PRACTICAL 1

Question:

Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.

Code:

#include <iostream>

using namespace std;

int main()

{

int mSize;

int message[255];

cout << "Enter Message Size: ";

cin >> mSize; cout << "Enter Message: ";

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

cin >> message[i];

int gSize;

int generator[64];

cout << "Enter Generator Size: ";

cin >> gSize; cout << "Enter Generator: ";

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

cin >> generator[i];

if (!(generator[0] == 1 && generator[gSize - 1] == 1))

{

cerr << "\nERROR: MSB and LSB of the Generator must be 1\n";

return -1;

}

cout << "\nSENDER\n======\n";

cout << "Message: ";

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

cout << message[i];

cout << endl;

cout << "Generator: ";

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

cout << generator[i];

cout << endl;

int codeword[mSize + (gSize - 1)];

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

codeword[i] = message[i];

for (int i = mSize; i < mSize + (gSize - 1); i++)

codeword[i] = 0;

int temp[mSize + (gSize - 1)];

for (int i = 0; i < mSize + (gSize - 1); i++)

temp[i] = codeword[i];

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

{

int j = 0, k = i;

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

while (j < gSize)

temp[k++] ^= generator[j++];

}

int crc[64];

for (int i = 0, j = mSize; i < (gSize - 1); i++, j++)

crc[i] = temp[j];

cout << "CRC: ";

for (int i = 0; i < (gSize - 1); i++)

cout << crc[i];

cout << endl;

for (int i = 0, j = mSize; i < (gSize - 1); i++, j++)

codeword[j] = crc[i];

cout << "Transmitted Codeword: ";

for (int i = 0; i < mSize + (gSize - 1); i++)

cout << codeword[i];

cout << endl;

cout << "\nNOISY CHANNEL SIMULATION\n========================\n";

int nb, n;

cout << "Enter Number of Bits to Flip: ";

cin >> nb;

if (nb > 0 && nb < mSize + (gSize - 1))

{

if (nb == 0)

cout << "Codeword Not Changed.\n";

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

{

cout << "Enter Bit Position to Flip: ";

cin >> n;

if (n > 0 && n < mSize + (gSize - 1))

codeword[n - 1] = codeword[n - 1] == 0 ? 1 : 0;

else

cout << "Invalid Position. Codeword Not Changed.\n";

}

}

else

cout << "Invalid Request. Codeword Not Changed.\n";

cout << "\nRECEIVER\n========\n";

cout << "Received Codeword: ";

for (int i = 0; i < mSize + (gSize - 1); i++)

cout << codeword[i]; cout << endl;

int temp2[mSize + (gSize - 1)];

for (int i = 0; i < mSize + (gSize - 1); i++)

temp2[i] = codeword[i];

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

{

int j = 0, k = i;

if (temp2[k] >= generator[j])

while (j < gSize)

temp2[k++] ^= generator[j++];

}

int rem[64];

for (int i = mSize, j = 0; i < mSize + (gSize - 1); i++, j++)

rem[j] = temp2[i];

cout << "Remainder: ";

for (int i = 0; i < (gSize - 1); i++)

cout << rem[i];

cout << endl;

int flag = false;

for (int i = 0; i < (gSize - 1); i++)

if (rem[i] != 0) flag = true;

// Declare Result

cout << endl;

if (!flag)

cout << "TRANSMISSION OK!" << endl;

