**Registration Number: 19BCE2119**

**Name: Gaurav Kumar Singh**

**Course: Network and Communication CSE1004**

**Digital Assignment 3**

**Stop and Wait ARQ protocol, Go Back-N ARQ protocol, Selective Repeat ARQ protocol, IPv4 Classless Addressing**

**STOP AND WAIT**

#include <iostream>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include <time.h>

#include <windows.h>

using namespace std;

struct frame {

int frame\_no;

int reciever\_flag; //activates when frame recieved by reciever

int sender\_flag; //activates when ACK recieved by sender

};

int RNG(int lower, int upper) {

int num = (rand() % (upper - lower + 1)) + lower;

return num;

}

void init() {

srand(time(NULL));

}

void input(struct frame arr[], int n) {

for (int i = 0; i < n; i++) {

arr[i].frame\_no = i + 1;

arr[i].reciever\_flag = 0;

arr[i].sender\_flag = 0;

}

}

void SAW(struct frame arr[], int n, int timer) {

int frame\_status = -1;

int ACK\_status = -1;

for (int i = 0; i < n; i++) {

cout << "SENDER: Sending frame " << i + 1 << "..." << endl;

Sleep(timer);

frame\_status = RNG(0, 1);

ACK\_status = RNG(0, 1);

while (frame\_status != 1) {

Sleep(timer);

cout << "SENDER: Acknowledgement not recieved from reciever end...\t Resending frame\n";

Sleep(timer);

frame\_status = RNG(0, 1);

}

cout << "RECIEVER: Frame recieved...\t Sending Acknowledgement\n";

arr[i].reciever\_flag = 1;

while (ACK\_status != 1) {

cout << "SENDER: Acknowledgement not recieved from reciever end...\t Resending frame\n";

Sleep(timer);

cout << "RECIEVER: Duplicate frame recieved...\t Sending Acknowledgement again\n";

Sleep(timer);

ACK\_status = RNG(0, 1);

}

arr[i].sender\_flag = 1;

cout << "SENDER: Acknowledgement recieved for frame " << i + 1 << "\nTransmission completed for frame" << i + 1 << "\n\n";

}

}

int main() {

init();

int n;

int timer = 100;

cout << "Enter Number of frames: ";

cin >> n;

cout << "\n";

