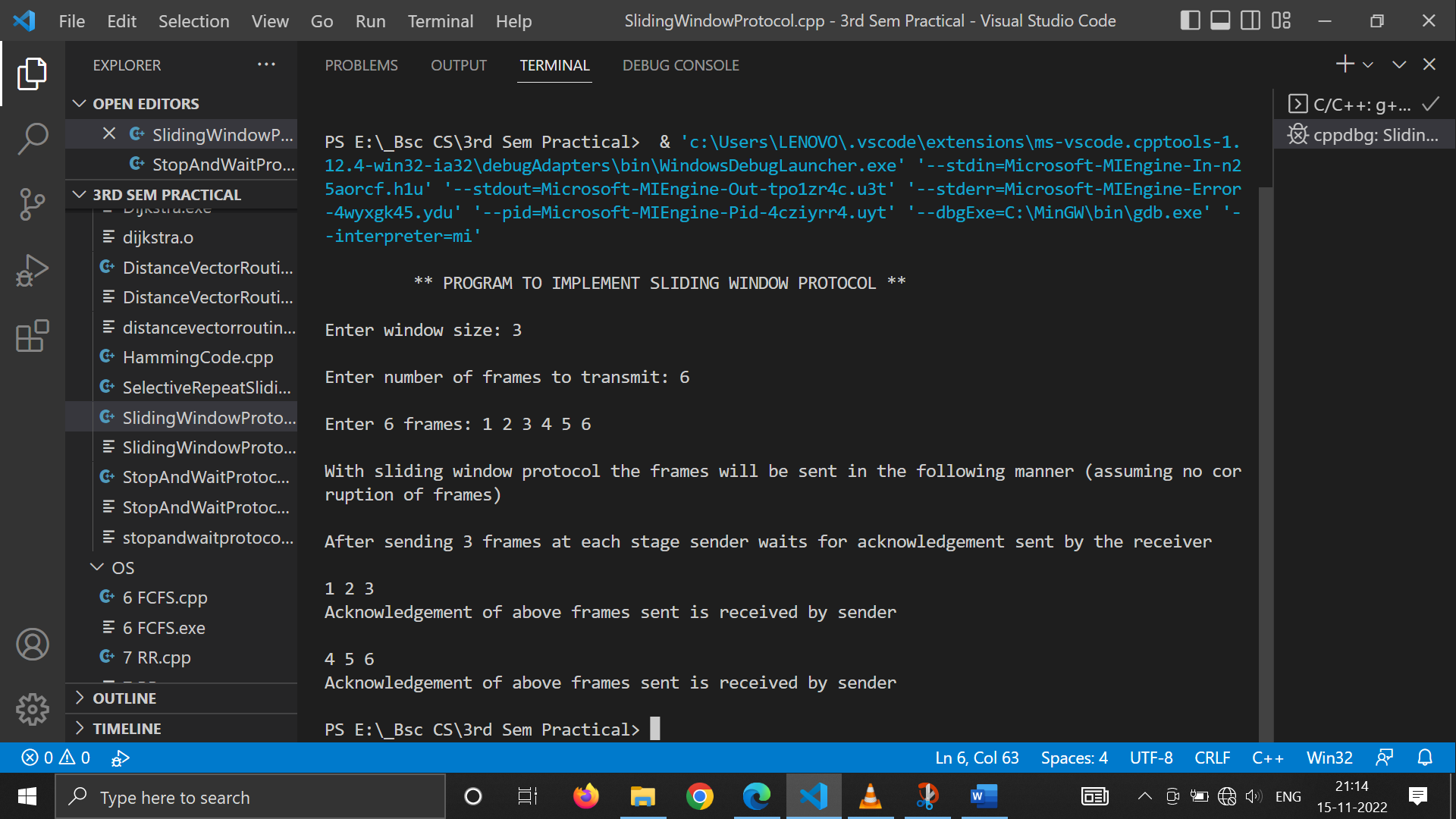
    }

    if(f%w!=0)

        cout<<"\nAcknowledgement of above frames sent is received by sender\n";

    return0;

}



1. **Simulate and implement the selective repeat sliding window protocol :**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#include<time.h>