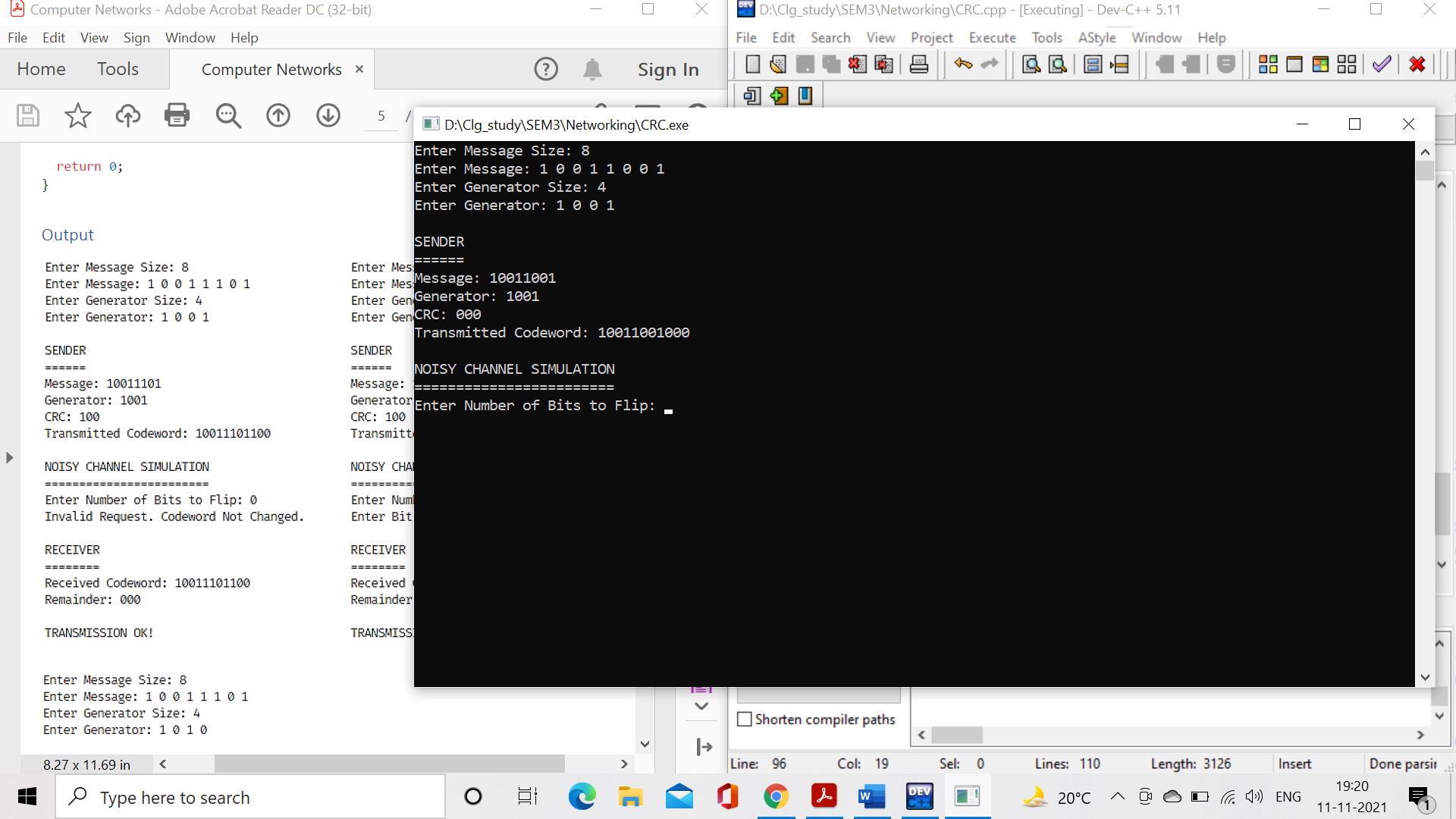
else

cout << "TRANSMISSION ERROR DETECTED!" << endl;

return 0;

}

Output:



PRACTICAL 2:

Question: Simulate and implement stop and wait protocol for noisy channel.

Code:

#include <cstdio>

#include <string>

#define MAX\_PKT 4

using namespace std;

typedef enum

{

dat, ack, nak

}

frameKind;

typedef enum

{

wait, frameArrival

}

eventType;

typedef struct

{

unsigned char data[MAX\_PKT];

}

packet;

typedef struct

{

packet \*info;

frameKind kind;

unsigned int seq;

unsigned int ack;

}

frame;

class Protocol

{

public:

int sentSeq;

int receivedSeq;

packet dataPacket;

frame senderFrame,

receiverFrame;

Protocol()

{

sentSeq = receivedSeq = -1;

}

int waitForEvent(eventType e)

{

return e == frameArrival;

}

string showkind(frameKind k)

//display the event type

{

switch (k)

{

case dat:

return "data";

break;

case ack:

return "ack";

break;

case nak:

return "nak";

break;