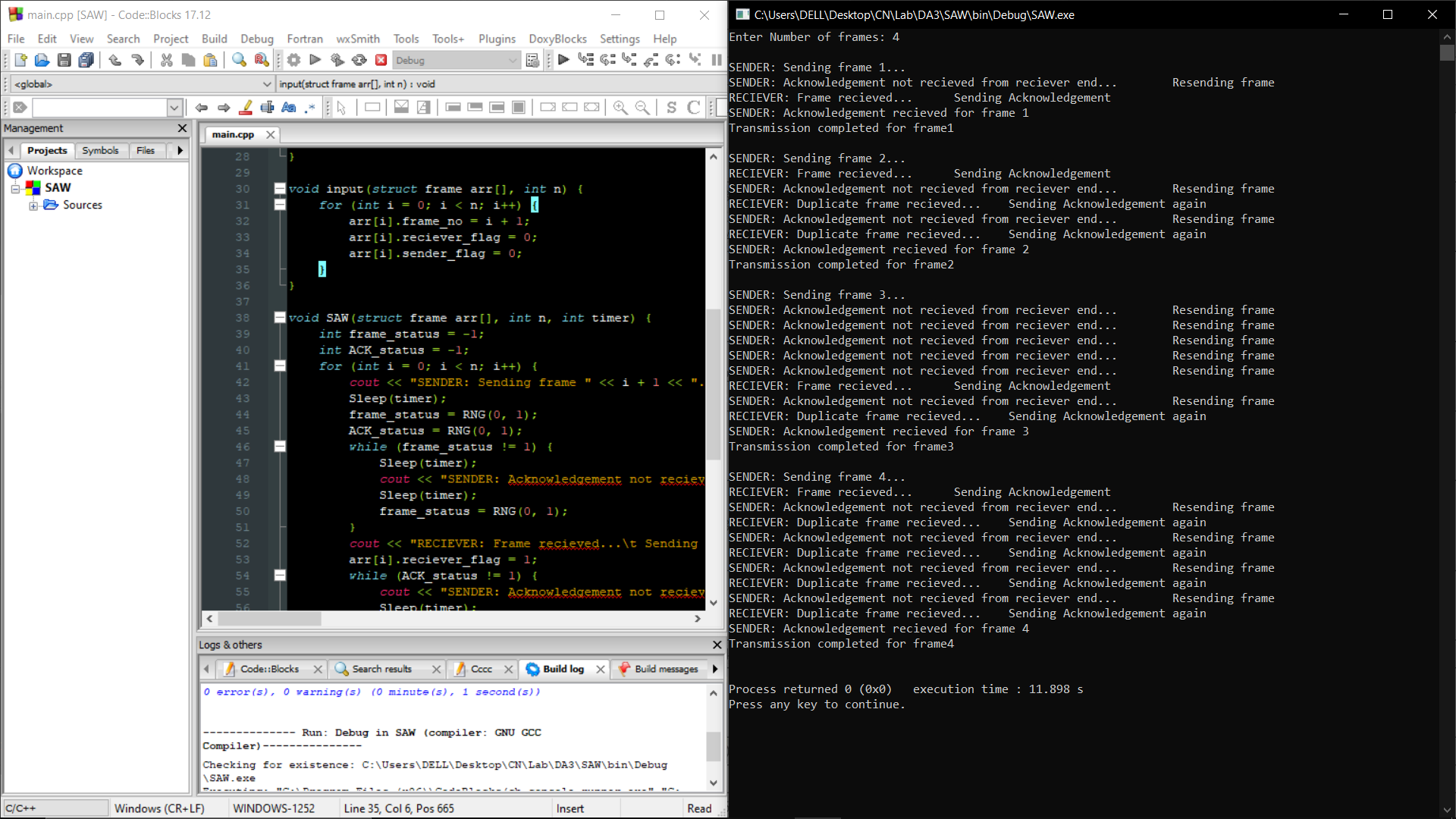
struct frame arr[100];

input(arr, n);

SAW(arr, n, timer);

return 0;

}



**GO BACK-N**

#include <iostream>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include <time.h>

#include <windows.h>

using namespace std;

struct frame {

int frame\_no;

int reciever\_flag; //activates when frame recieved by reciever

int sender\_flag; //activates when ACK recieved by sender

};

int RNG(int lower, int upper) {

int num = (rand() % (upper - lower + 1)) + lower;

return num;

}

void init() {

srand(time(NULL));

}

void input(struct frame arr[], int n, int window\_size) {

for (int i = 0; i < n; i++) {

arr[i].frame\_no = i + 1;

arr[i].reciever\_flag = 0;

arr[i].sender\_flag = 0;

}

for (int i = n; i< n+window\_size; i++){

arr[i].frame\_no = -1;

arr[i].reciever\_flag = 1;

arr[i].sender\_flag = 1;

}

}

void GBN(struct frame arr[], int n, int window\_size,int timer) {

int i=0;

for (int j=i; j<i+window\_size; j++){

cout<<"SENDER: Sending frame "<<j+1<<"\n";

}

Sleep(timer);

while (i<n){

arr[i].reciever\_flag=RNG(0,1);

Sleep(timer);

while(arr[i].reciever\_flag!=1){

cout<<"SENDER: Acknowledgement not Recieved for frame "<<i+1<<"... Resending entire window\n";

for (int j=i; j<i+window\_size; j++){

cout<<"SENDER: Resending Sending frame "<<j+1<<"\n";

}

arr[i].reciever\_flag=RNG(0,1);

}

cout<<"RECIEVER: Frame "<<i+1<<" Recieved... Sending Acknowledgement\n";

arr[i].sender\_flag=RNG(0,1);

while(arr[i].sender\_flag!=1){

cout<<"SENDER: Acknowledgement not Recieved for frame "<<i+1<<"... Resending entire window\n";

for (int j=i; j<i+window\_size; j++){

cout<<"SENDER: Resending Sending frame "<<j+1<<"\n";