#include<math.h>

usingnamespacestd;

#define TOT\_FRAMES 500

#define FRAMES\_SEND 10

classsel\_repeat

{

private:

    intfr\_send\_at\_instance;

    intarr[TOT\_FRAMES];

    intsend[FRAMES\_SEND];

    intrcvd[FRAMES\_SEND];

    charrcvd\_ack[FRAMES\_SEND];

    intsw;

    intrw;

public:

    voidinput();

    voidsender(int);

    voidreceiver(int);

};

voidsel\_repeat::input()

{

    intn;

    intm;

    inti;

    cout<<"\*\*\*\*\*Please enter the no. of bits for the sequence no.\*\*\*\* : ";

    cin>>n;

    m = pow(2, n);

    intt = 0;

    fr\_send\_at\_instance = (m / 2);

    for (i = 0; i<TOT\_FRAMES; i++)

    {

        arr[i] = t;

        t = (t + 1) % m;

    }

    for (i = 0; i<fr\_send\_at\_instance; i++)

    {

        send[i] = arr[i];

        rcvd[i] = arr[i];

        rcvd\_ack[i] = 'n';

    }

    rw = sw = fr\_send\_at\_instance;

    sender(m);

}

voidsel\_repeat::sender(intm)

{

    cout<<"--------------------------------------------------------"<<endl;

    for (inti = 0; i<fr\_send\_at\_instance; i++)

    {

        if (rcvd\_ack[i] == 'n')

            cout<<"SENDER : Frame "<<send[i] <<" is============> SENT!!\n";

    }

    cout<<"\n----------------------------------------------------"<<endl;

    receiver(m);

}

voidsel\_repeat::receiver(intm)

{

    time\_tt;

    intf;

    intj;

    intf1;

    inta1;

    charch;

    srand((unsigned)time(&t));

    for (inti = 0; i<fr\_send\_at\_instance; i++)

    {

        if (rcvd\_ack[i] == 'n')

        {

            f = rand() % 10;

            if (f != 5)

            {

                for (intj = 0; j<fr\_send\_at\_instance; j++)

                    if (rcvd[j] == send[i])

                    {

                        cout<<"reciever:Frame"<<rcvd[j] <<"recieved correctly\n";

                        rcvd[j] = arr[rw];

                        rw = (rw + 1) % m;

                        break;

                    }

                intj;

                if (j == fr\_send\_at\_instance)

                    cout<<"reciever:Duplicate frame"<<send[i] <<"discarded\n";

                a1 = rand() % 5;

                if (a1 == 3)

                {

                    cout<<"(acknowledgement "<<send[i] <<" lost)\n";

                    cout<<"(sender timeouts-->Resend the frame)\n";

                    rcvd\_ack[i] = 'n';

                }

                else

                {

                    cout<<"(acknowledgement "<<send[i] <<" recieved)\n";

                    rcvd\_ack[i] = 'p';

                }

            }

            else

            {

                intld = rand() % 2;

                if (ld == 0)

                {

                    cout<<"RECEIVER : Frame "<<send[i] <<" is damaged\n";

                    cout<<"RECEIVER : Negative Acknowledgement "<<send[i] <<" sent\n";

                }

                else

                {

                    cout<<"RECEIVER : Frame "<<send[i] <<" is lost\n";

                    cout<<"(SENDER TIMEOUTS-->RESEND THE FRAME)\n";

                }

                rcvd\_ack[i] = 'n';

            }

        }

    }

    for (intj = 0; j<fr\_send\_at\_instance; j++)

    {

        if (rcvd\_ack[j] == 'n')

            break;

    }

    inti = 0;

    for (intk = j; k<fr\_send\_at\_instance; k++)

    {

        send[i] = send[k];

        if (rcvd\_ack[k] == 'n')

            rcvd\_ack[i] = 'n';

        else

            rcvd\_ack[i] = 'p';

        i++;

    }

    if (i != fr\_send\_at\_instance)

    {

        for (intk = i; k<fr\_send\_at\_instance; k++)

        {

            send[k] = arr[sw];

            sw = (sw + 1) % m;

            rcvd\_ack[k] = 'n';