}

return "";

}

void fromNetworkLayer(packet &i)

{

printf("\nEncapsulating Packet<data='%s'> ...", i.data);

senderFrame.seq = ++sentSeq;

senderFrame.kind = dat;

senderFrame.info = &i;

}

void toPhysicalLayer(frame &f)

{

if (f.kind == dat)

printf("\nSending DataFrame<kind=%s, sequence=%i> to Physical Layer ...",showkind(f.kind).c\_str(), f.seq);

else

printf("\nSending ControlFrame<kind=%s, ack=%i> to Physical Layer ...", showkind(f.kind).c\_str(), f.ack);

}

void fromPhysicalLayer(frame &f)

{

printf("\nReceived DataFrame<kind=%s, sequence=%i> from Physical Layer ...", showkind(f.kind).c\_str(), f.seq);

printf("\nValidating Sequence Number ... ");

if (receivedSeq != f.seq)

printf("\nDecapsulating Frame ...");

else

{

printf("\nDuplicate Frame Encountered ...");

printf("\nDiscarding Frame ...");

}

}

void toNetworkLayer(packet &p)

{

printf("\nSending Packet<data='%s'> to Network Layer ...", p.data);

receivedSeq = senderFrame.seq;

receiverFrame.seq = 0;

receiverFrame.kind = ack;

receiverFrame.ack = senderFrame.seq + 1;

}

};

#include <cstring>

#include <cstdlib>

#include "protocol.cpp"

class stopAndWait : public Protocol

{

public: string buf;

eventType event;

bool flag, start;

int coeff, count, len, lim;

stopAndWait(string s, int t)

{

buf = s;

coeff = t;

lim = 1e6;

flag = false;

start = false;

count = 0;

}

void sender();

void receiver();

};

void stopAndWait::sender()

{

if (!start)

{

lim = buf.length() % MAX\_PKT == 0 ? buf.length() / MAX\_PKT : buf.length() / MAX\_PKT + 1;

printf("\nDividing Data into Groups of %d-bytes Each ...", MAX\_PKT);

start = !start;

}

printf("\n\nSENDER\n======");

if (count > 0)

{

if (count % coeff == 0)

{

printf("\nERROR: SIMULATED TIMEOUT ...");

flag = true;

}

else

{

printf("\nReceived ControlFrame<kind=%s, ack=%d> ...", showkind(receiverFrame.kind).c\_str(), receiverFrame.ack);

if (flag)

{

count--;

flag = !flag;

}

}

if (receiverFrame.kind == nak || flag)

{

printf("\nResending Previous Frame ...");

count--;

flag = true;

}

}

if (count == lim)

{

printf("\n\nData '%s' Sent Successfully ...", buf.c\_str());

exit(0);

}

while (count < lim)

{

while (event != wait)

{

if (!flag)

{

printf("\nEncapsulating Data D%d into a Packet ...", count + 1);

for (int i = 0; i < MAX\_PKT; i++)

dataPacket.data[i] = buf[i + count \* MAX\_PKT];

printf("\nPassing Packet to Data Link Layer ...");

event = frameArrival;

}

if (waitForEvent(event))

{

if (!flag) fromNetworkLayer(dataPacket);

toPhysicalLayer(senderFrame);

event = wait;

}

receiver();

}

}

}

void stopAndWait::receiver()

{

printf("\n\nRECEIVER\n========");

if (event == wait)

{

fromPhysicalLayer(senderFrame);

if (!flag)

{

toNetworkLayer(dataPacket);

count++;

}

else

count += 2;

toPhysicalLayer(receiverFrame);

event = frameArrival;

sender();

}

}

int main()

{

char temp[50];

printf("Enter Data: ");

scanf("%s", temp);

int temp2;

printf("Enter Simulation Noise (>=2): ");

scanf("%i", &temp2);

stopAndWait \*obj = new stopAndWait(string(temp), temp2);