}

Sleep(timer);

arr[i].sender\_flag=RNG(0,1);

cout<<"RECIEVER: Duplicate data detected... Resending Acknowledgement\n";

}

cout<<"SENDER: Acknowledgement recieved... frame "<<i+1<<" transmitted...\nSHIFTING WINDOW\n";

i++;

if (arr[i+window\_size].frame\_no != -1){

cout<<"SENDER: Sending frame "<<i+window\_size<<"\n";

}

}

}

int main() {

init();

int timer = 100;

int n;

int window\_size;

cout << "Enter Window Size: ";

cin>> window\_size;

cout << "\n";

cout << "Enter Number of frames: ";

cin >> n;

cout << "\n";

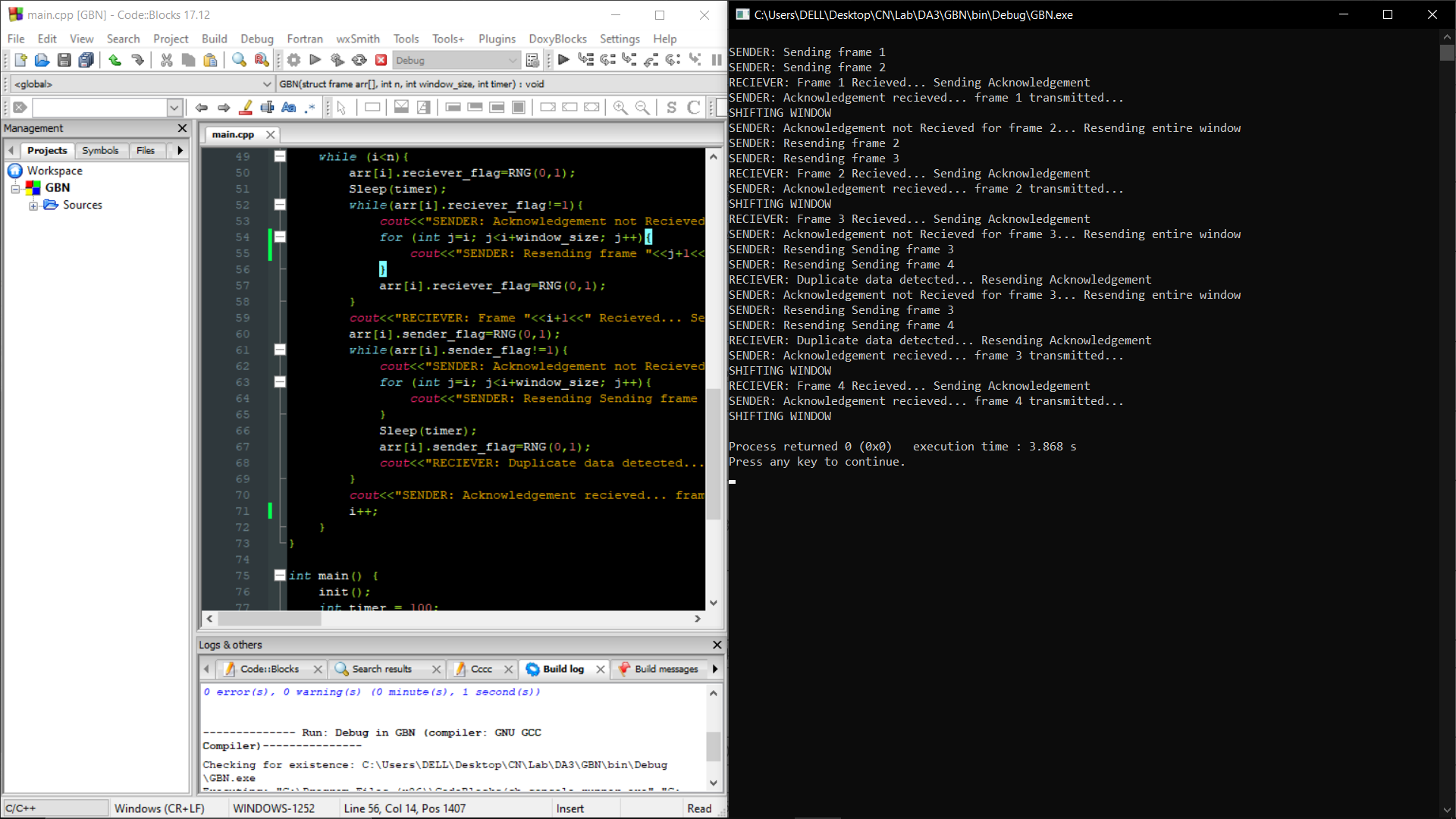
struct frame arr[100];

input(arr, n, window\_size);