        }

    }

    cout<<"Do you want to continue?[Y/n]";

    cin>>ch;

    cout<<"\n";

    if (ch == 'y')

        sender(m);

    else

        exit(0);

}

intmain()

{

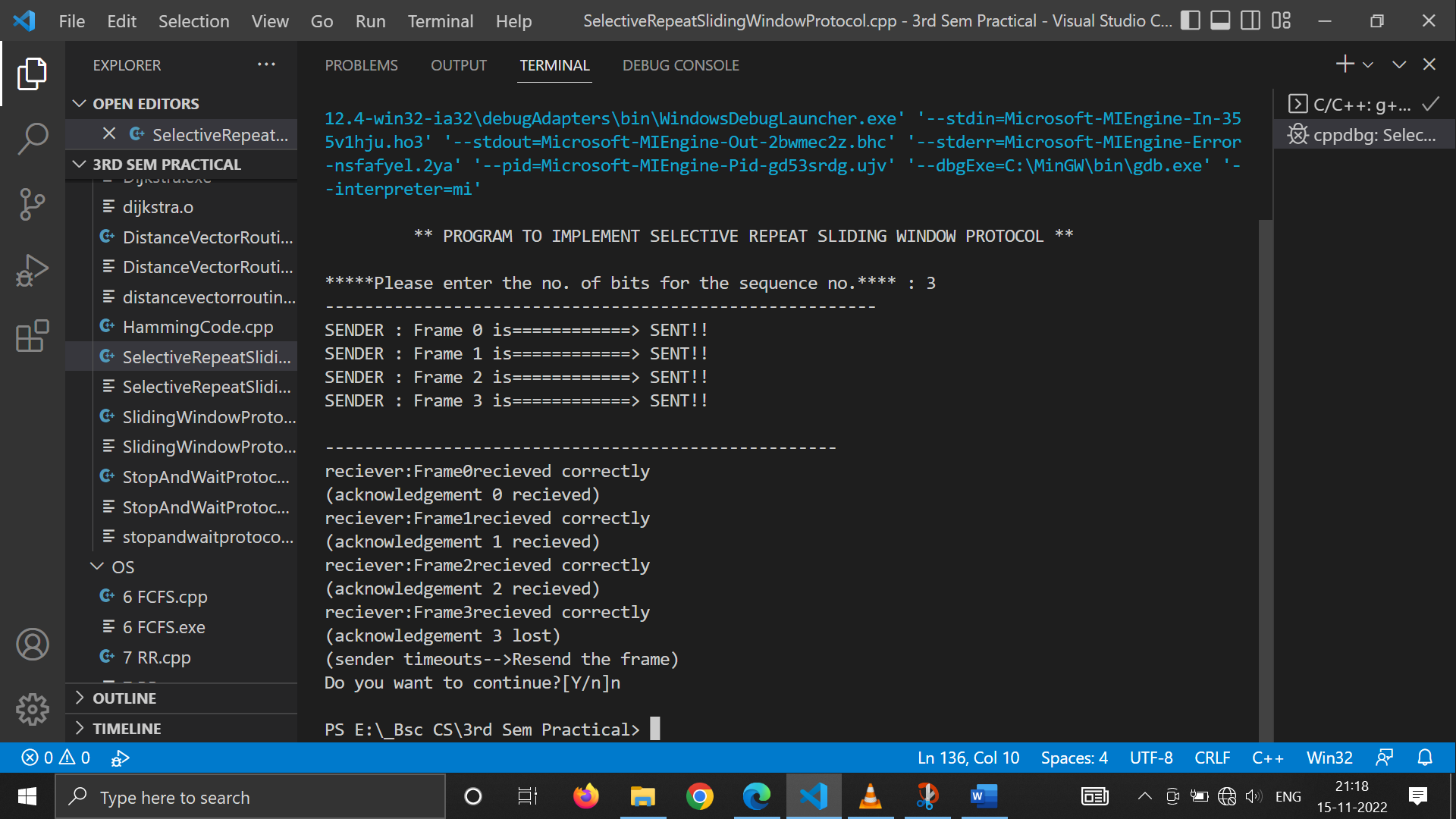
    cout<<endl<<"\t \*\* PROGRAM TO IMPLEMENT SELECTIVE REPEAT SLIDING WINDOW PROTOCOL \*\* "<<endl<<endl;

    sel\_repeatsr;

    sr.input();

    return0;