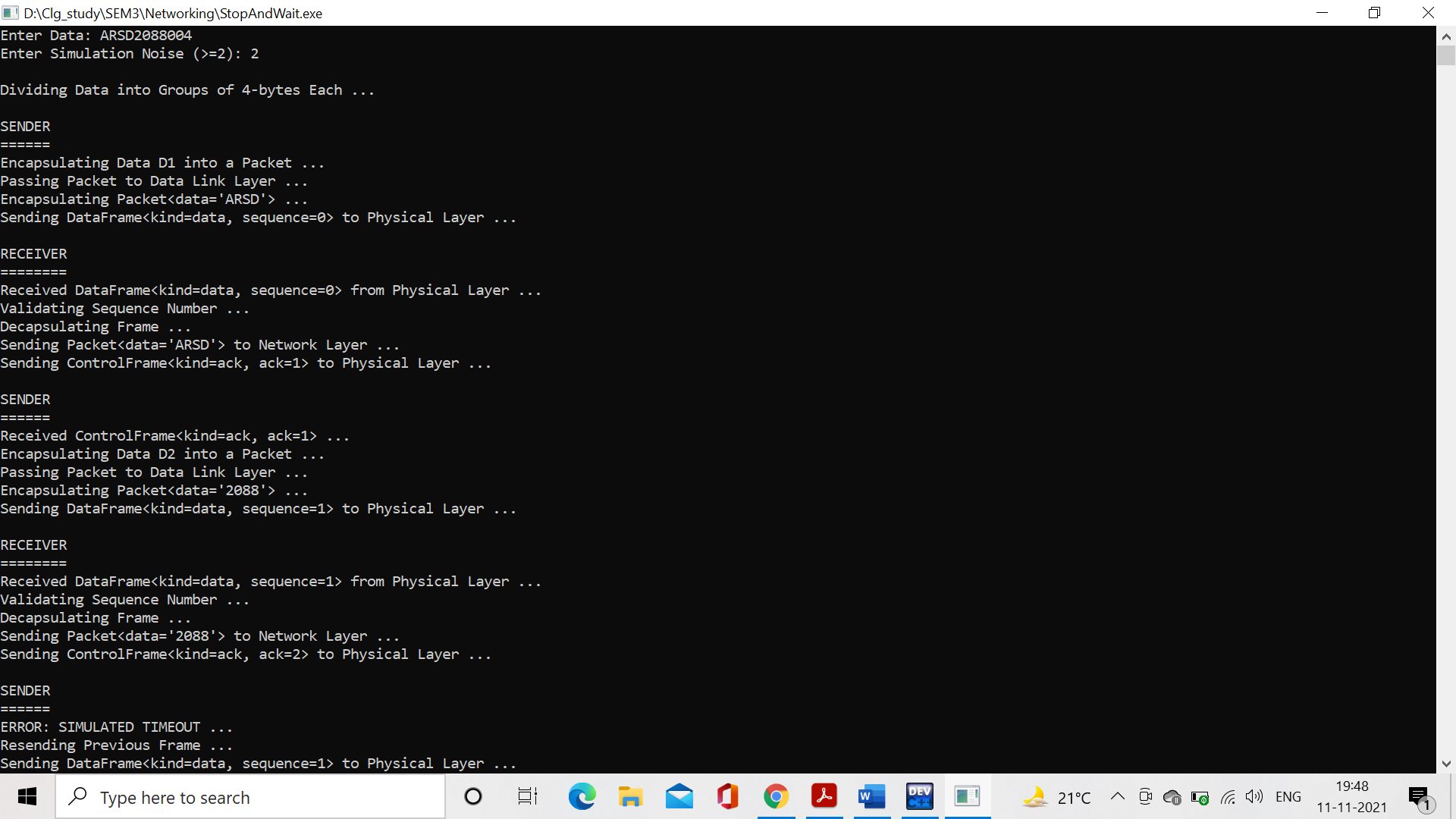
obj->sender();