GBN(arr, n, window\_size, timer);

return 0;

}



**SELECTIVE REPEAT**

#include <iostream>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include <time.h>

#include <windows.h>

using namespace std;

struct frame {

int frame\_no;

int reciever\_flag; //activates when frame recieved by reciever

int sender\_flag; //activates when ACK recieved by sender

int dup;

};

int RNG(int lower, int upper) {

int num = (rand() % (upper - lower + 1)) + lower;

return num;

}

void init() {

srand(time(NULL));

}

void input(struct frame arr[], int n, int window\_size) {

for (int i = 0; i < n; i++) {

arr[i].frame\_no = i + 1;

arr[i].reciever\_flag = 0;

arr[i].sender\_flag = 0;

arr[i].dup=0;

}

for (int i = n; i< n+window\_size; i++){

arr[i].frame\_no = -1;

arr[i].reciever\_flag = 1;

arr[i].sender\_flag = 1;

}

}

int check\_empty (struct frame arr[]) {

if (arr[0].frame\_no==-1){

return 0; //0 means empty

}

else {

return 1; //1 means not empty

}

}

struct frame Dequeue(struct frame arr[]){

struct frame temp;

if (arr[0].frame\_no != -1){

temp=arr[0];

}

int i=0;

while (arr[i].frame\_no != -1){

arr[i]=arr[i+1];

i++;

}

return temp;

};

void Enqueue(struct frame buffer[], struct frame temp){

int i=0;

int j=0;

while (buffer[i].frame\_no != -1){

i++;

}

buffer[i]=temp;

}

void SR(struct frame arr[], int n, int window\_size,int timer) {

int i=0;

int counter=0;

struct frame buffer[100];

struct frame ready[window\_size+1];

input(ready, 0, window\_size+1);

struct frame temp;

input(buffer, 0, 100);

for (int j=0; j<window\_size; j++){

cout<<"SENDER: Sending frame "<<j+1<<"\n";

ready[j]=arr[j];

counter++;

}

Sleep(timer);

while (check\_empty(ready) || check\_empty(buffer) || i<n){

if (!check\_empty(ready)){

if(check\_empty(buffer)){

Enqueue(ready, Dequeue(buffer));

}

else if(counter!=n && !check\_empty(buffer)){

Enqueue(ready, arr[counter]);

counter++;

}

}

ready[0].reciever\_flag=RNG(0,1);

Sleep(timer);

if (ready[0].reciever\_flag){

if (ready[0].dup==1){

cout<<"RECIEVER: Duplicate data found for frame "<<ready[0].frame\_no<<"... Resending Acknowledgement\n";

}

else {

cout<<"RECIEVER: Frame "<<ready[0].frame\_no<<" Recieved... Sending Acknowledgement\n";

ready[0].dup=1;

}

temp=Dequeue(ready);

if(check\_empty(buffer)){

Enqueue(ready, Dequeue(buffer));

}

else if(counter!=n && !check\_empty(buffer)){

Enqueue(ready, arr[counter]);

counter++;

}

Sleep(timer);

temp.sender\_flag=RNG(0,1);

if (temp.sender\_flag){

cout<<"SENDER: Acknowledgement for frame "<<temp.frame\_no<<" Recieved... Transmission Completed\n";

}

else{

cout<<"SENDER: Acknowledgement for frame "<<temp.frame\_no<<" Not recieved... Queuing frame for retransmission.\n";

Enqueue(buffer, temp);

}

}

else {

cout<<"RECIEVER: Corrupted data recieved for frame "<<ready[0].frame\_no<<"... Sending Negative ACK.\n";

Enqueue(buffer, Dequeue(ready));

if(check\_empty(buffer)){

Enqueue(ready, Dequeue(buffer));

}

else if(counter!=n && !check\_empty(buffer)){

Enqueue(ready, arr[counter]);

counter++;

}

}

i++;

}

}

void display(struct frame arr[], int n){

for (int i=0; i<n; i++){

cout<<arr[i].frame\_no<<"\t";

}

}

int main() {

init();

int timer = 100;

int n;

int window\_size;

cout << "Enter Window Size: ";

cin>> window\_size;

cout << "\n";

cout << "Enter Number of frames: ";

cin >> n;

cout << "\n";

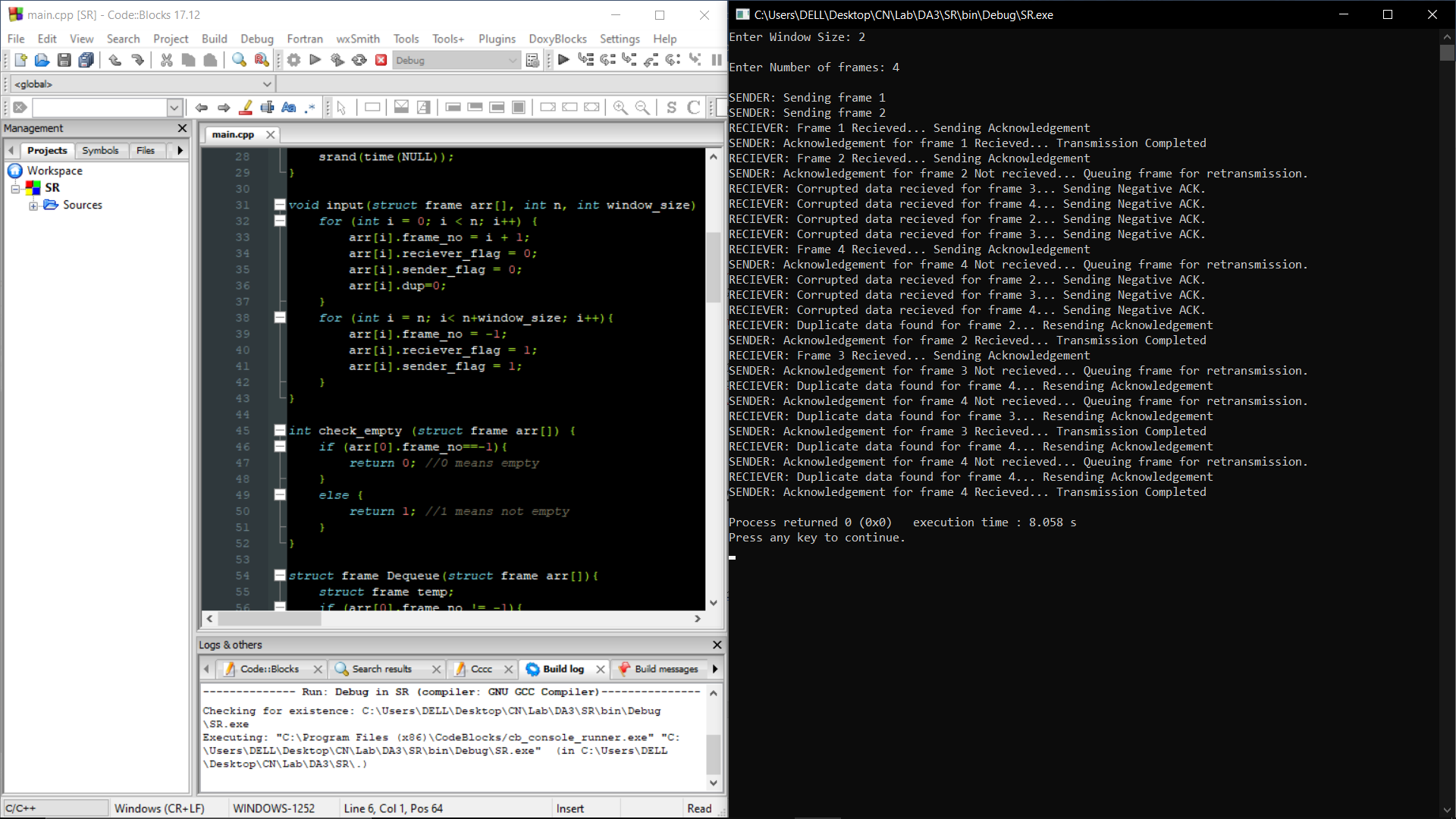
struct frame arr[100];

input(arr, n, window\_size);