}



1. **Simulate and implement the selective repeat sliding window protocol :**

#include<stdio.h>

#include<iostream>

usingnamespacestd;

structnode

{

    unsigneddist[6];

    unsignedfrom[6];

}

DVR[10];

intmain()

{

    cout<<endl<<"\t \*\* PROGRAM TO IMPLEMENT DISTANCE VECTOR ROUTING ALGORITHM \*\* "<<endl<<endl;

    intcostmat[10][10];

    intnodes, i, j, k;

    cout<<"-> Enter the number of nodes : ";

    cin>>nodes; //Enter the nodes

    cout<<"-> Enter the cost matrix : \n" ;

    for(i = 0; i<nodes; i++)

     {

        for(j = 0; j<nodes; 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 thedirect 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++)

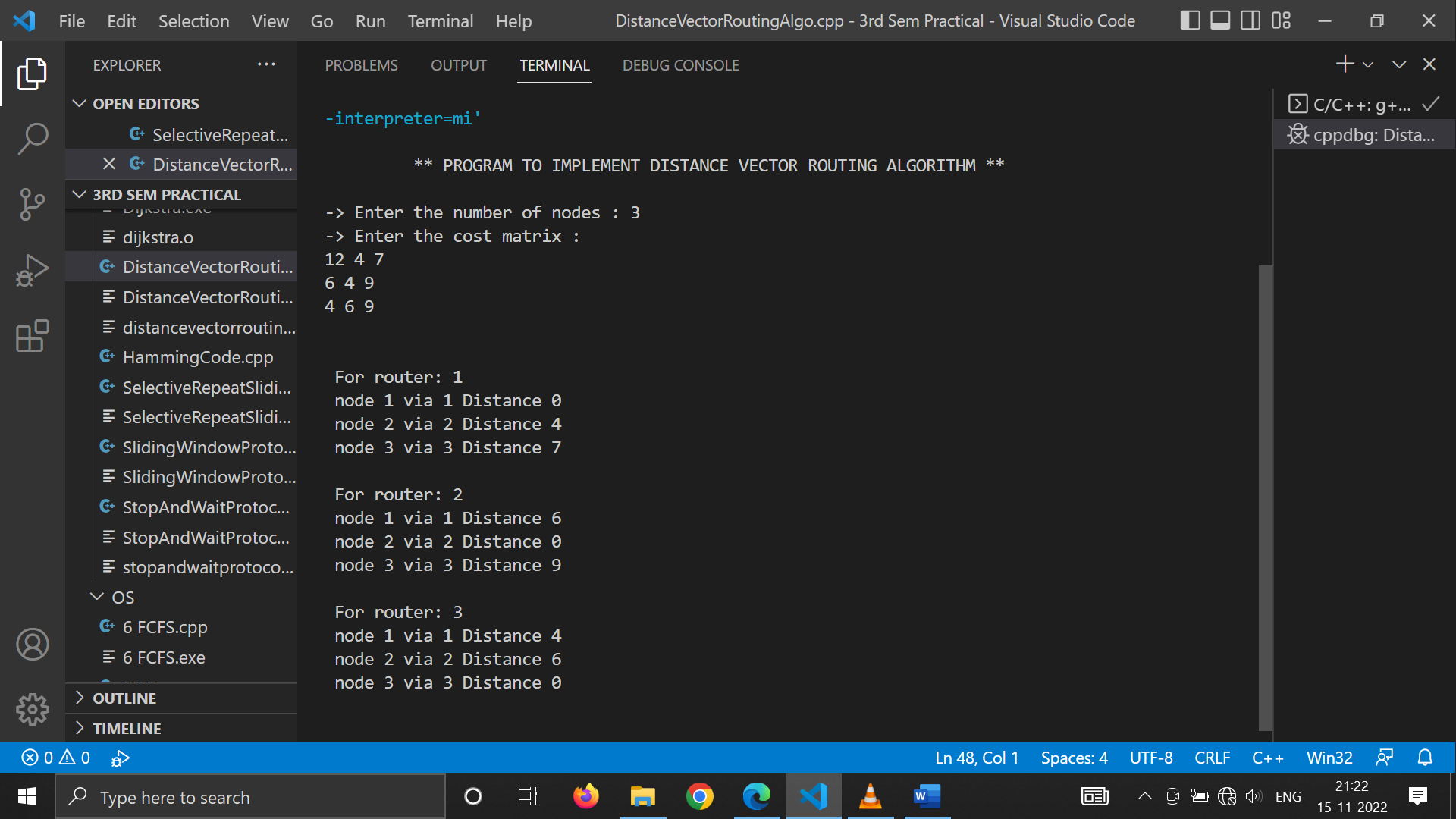
                cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Distance "<<DVR[i].dist[j];