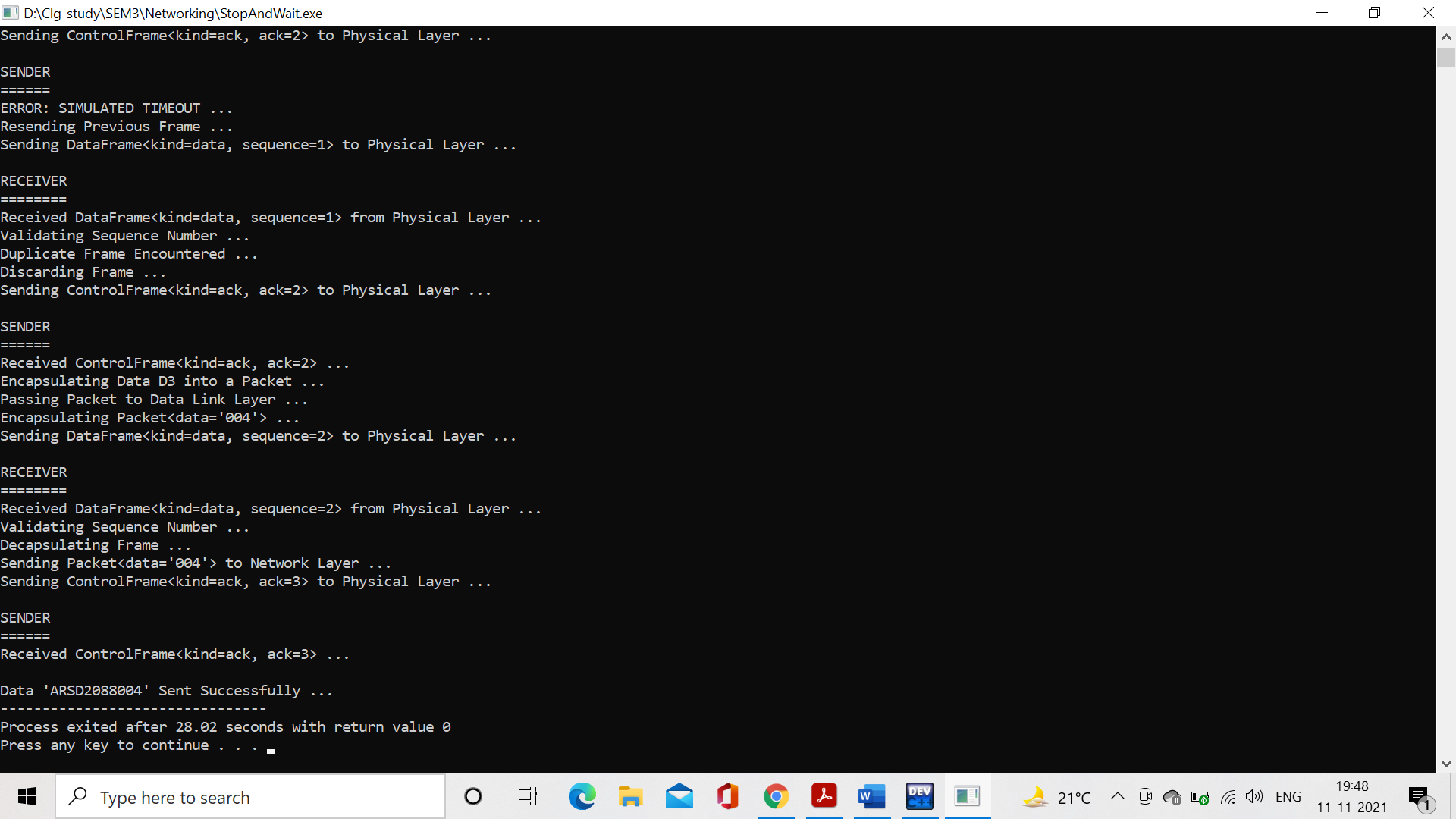
delete obj;

return 0;

}

Output:





PRACTICAL 3

Question: Simulate and implement distance vector routing algorithm

Code:

#include <cstdio>

#include <string>

#include <iostream>

#define MAX\_PKT 5

using namespace std;

typedef enum

{

dat, ack, nak

}

frameKind;

typedef enum

{

timeout, checksumError, frameArrival, networkLayerReady

} eventType;

typedef struct

{

unsigned char data;

} packet;

typedef struct

{

packet \*info;

frameKind kind;

unsigned int seq;

unsigned int ack;

} frame;

class Protocol

{

public: eventType event;

bool noNak, errorDetected;

int MAX\_SEQ, flag, err, buf;

int frameExpected, frameToSend;

packet dataPacket;

frame senderFrame, receiverFrame;