SR(arr, n, window\_size, timer);

return 0;

}



**IPV4 CLASSLESS ADDRESSING**

import re

import numpy as np

def intToBinary(var):

binary=bin(var).split("0b")[1]

while(len(binary)<8):

binary='0'+binary

return binary

def intToBinary32(var):

binary=bin(var).split("0b")[1]

while(len(binary)<32):

binary='0'+binary

return binary

def binToInteger(var):

return (int(var,2))

def display(temp):

var=temp

arr=[]

for i in range(0,32,8):

arr.append(binToInteger(var[i:i+8]))

return arr

def IPv4Format(arr):

ipv4=str(arr[0])

for i in range(1,4,1):

ipv4=ipv4+'.'+str(arr[i])

return ipv4

def complement(var):

temp=''

for i in range(len(var)):

if var[i]=='0':

temp=temp+'1'

else:

temp=temp+'0'

return temp

def binaryAND( var1, var2):

ans=''

for i in range(len(var1)):

if var1[i]=='1' and var2[i]=='1':

ans=ans+'1'

else:

ans=ans+'0'

return ans

def binaryOR( var1, var2):

ans=''

for i in range(len(var1)):

if var1[i]=='1' or var2[i]=='1':

ans=ans+'1'

else:

ans=ans+'0'

return ans

nsubnet = int(input("Enter number of subnets in the network: "))

subnet\_requirements\_arr =[]\*nsubnet

for i in range (nsubnet):

print("Enter number of customers in subnet ", i+1, end=" : ")

ncustomers= int(input())

print("Enter number of IP addresses required per customer ", end=" : ")

nIPs= int(input())

subnet\_requirements\_arr.append([ncustomers, nIPs])

start\_IP=input("Enter starting IP address: ")

segments=re.split('\.|/', start\_IP)

try:

segments[4]

except IndexError:

errmask=input("Please enter mask of the IP address provided in CIDR notation(IP/mask or /mask for just mask): ")

segments.append(errmask.split('/')[-1])

binary\_IP=''

for i in range(4):

binary\_IP=str(binary\_IP)+str(intToBinary(int(segments[i])))

mask=''

for i in range (int(segments[4])):

mask=mask+'1'

while len(mask)<32:

mask=mask+'0'

init\_mask=[]

for i in range(0,32,8):

init\_mask.append(binToInteger(mask[i:i+8]))

print("Mask: ",init\_mask)

start\_IP=intToBinary32(binToInteger(binary\_IP)&binToInteger(mask))

init\_start\_IP=display(start\_IP)

print("Start IP: ",init\_start\_IP)

first\_address=''

for i in range(4):

first\_address=first\_address+intToBinary(init\_start\_IP[i])

print(first\_address)

for i in range(nsubnet):

print("\n\n\nSubnet ",i+1,"\n")

nums=range(0,33,1)

portbits=np.log2(subnet\_requirements\_arr[i][1])

if portbits not in nums:

portbits=int(portbits)+1

nmask=int(32-portbits)

mask=''

for k in range(nmask):

mask=mask+'1'

while len(mask)<32:

mask=mask+'0'

print("CUSTOMER\t\tSTARTING IP\t\tENDING IP")

for j in range(subnet\_requirements\_arr[i][0]):

print("Customer ", j+1,end='\t\t')

print(IPv4Format(display(binaryAND(mask, first\_address)))+'/'+str(nmask),end='\t\t')

print(IPv4Format(display(binaryOR(complement(mask),first\_address)))+'/'+str(nmask))

first\_address=intToBinary32(binToInteger(binaryOR(complement(mask),first\_address))+1)