        }

    cout<<" \n\n ";

    return 0;

}



1. **Simulate and implement the Dijkstra Algorithm for shortest path routing :**

#include<iostream>

#include<iomanip>

usingnamespacestd;

intfindMinVertex(intdistance[], bool Tset[], int size) // finding minimum distance

{

    intminVertex=-1;

    for(intk=0;k<size;k++)

    {

        if(Tset[k]==false && (minVertex==-1 || distance[k]<distance[minVertex]))

        {

            minVertex=distance[k];

        }

    }

    Return minVertex;

}

voidDijkstraAlgo(intgraph[100][100],int src, intsize) // adjacency matrix

{

    intdistance[size]; // // array to calculate the minimum distance for each node

    boolvisited[size];// boolean array to mark visited and unvisited for each node

    for(intk = 0; k<size; k++)

    {

        distance[k] = INT\_MAX;

        visited[k] = false;

    }//set all the values suitably

    distance[src] = 0;   // Source vertex distance is set 0

    for(intk = 0; k<size-1; k++){

        int minVertex=findMinVertex(distance, visited, size); //minimum distance / vertex

        visited[minVertex]=true;

        for(intj = 0; j<size; j++)

        {

            if (graph[minVertex][j]!=0 && !visited[j]){

                int dist= distance[minVertex]+graph[minVertex][j];

                if (dist<distance[j]){

                    distance[j]=dist;

                }

            }

            // updating the distance of neighbouring vertex

        }

    }

    cout<<"Vertex\t\tDistance from source vertex"<<endl;

    for(intk = 0; k<size; k++){

        charstr=65+k;

        cout<<str<<"\t\t"<<distance[k]<<endl;

    }

}

voidinput(intA[100][100], intsize){

    cout<<"-> Enter the Adjacency Matrix for the graph here : "<<endl;

    for (inti=0; i<size; i++){

        for (intj=0; j<size; j++){

            cin>>A[i][j];

        }

    }

}

intmain()

{

    cout<<endl<<"\t \*\* PROGRAM TO IMPLEMENT DIJKSTRA ALGORITHM  \*\* "<<endl<<endl;

    int size;

    cout<<"-> Enter the size of the graph : ";

    cin>>size;