Protocol()

{

noNak = true;

err = buf = -1;

flag = frameToSend = frameExpected = 0;

errorDetected = false;

}

int waitForEvent(eventType e)

{

return e == frameArrival;

}

string showkind(frameKind k)

{

switch (k)

{

case dat:

return "data";

break;

case ack:

return "ack";

break;

case nak:

return "nak";

break;

}

return "";

}

void fromNetworkLayer(packet &i)

{

printf("\nEncapsulating Packet<data='%c'> ...", i.data);

senderFrame.seq = frameToSend; senderFrame.kind = dat;

senderFrame.info = &i;

frameToSend = (frameToSend + 1) % (MAX\_SEQ + 1);

}

void toPhysicalLayer(frame &f)

{

if (event == timeout)

{

cout << "\nTimeout period expired. Resending frame with sequence no. " << err;

f.seq = err;

err = -1;

frameToSend = (err + 1) % (MAX\_SEQ + 1);

event = frameArrival;

}

else if (f.kind == dat)

printf("\nSending DataFrame<kind=%s, sequence=%i> to Physical Layer ...", showkind(f.kind).c\_str(), f.seq);

else

{

if (err != -1)

{

if (!noNak)

{

f.kind = nak;

f.ack = err;

noNak = true;

}

else

{

f.kind = ack;

f.ack = err - 1;

}

}

else if (buf != -1)

{

f.ack = buf;

frameExpected = (buf + 1) % (MAX\_SEQ + 1);

frameToSend = frameExpected;

buf = -1;

}

printf("\nSending ControlFrame<kind=%s, ack=%i> to Physical Layer ...", showkind(f.kind).c\_str(), f.ack);

}

}

void toNetworkLayer(packet &p)

{

printf("\nSending Packet<data='%c'> to Network Layer ...", p.data);

receiverFrame.seq = frameToSend - 1;

receiverFrame.kind = ack;

receiverFrame.ack = frameExpected;

frameExpected = (frameExpected + 1) % (MAX\_SEQ + 1);

}

void fromPhysicalLayer(frame &f)

{

printf("\nReceived DataFrame<kind=%s, sequence=%i> from Physical Layer ...", showkind(f.kind).c\_str(), f.seq);

printf("\nValidating Sequence Number ... ");

{

if (frameExpected == f.seq)

{

if (f.seq == 1 && flag == 0)

// Error Simulation

{

cout << "\nError in received frame ...";

flag = 1;

noNak = false;

errorDetected = true;

err = f.seq;

}

else

{

printf("\nDecapsulating Frame ...");

noNak = true;

toNetworkLayer(dataPacket);

}

}

else

{

printf("\nFrame out of order. Storing in buffer ...");

buf = f.seq;

}

}

}

};

#include <cstring>

#include <cstdlib>

#include <cmath>

#include "protocol2.cpp"

void getch()

{

cin.ignore();

cin.get();

return;

}

void clrscr()

{

#ifdef \_WIN32

system("cls");

#elif \_\_unix\_\_

system("clear");

#endif

return;

}

class selectiveRepeatSlidingWindow : public Protocol

{

public: string in\_buf;

selectiveRepeatSlidingWindow(int n, string s)

{

MAX\_SEQ = n;

in\_buf = s;

}

void sender();

void receiver();

};

void selectiveRepeatSlidingWindow::sender()

{

event = frameArrival;

printf("\n\nSENDER\n======");

if (frameToSend == (err + (MAX\_SEQ / 2)) % (MAX\_SEQ + 1) && errorDetected == true && err >= 0)

{

event = timeout;

frameToSend = err; errorDetected = 0;

}

else if (frameToSend == MAX\_SEQ && frameToSend != frameExpected && errorDetected == true)

{

fromNetworkLayer(dataPacket);

frameToSend = frameExpected;

errorDetected = false;

}

else if (event == frameArrival)

{

printf("\nEncapsulating Data '%c' into a Packet ...", in\_buf[frameToSend]);

dataPacket.data = in\_buf[frameToSend];

printf("\nPassing Packet to Data Link Layer ...");

fromNetworkLayer(dataPacket);