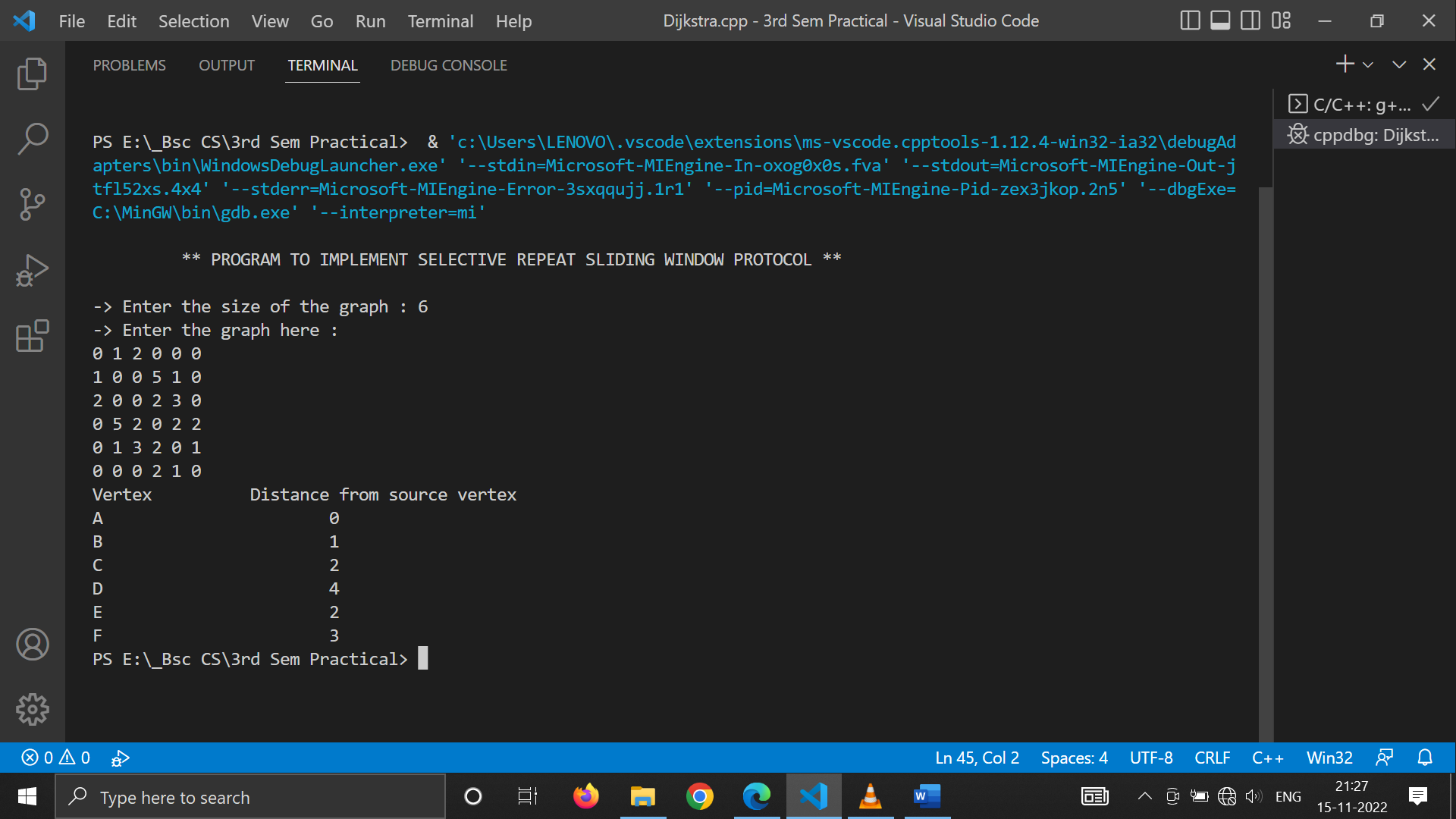
    intgraph[100][100];

    input(graph, size);

    DijkstraAlgo(graph,0, size);

    return0;

}



1. **Write a program to implement Hamming code :**

#include<iostream>

#include<cmath>

#include<string>

usingnamespacestd;

classHamming

{

    stringmessage;

    intcodeword[50], temp[50];

    intn, check;

    charparity;

public:

    Hamming()

    {

        parity = 'E';

        message="";

        n = check = 0;

        for (inti = 0; i<50; i++)

        {

            temp[i] = codeword[i] = 0;

        }

    }

    voidgenerate()

    {

        do

        {

            cout<<"Enter the message in binary : ";

            cin>>message;

        } while (message.find\_first\_not\_of("01") != string::npos);

        n = message.size();

        cout<<"Odd(O)/Even(E) Parity ? ";

        cin>>parity;

        for (unsignedinti = 0; i<message.size(); i++)

        {

            if (message[i] == '1')

                temp[i + 1] = 1;

            else

                temp[i + 1] = 0;

        }

        computeCode();

    }

    voidcomputeCode()