}

toPhysicalLayer(senderFrame);

receiver();

}

void selectiveRepeatSlidingWindow::receiver()

{

printf("\n\nRECEIVER\n========");

fromPhysicalLayer(senderFrame);

toPhysicalLayer(receiverFrame);

getch();

clrscr();

sender();

}

int main()

{

int n;

cout << "\nEnter bits needed to identify window: ";

cin >> n;

char temp[50];

printf("Enter Data: ");

scanf("%s", temp);

selectiveRepeatSlidingWindow \*obj = new selectiveRepeatSlidingWindow( pow(2, n) - 1, string(temp));

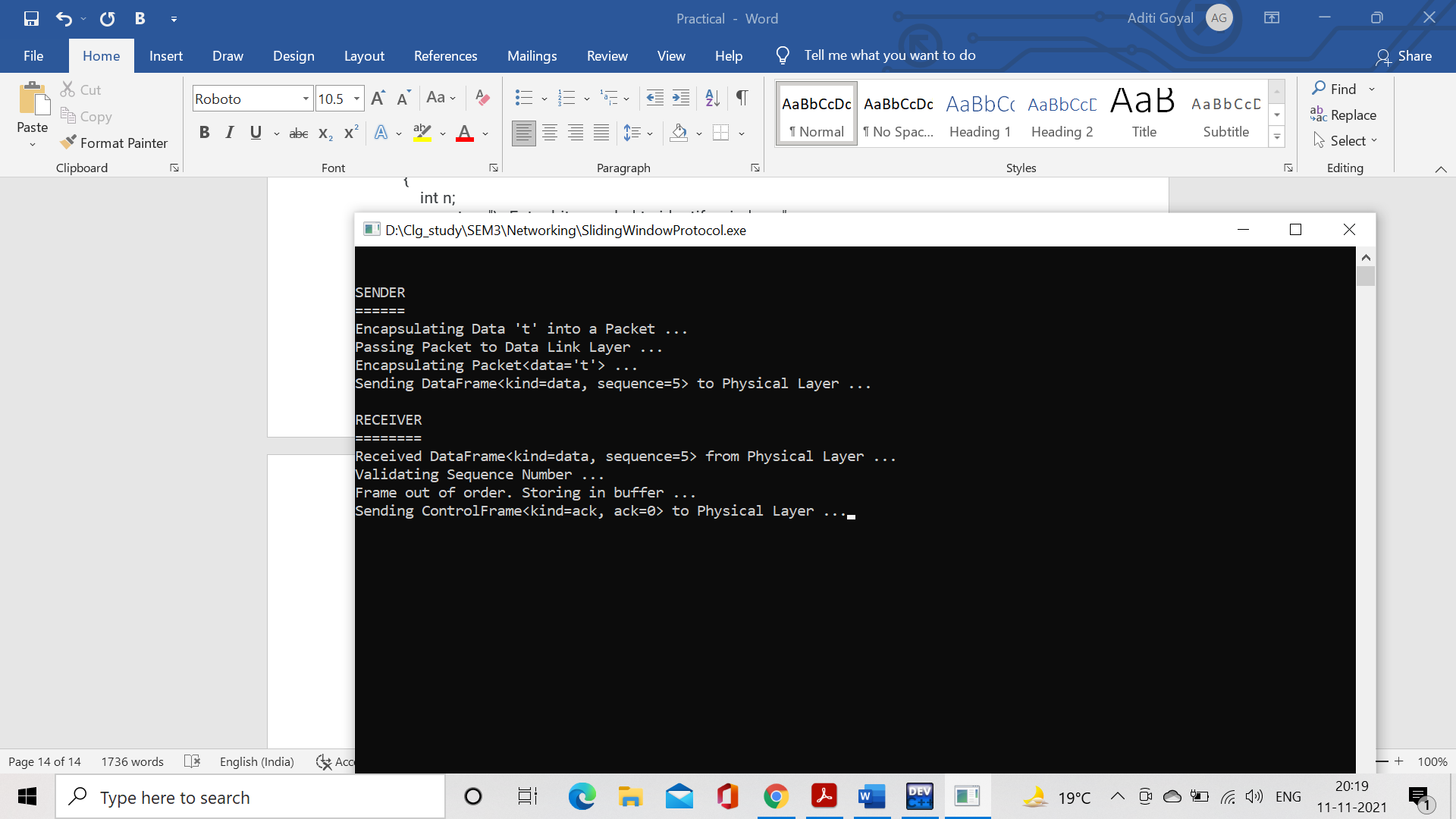
obj->sender();

delete obj;

return 0;

}

Output:



PRACTICAL 4

Question: Simulate and implement Dijkstra algorithm for shortest path routing.