    {

        check = findr();

        cout<<"Number of Check Bits : "<<check<<endl;

        cout<<"Number of Bits in Codeword : "<<n + check<<endl;

        for (inti = (n + check), j = n; i>0; i--)

        {

            if ((i& (i - 1)) != 0)

                codeword[i] = temp[j--];

            else

                codeword[i] = setParity(i);

        }

        cout<<"Parity Bits - ";

        for (inti = 0; i<check; i++)

            cout<<"P"<<pow(2, i) <<" : "<<codeword[(int)pow(2, i)] <<"\t";

        cout<<endl;

        cout<<"Codeword :"<<endl;

        for (inti = 1; i<= (n + check); i++)

            cout<<codeword[i] <<" ";

        cout<<endl;

    }

    intfindr()

    {

        for (inti = 1;; i++)

        {

            if (n + i + 1<= pow(2, i))

                returni;

        }

    }

    intsetParity(intx)

    {

        boolflag = true;

        intbit;

        if (x == 1)

        {

            bit = codeword[x + 2];

            for (intj = x + 3; j<= (n + check); j++)

            {

                if (j % 2)

                {

                    bit ^= codeword[j];

                }

            }

        }

        else

        {

            bit = codeword[x + 1];

            for (inti = x; i<= (n + check); i++)

            {

                if (flag)

                {

                    if (i == x || i == x + 1)

                        bit = codeword[x + 1];

                    else

                        bit ^= codeword[i];

                }

                if ((i + 1) % x == 0)

                    flag = !flag;

            }

        }

        if (parity == 'O' || parity == 'o')

            return !bit;

        else

            returnbit;

    }

    voidcorrect()

    {

        do

        {

            cout<<"Enter the received codeword : ";

            cin>>message;

        } while (message.find\_first\_not\_of("01") != string::npos);

        for (unsignedinti = 0; i<message.size(); i++)

        {

            if (message[i] == '1')

                codeword[i + 1] = 1;

            else

                codeword[i + 1] = 0;

        }

        detect();

    }

    voiddetect()

    {

        intposition = 0;

        cout<<"Parity Bits - ";

        for (inti = 0; i<check; i++)

        {

            boolflag = true;

            intx = pow(2, i);

            intbit = codeword[x];

            if (x == 1)

            {

                for (intj = x + 1; j<= (n + check); j++)

                {

                    if (j % 2)

                    {

                        bit ^= codeword[j];

                    }

                }

            }

            else

            {

                for (intk = x + 1; k<= (n + check); k++)

                {

                    if (flag)

                    {

                        bit ^= codeword[k];

                    }

                    if ((k + 1) % x == 0)

                        flag = !flag;

                }

            }

            cout<<"P"<<x<<": "<<bit<<"\t";

            if ((parity == 'E' || parity == 'e') &&bit == 1)

                position += x;

            if ((parity == 'O' || parity == 'o') &&bit == 0)

                position += x;

        }

        cout<<endl

             <<"Received Codeword :"<<endl;

        for (inti = 1; i<= (n + check); i++)

            cout<<codeword[i] <<" ";

        cout<<endl;

        if (position != 0)

        {

            cout<<"Error at bit : "<<position<<endl;

            codeword[position] = !codeword[position];

            cout<<"Corrected Codeword : "<<endl;

            for (inti = 1; i<= (n + check); i++)

                cout<<codeword[i] <<" ";

            cout<<endl;

        }

        else

            cout<<"No Error in Received code."<<endl;

        cout<<"Received Message is : ";

        for (inti = 1; i<= (n + check); i++)

            if ((i& (i - 1)) != 0)

                cout<<codeword[i] <<" ";

        cout<<endl;

    }

};

intmain()

{

    cout<<endl<<"\t \*\* PROGRAM TO IMPLEMENT HAMMING CODE \*\* "<<endl<<endl;

    charchoice;

    do

    {

        Hamminga;

        cout<<"At Sender's side : "<<endl;

        a.generate();

        cout<<endl

             <<"At Receiver's Side : "<<endl;

        a.correct();

        cout<<endl

             <<"Enter another code ? (Y/N) : ";

        cin>>choice;

        cout<<endl;

    } while (choice == 'y' || choice == 'Y');

    return0;

}