Code:

#include <cstdio>

#include <climits>

#include <iomanip>

#include <iostream>

#define MAX\_NODES 10

using namespace std;

class Graph

{

public:

int edges;

int vertices;

int nextHop[MAX\_NODES][MAX\_NODES];

int distances[MAX\_NODES][MAX\_NODES];

int adjMatrix[MAX\_NODES][MAX\_NODES];

void input(int v, int e)

{

edges = e;

vertices = v;

// initialize the adjacency matrix

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

for (int j = 0; j < v; j++)

adjMatrix[i][j] = 0;

int src, dest, weight;

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

{

cout << "\nEDGE " << (i + 1) << "\n======\n";

cout << "Enter Source: ";

cin >> src;

cout << "Enter Destination: ";

cin >> dest;

cout << "Enter Weight: ";

cin >> weight;

adjMatrix[src - 1][dest - 1] = weight;

adjMatrix[dest - 1][src - 1] = weight;

}

}

void display()

{

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

{

for (int j = 0; j < vertices; j++)

cout << setw(5) << adjMatrix[i][j] << " ";

cout << endl;

}

}

void distanceVector()

{

// populate additional data structures

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

for (int j = 0; j < vertices; j++)

{

if (i == j)

distances[i][j] = 0;

else if (adjMatrix[i][j] == 0)

distances[i][j] = INT\_MAX / 2;

else

distances[i][j] = adjMatrix[i][j];

nextHop[i][j] = -1;

}

cout << "Initial Distance Matrix\n";

cout << "======================\n";

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

{

for (int j = 0; j < vertices; j++)

if (distances[i][j] == INT\_MAX / 2)

cout << setw(5) << right << "INF" << " ";

else cout << setw(5) << distances[i][j] << " ";

cout << endl;

}

bool flag;

do

{

flag = false;

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

for (int j = 0; j < vertices; j++)

for (int k = 0; k < vertices; k++)

if (distances[i][j] > (distances[i][k] + distances[k][j]))

{

distances[i][j] = distances[j][i] = distances[i][k] + distances[k][j];

nextHop[i][j] = nextHop[j][i] = k;

flag = true;

}

} while (flag);

cout << "\nFinal Distance Matrix\n";

cout << "======================\n";

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

{

for (int j = 0; j < vertices; j++)

cout << setw(5) << distances[i][j] << " ";

cout << endl;

}

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

{

cout << "\nRouting Table for Router " << (i + 1) << ":";

cout << "\nDest Router \t Via \t\t Distance";

cout << "\n=========== \t ======== \t ========\n";

for (int j = 0; j < vertices; j++)

{

if (i == j)

continue;

cout << (j + 1) << " \t\t ";

if (nextHop[i][j] == -1)

cout << "-";

else

cout << (nextHop[i][j] + 1);

cout << " \t\t " << distances[i][j] << endl;

}

}

}

};

int main()

{

int v, e;

int link1, link2;

Graph graph;

cout << "Enter No. of Nodes: ";

cin >> v;

cout << "Enter No. of Edges: ";

cin >> e;

graph.input(v, e);

cout << "\nGRAPH\n=====\n";

graph.display();

cout << endl;

graph.distanceVector();

cout << "\nSimulating Link Failure\n";

cout << "=======================\n";

cout << "Enter Routers to Break Link Between: ";

cin >> link1 >> link2;

cout << endl;

graph.adjMatrix[link1 - 1][link2 - 1] = graph.adjMatrix[link2 - 1][link1 - 1] = 0;

graph.distanceVector();

return 0;

}

Output:

