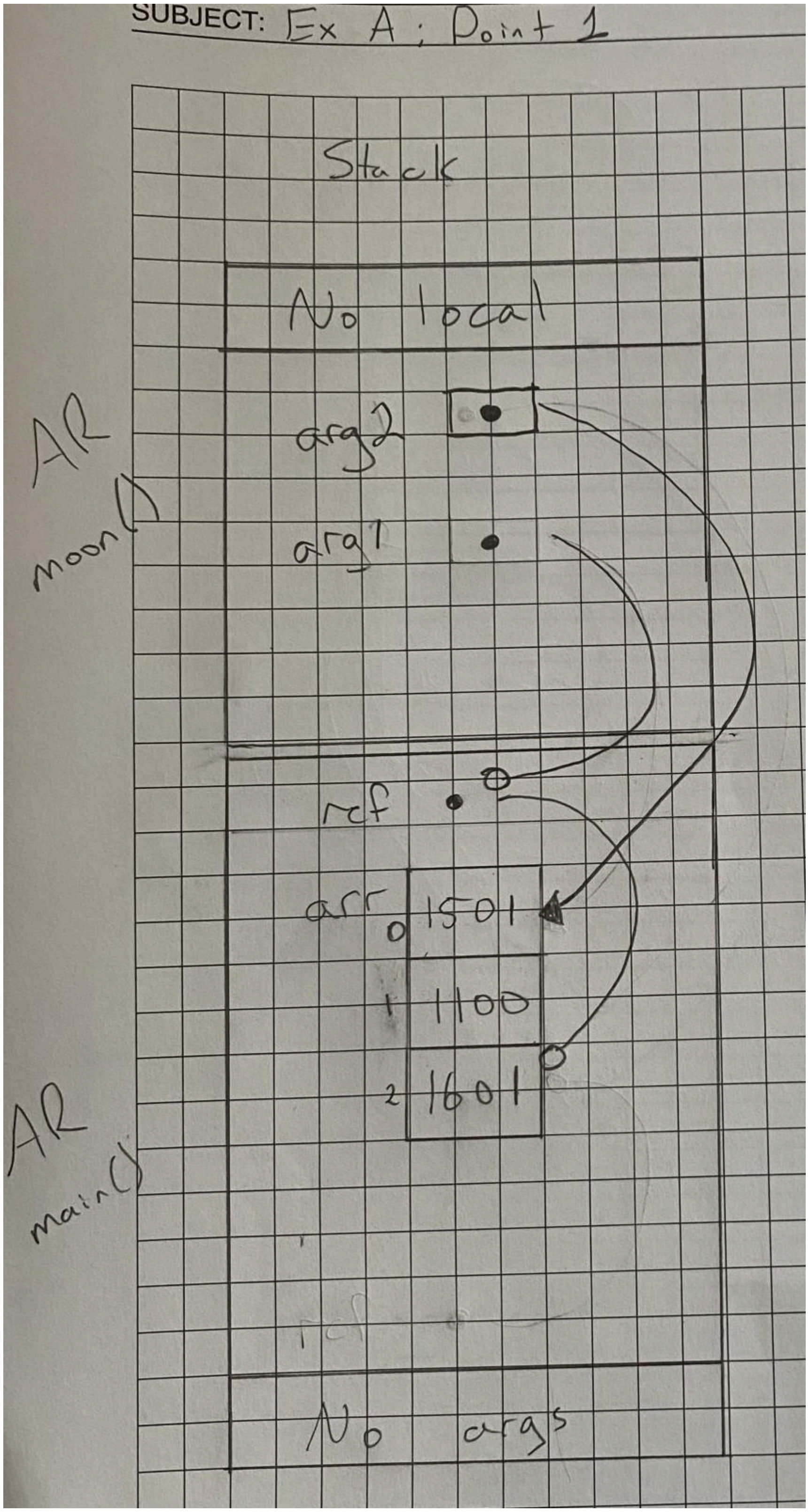
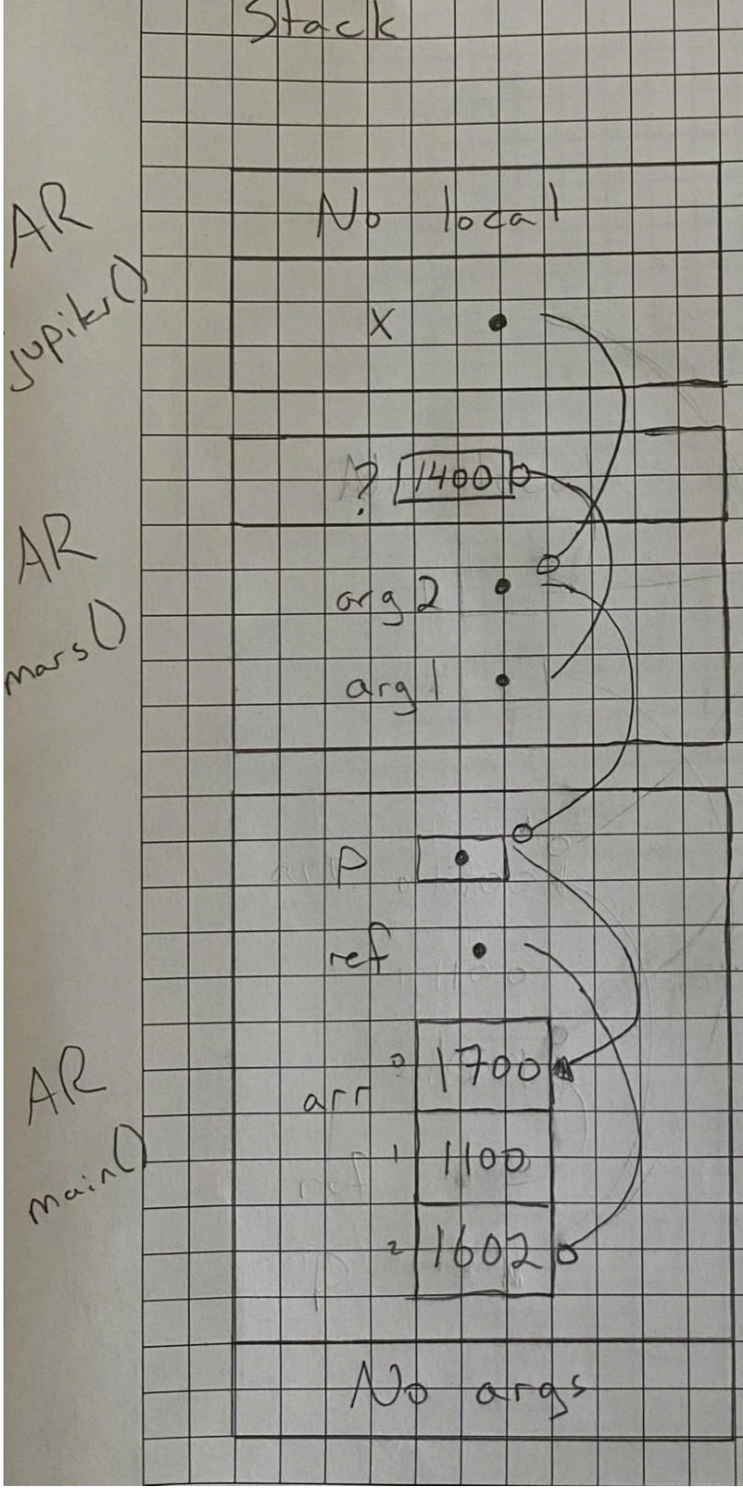
Course: ENSF 694 – Summer 2024
Lab Assignment #: Lab 3
Instructor: Mahmood Moussavi
Student Name: Jeff Wheeler, UCID: 30265340
Submission Date: July 17, 2024

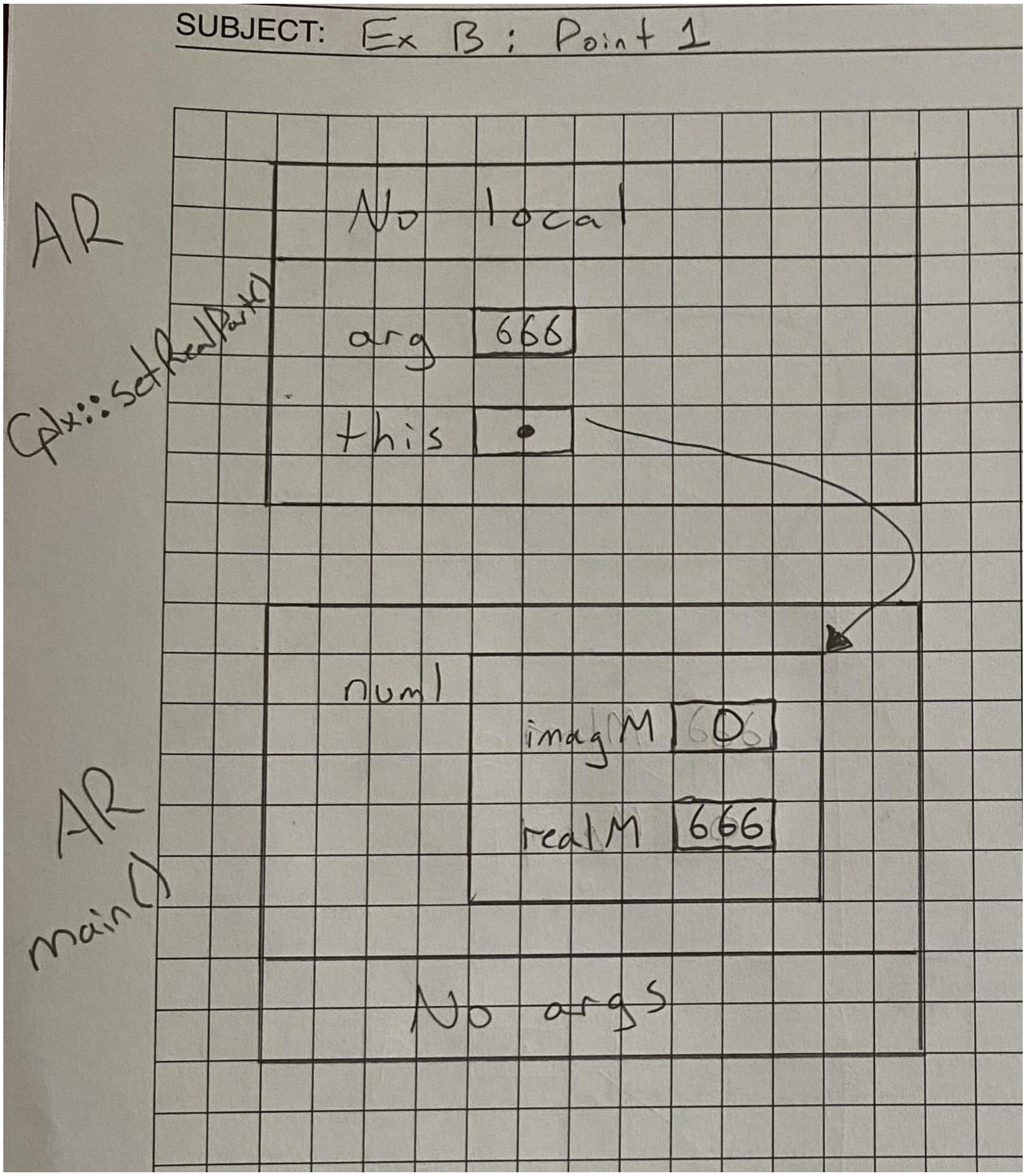
Exercise A


SUBJECT: Ex A ; Point 1
Stack
No local
arg2
AR moon()
arg1
ref
arr
0
1501
1
1100
2
1601
AR main()
No args

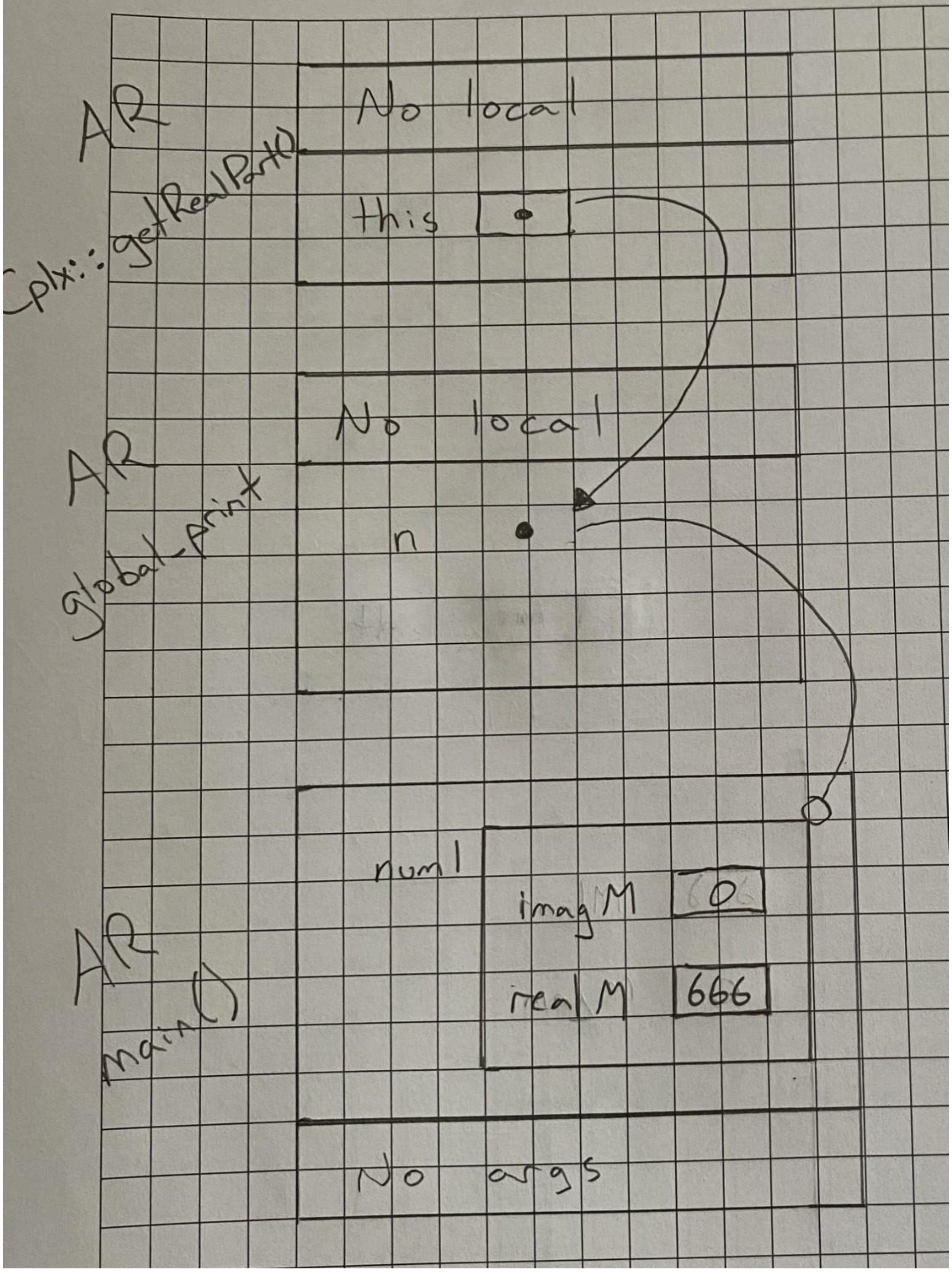
SUBJECT: Ex A: Point 2



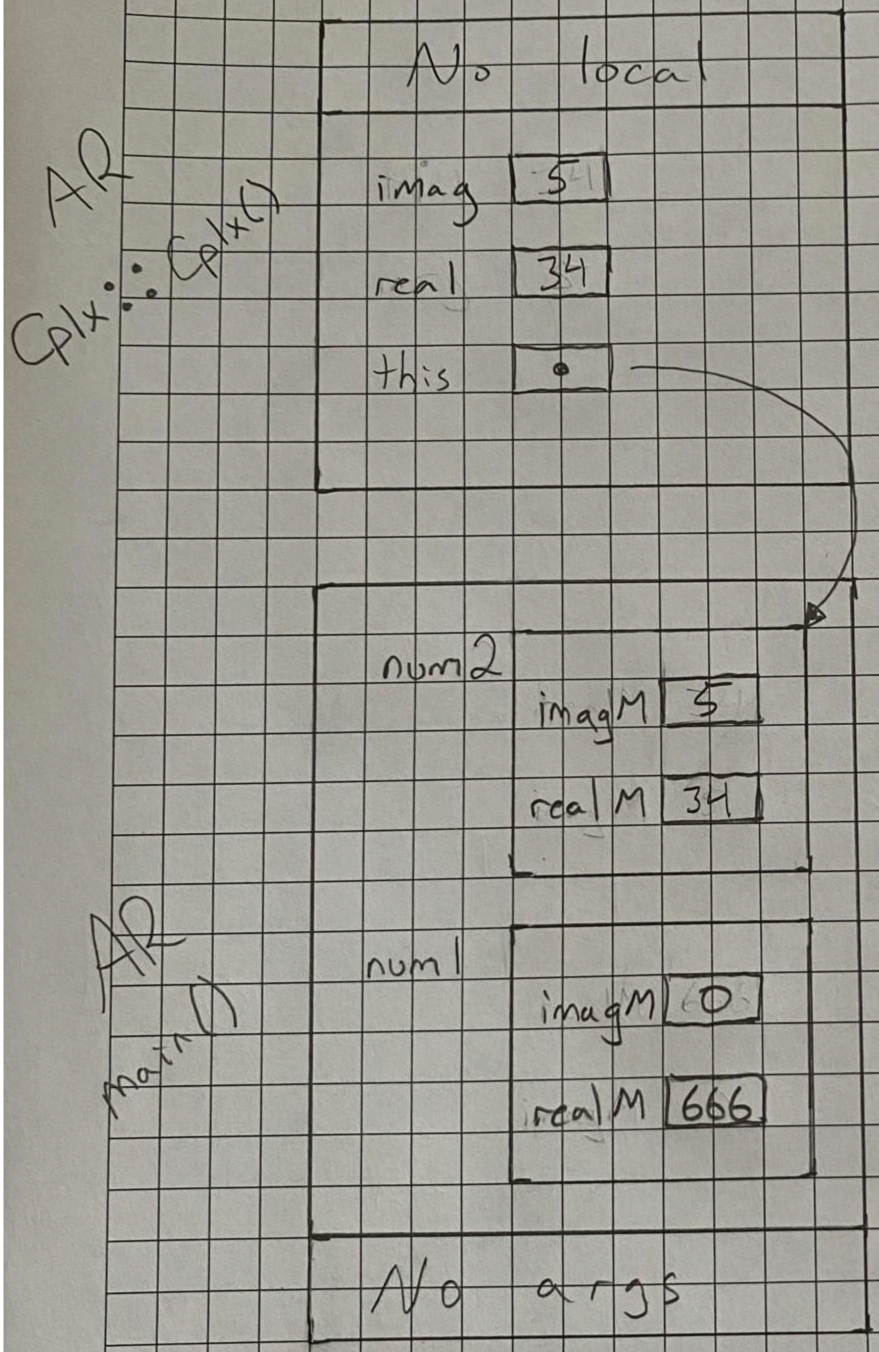
Exercise B


SUBJECT: Ex B : Point 1
AR
Cplx::setRealPart()
No local
arg
666
this
AR
main()
num1
imagM
0
realM
666
No args

SUBJECT: Ex B : Point 2



SUBJECT: Ex B: Point 3

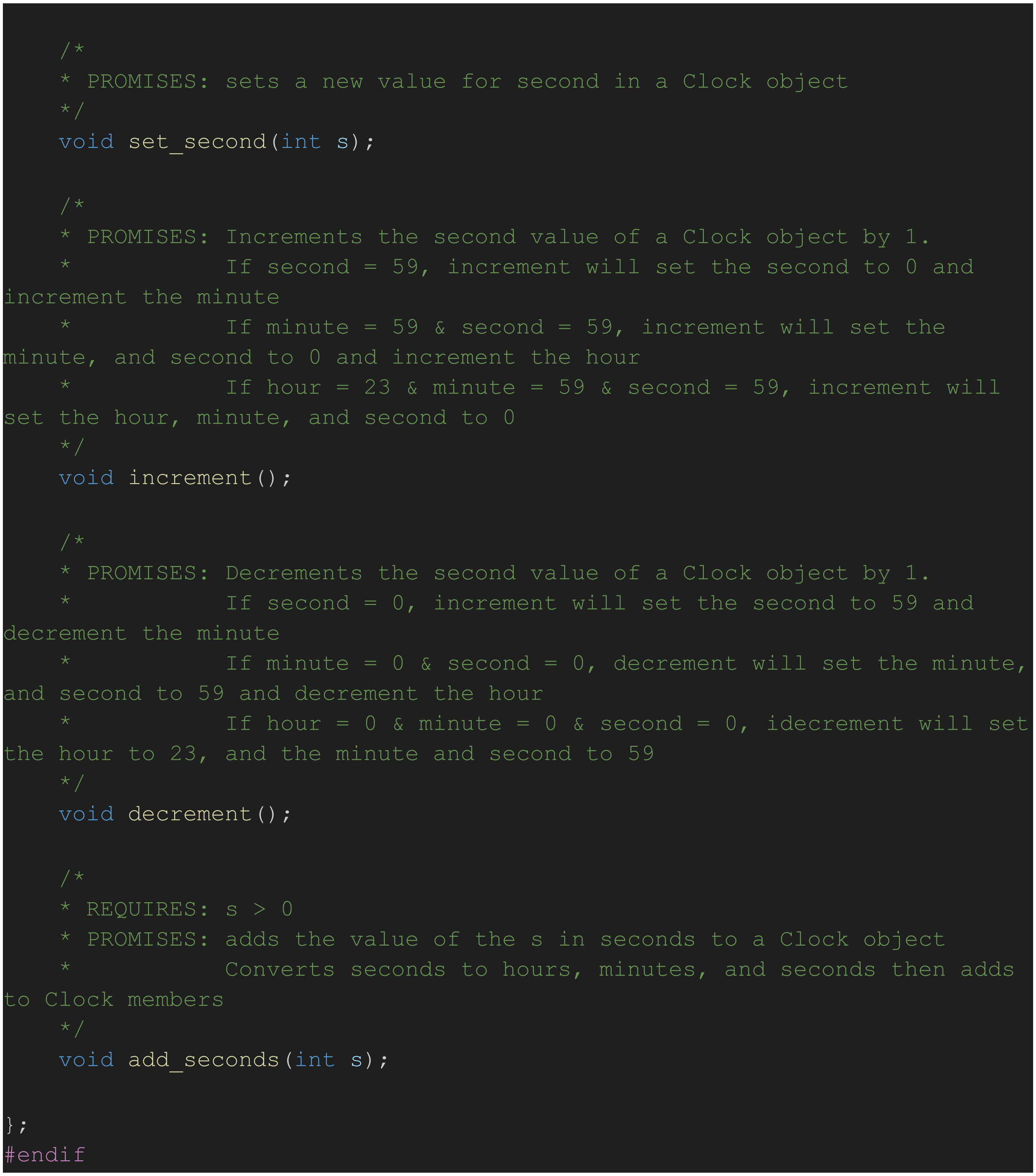


Exercise C

Header code:

```
/*
 *  lab3clock.h
 *  ENSF 694 Lab 3 - Exercise C
 *  Created by Jeff Wheeler
 *  Submission date: July 17, 2024
 */

#ifndef lab3clock_h
#define lab3clock_h

class Clock{
private:
    /*
    * REQUIRES: hour >= 0, hour < 24
    */
    int hour;

    /*
    * REQUIRES: minute >= 0, minute < 60
    */
    int minute;

    /*
    * REQUIRES: second >= 0, second < 60
    */
    int second;

    /*
    * PROMISES: converts a Clock objects hour, minute, and second into
seconds and returns that value
    */
    int hms_to_sec();

    /*
    * REQUIRES: s > 0
    * PROMISES: takes the given s in seconds and assigns an hour, minute
and second value to a Clock object
    */
    void sec_to_hms(int s);
```

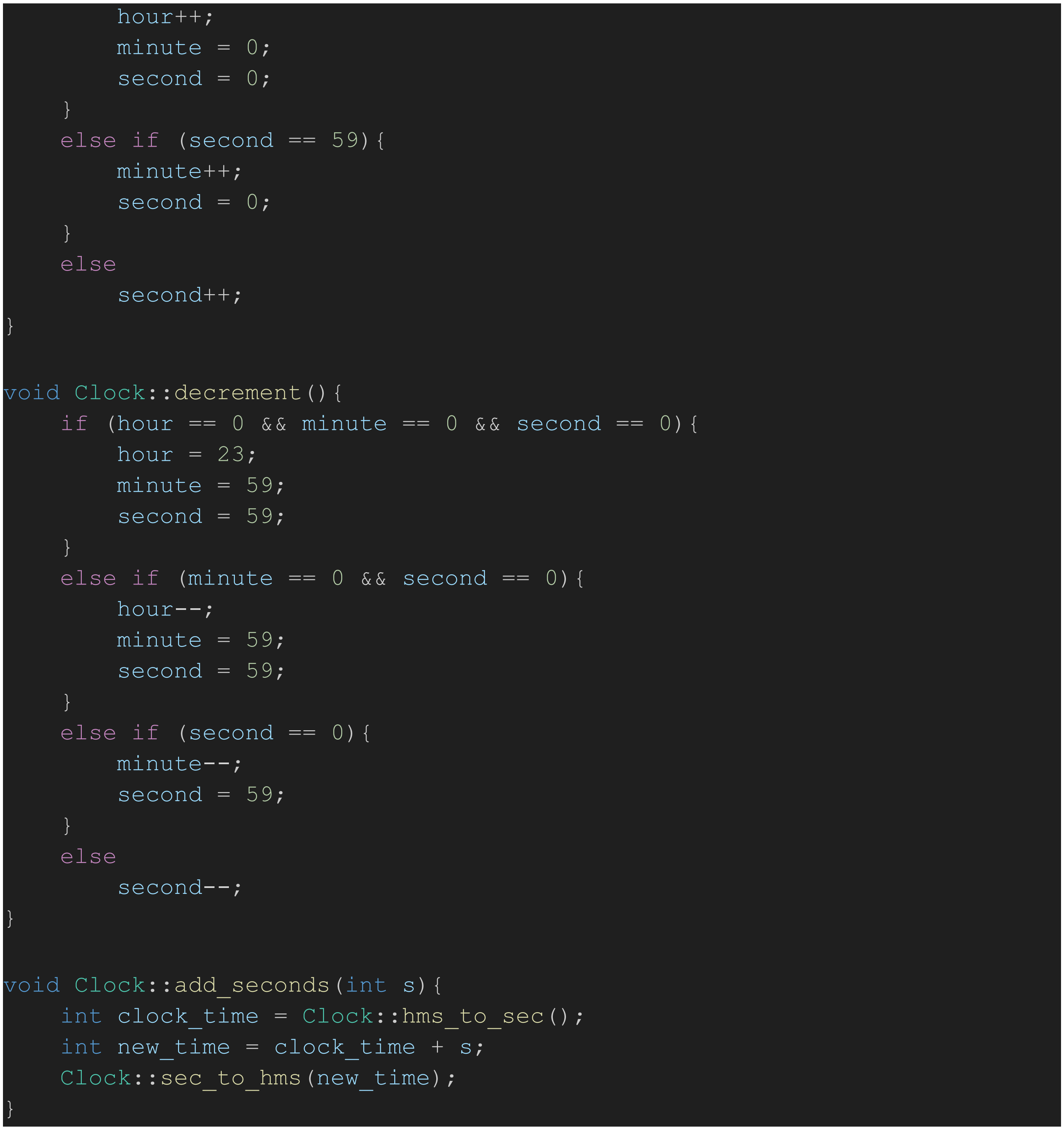
```
public:
    /*
    * PROMISES: Default constructor for clock where hour, minute and
second are all set to 0
    */
    Clock();

    /*
    * PROMISES: Constructor for clock that takes second and sets to hours,
minutes, and seconds
    */
    Clock(int s);

    /*
    * PROMISES: Constructor for clock that sets hours, minutes, and
seconds
    *            If s < 0 or s > 59, set second to 0.
    *            If m < 0 or m > 59, set minute to 0.
    *            If h < 0 or h > 23 set hour to 0.
    */
    Clock(int h, int m, int s);

    /*
    * PROMISES: returns the hour in a Clock object
    */
    int get_hour() const;

    /*
    * PROMISES: returns the minute in a Clock object
    */
    int get_minute() const;

    /*
    * PROMISES: returns the second in a Clock object
    */
    int get_second() const;

    /*
    * PROMISES: sets a new value for hour in a Clock object
    */
    void set_hour(int h);

    /*
    * PROMISES: sets a new value for minute in a Clock object
    */
    void set_minute(int m);
```



```
    /*
    * PROMISES: sets a new value for second in a Clock object
    */
    void set_second(int s);

    /*
    * PROMISES: Increments the second value of a Clock object by 1.
    *           If second = 59, increment will set the second to 0 and
increment the minute
    *           If minute = 59 & second = 59, increment will set the
minute, and second to 0 and increment the hour
    *           If hour = 23 & minute = 59 & second = 59, increment will
set the hour, minute, and second to 0
    */
    void increment();

    /*
    * PROMISES: Decrements the second value of a Clock object by 1.
    *           If second = 0, increment will set the second to 59 and
decrement the minute
    *           If minute = 0 & second = 0, decrement will set the minute,
and second to 59 and decrement the hour
    *           If hour = 0 & minute = 0 & second = 0, idecrement will set
the hour to 23, and the minute and second to 59
    */
    void decrement();

    /*
    * REQUIRES: s > 0
    * PROMISES: adds the value of the s in seconds to a Clock object
    *           Converts seconds to hours, minutes, and seconds then adds
to Clock members
    */
    void add_seconds(int s);

};
#endif
```

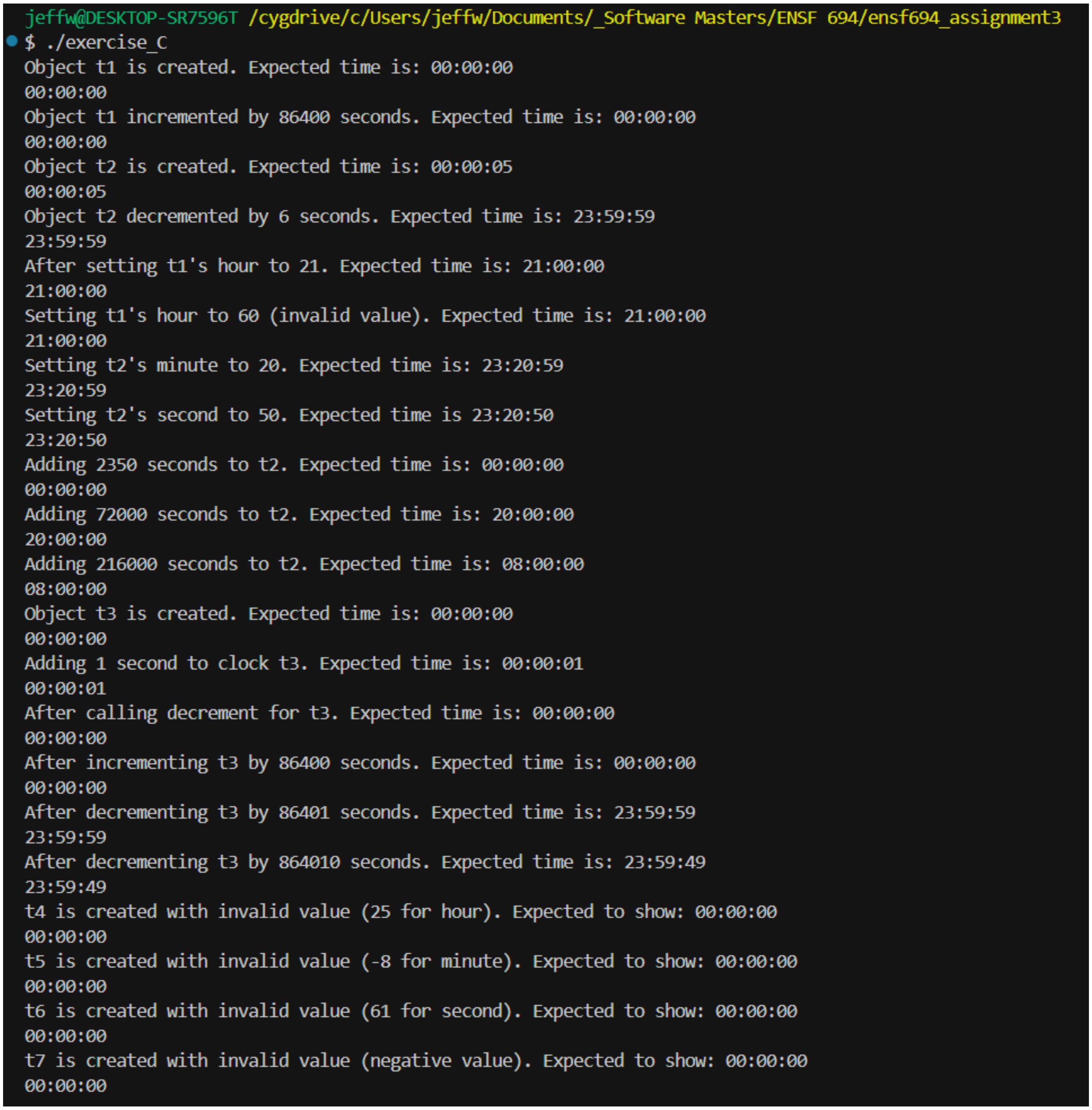
CPP file code:

```
/*
 *  lab3clock.cpp
 *  ENSF 694 Lab 3 - Exercise C
 *  Created by Jeff Wheeler
 *  Submission date: July 17, 2024
 */

#include "lab3clock.h"
using namespace std;

int Clock::hms_to_sec(){
    return (hour * 3600 + minute * 60 + second);
}

void Clock::sec_to_hms(int s){
    if (s >= 86400)
        s %= 86400;
    hour = s / 3600;
    s %= 3600;
    minute = s / 60;
    second = s % 60;
}

Clock::Clock(): hour(0), minute(0), second(0) {}

Clock::Clock(int s) {
    if (s > 0)
        Clock::sec_to_hms(s);
    else{
        Clock::sec_to_hms(0);
    }
}

Clock::Clock(int h, int m, int s) {
    if (h < 0 || h > 23){
        Clock::sec_to_hms(0);
        return;
    }
    else
        hour = h;

    if (m < 0 || m > 59){
        Clock::sec_to_hms(0);
        return;
    }
    else
        minute = m;

    if (s < 0 || s > 59){
        Clock::sec_to_hms(0);
        return;
    }
    else
        second = s;
}

int Clock::get_hour() const{
    return hour;
}

int Clock::get_minute() const{
    return minute;
}

int Clock::get_second() const{
    return second;
}

void Clock::set_hour(int h){
    if (h >= 0 && h <= 23)
        hour = h;
}

void Clock::set_minute(int m){
    if (m >= 0 && m <= 59)
        minute = m;
}

void Clock::set_second(int s){
    if (s >= 0 && s <= 59)
        second = s;
}

void Clock::increment(){
    if (hour == 23 && minute == 59 && second == 59){
        hour = 0;
        minute = 0;
        second = 0;
    }
    else if (minute == 59 && second == 59){
```



```cpp
        hour++;
        minute = 0;
        second = 0;
    }
    else if (second == 59){
        minute++;
        second = 0;
    }
    else
        second++;
}

void Clock::decrement(){
    if (hour == 0 && minute == 0 && second == 0){
        hour = 23;
        minute = 59;
        second = 59;
    }
    else if (minute == 0 && second == 0){
        hour--;
        minute = 59;
        second = 59;
    }
    else if (second == 0){
        minute--;
        second = 59;
    }
    else
        second--;
}

void Clock::add_seconds(int s){
    int clock_time = Clock::hms_to_sec();
    int new_time = clock_time + s;
    Clock::sec_to_hms(new_time);
}
```

Output:

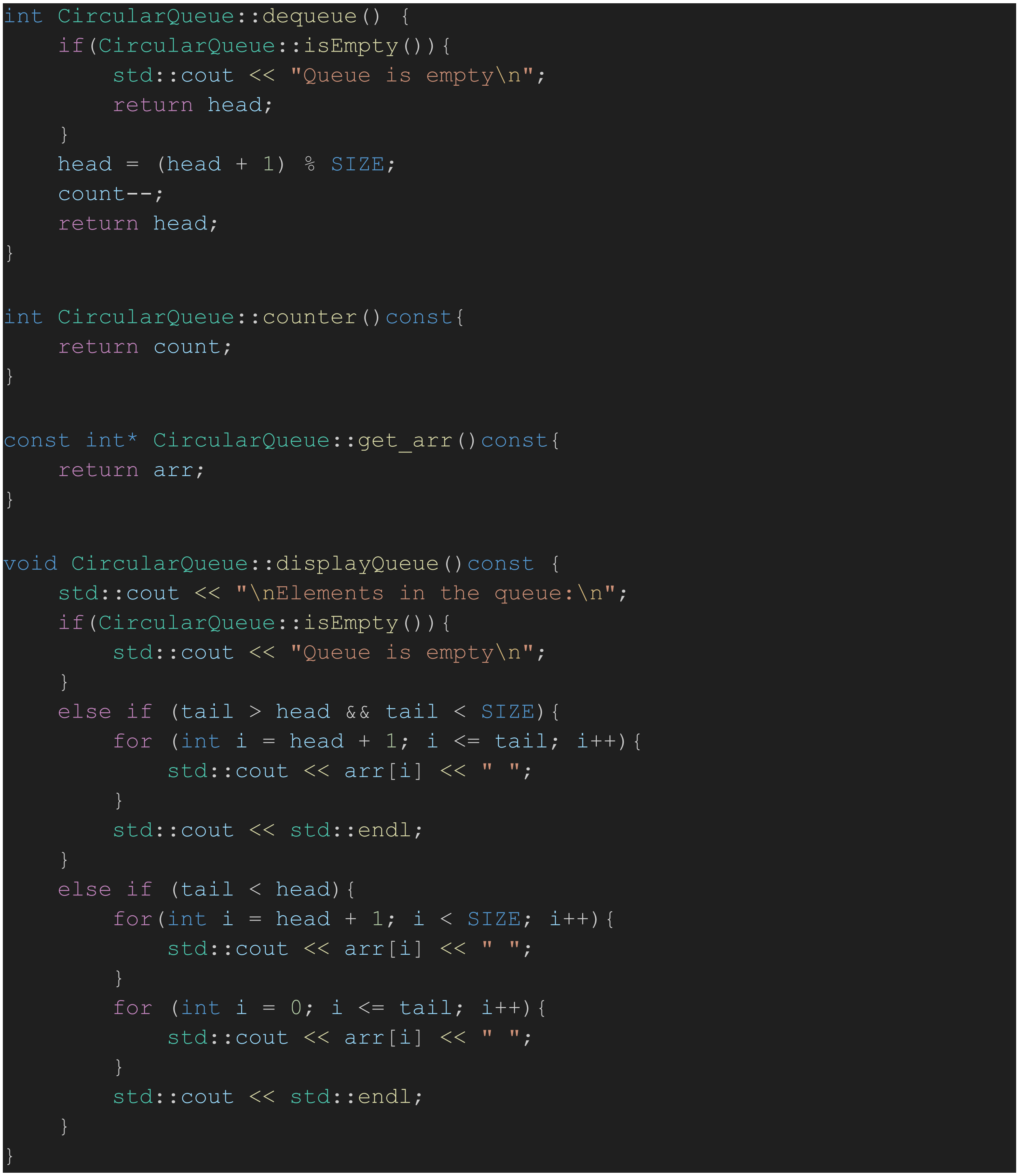


```
jeffw@DESKTOP-SR7596T /cygdrive/c/Users/jeffw/Documents/_Software Masters/ENSF 694/ensf694_assignment3
$ ./exercise_C
Object t1 is created. Expected time is: 00:00:00
00:00:00
Object t1 incremented by 86400 seconds. Expected time is: 00:00:00
00:00:00
Object t2 is created. Expected time is: 00:00:05
00:00:05
Object t2 decremented by 6 seconds. Expected time is: 23:59:59
23:59:59
After setting t1's hour to 21. Expected time is: 21:00:00
21:00:00
Setting t1's hour to 60 (invalid value). Expected time is: 21:00:00
21:00:00
Setting t2's minute to 20. Expected time is: 23:20:59
23:20:59
Setting t2's second to 50. Expected time is 23:20:50
23:20:50
Adding 2350 seconds to t2. Expected time is: 00:00:00
00:00:00
Adding 72000 seconds to t2. Expected time is: 20:00:00
20:00:00
Adding 216000 seconds to t2. Expected time is: 08:00:00
08:00:00
Object t3 is created. Expected time is: 00:00:00
00:00:00
Adding 1 second to clock t3. Expected time is: 00:00:01
00:00:01
After calling decrement for t3. Expected time is: 00:00:00
00:00:00
After incrementing t3 by 86400 seconds. Expected time is: 00:00:00
00:00:00
After decrementing t3 by 86401 seconds. Expected time is: 23:59:59
23:59:59
After decrementing t3 by 864010 seconds. Expected time is: 23:59:49
23:59:49
t4 is created with invalid value (25 for hour). Expected to show: 00:00:00
00:00:00
t5 is created with invalid value (-8 for minute). Expected to show: 00:00:00
00:00:00
t6 is created with invalid value (61 for second). Expected to show: 00:00:00
00:00:00
t7 is created with invalid value (negative value). Expected to show: 00:00:00
00:00:00
```

Exercise D

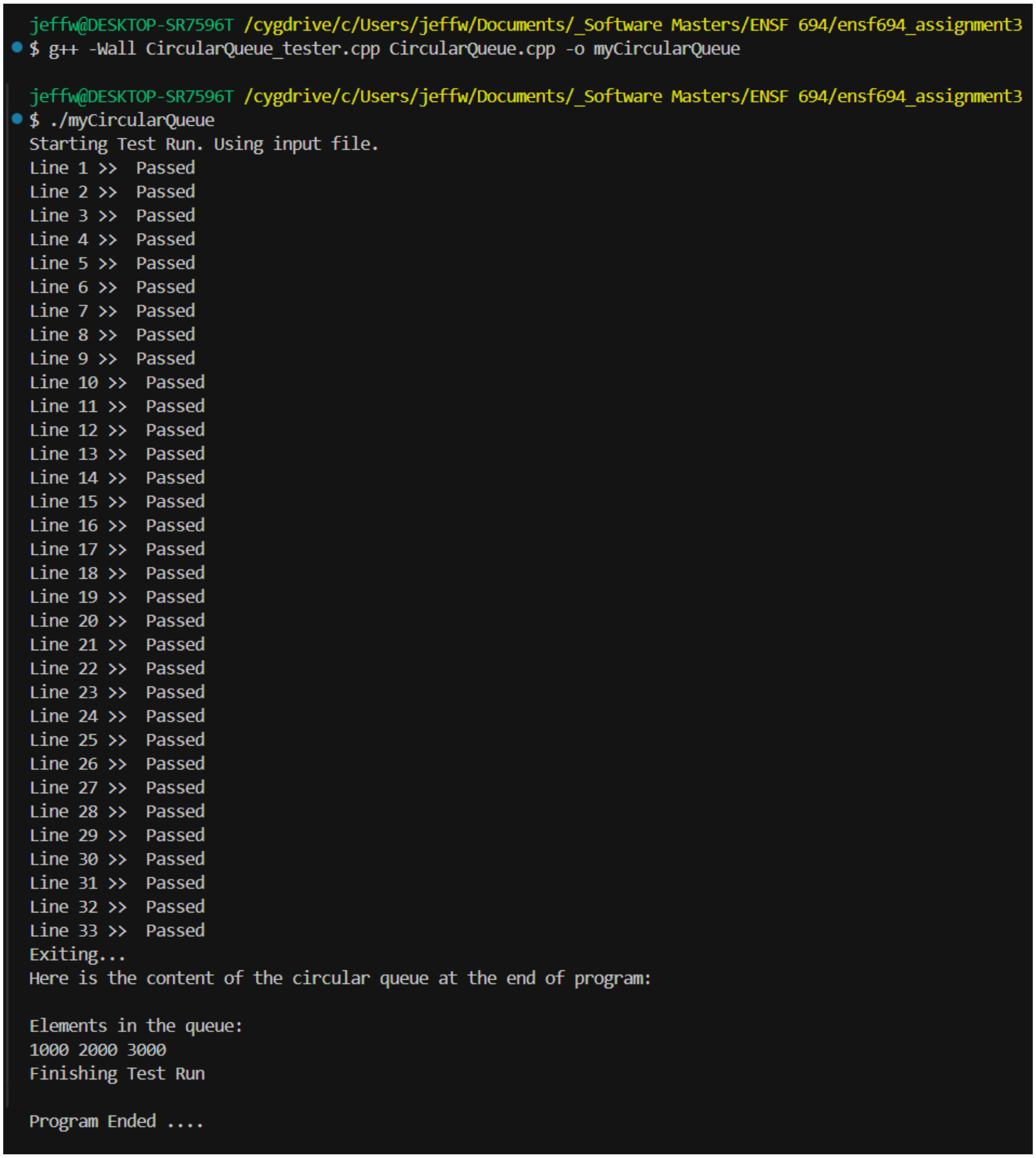
Code:

```
/*
 *  CircularQueue.cpp
 *  ENSF 694 Lab 3 - Exercise D
 *  Created by Mahmood Moussavi
 *  Completed by: Jeff Wheeler
 *  Submission date: July 17, 2024
 */

#include "CircularQueue.h"


CircularQueue::CircularQueue() {
    head = 1;
    tail = 1;
    count = 0;
}

bool CircularQueue::isFull()const {
    if (head == tail + 1)
        return true;
    return false;
}

bool CircularQueue::isEmpty()const {
    if (head == tail)
        return true;
    return false;
}

int CircularQueue::enqueue(int element) {
    if(CircularQueue::isFull()){
        std::cout << "Queue is full\n";
        return tail;
    }
    tail = (tail + 1) % SIZE;
    arr[tail] = element;
    count++;
    return tail;
}
```



```
int CircularQueue::dequeue() {
    if(CircularQueue::isEmpty()){
        std::cout << "Queue is empty\n";
        return head;
    }
    head = (head + 1) % SIZE;
    count--;
    return head;
}

int CircularQueue::counter()const{
    return count;
}

const int* CircularQueue::get_arr()const{
    return arr;
}

void CircularQueue::displayQueue()const {
    std::cout << "\nElements in the queue:\n";
    if(CircularQueue::isEmpty()){
        std::cout << "Queue is empty\n";
    }
    else if (tail > head && tail < SIZE){
        for (int i = head + 1; i <= tail; i++){
            std::cout << arr[i] << " ";
        }
        std::cout << std::endl;
    }
    else if (tail < head){
        for(int i = head + 1; i < SIZE; i++){
            std::cout << arr[i] << " ";
        }
        for (int i = 0; i <= tail; i++){
            std::cout << arr[i] << " ";
        }
        std::cout << std::endl;
    }
}
```

Output:

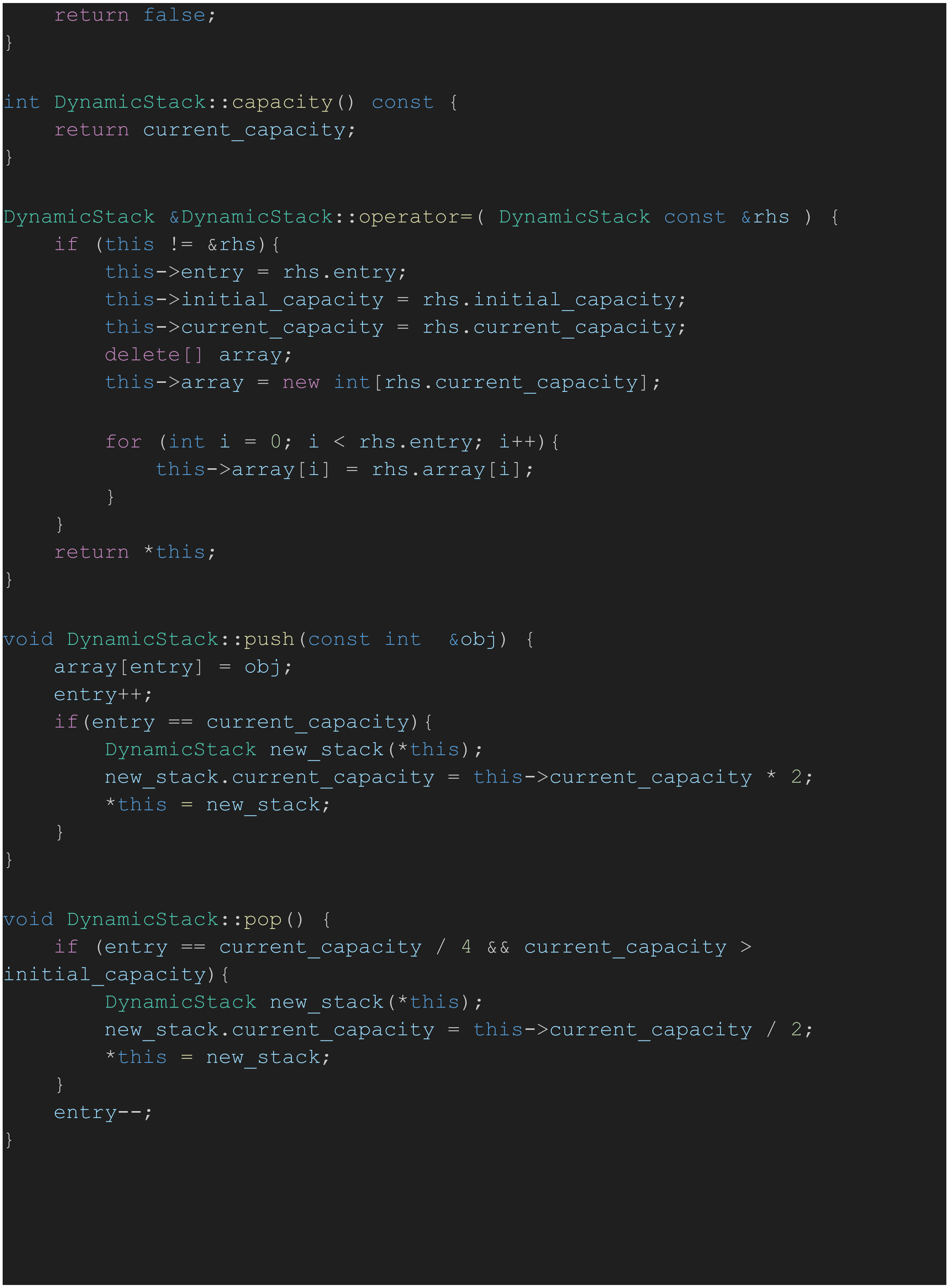


```
jeffw@DESKTOP-SR7596T /cygdrive/c/Users/jeffw/Documents/_Software Masters/ENSF 694/ensf694_assignment3
$ g++ -Wall CircularQueue_tester.cpp CircularQueue.cpp -o myCircularQueue

jeffw@DESKTOP-SR7596T /cygdrive/c/Users/jeffw/Documents/_Software Masters/ENSF 694/ensf694_assignment3
$ ./myCircularQueue
Starting Test Run. Using input file.
Line 1 >>  Passed
Line 2 >>  Passed
Line 3 >>  Passed
Line 4 >>  Passed
Line 5 >>  Passed
Line 6 >>  Passed
Line 7 >>  Passed
Line 8 >>  Passed
Line 9 >>  Passed
Line 10 >>  Passed
Line 11 >>  Passed
Line 12 >>  Passed
Line 13 >>  Passed
Line 14 >>  Passed
Line 15 >>  Passed
Line 16 >>  Passed
Line 17 >>  Passed
Line 18 >>  Passed
Line 19 >>  Passed
Line 20 >>  Passed
Line 21 >>  Passed
Line 22 >>  Passed
Line 23 >>  Passed
Line 24 >>  Passed
Line 25 >>  Passed
Line 26 >>  Passed
Line 27 >>  Passed
Line 28 >>  Passed
Line 29 >>  Passed
Line 30 >>  Passed
Line 31 >>  Passed
Line 32 >>  Passed
Line 33 >>  Passed
Exiting...
Here is the content of the circular queue at the end of program:

Elements in the queue:
1000 2000 3000
Finishing Test Run

Program Ended ....
```

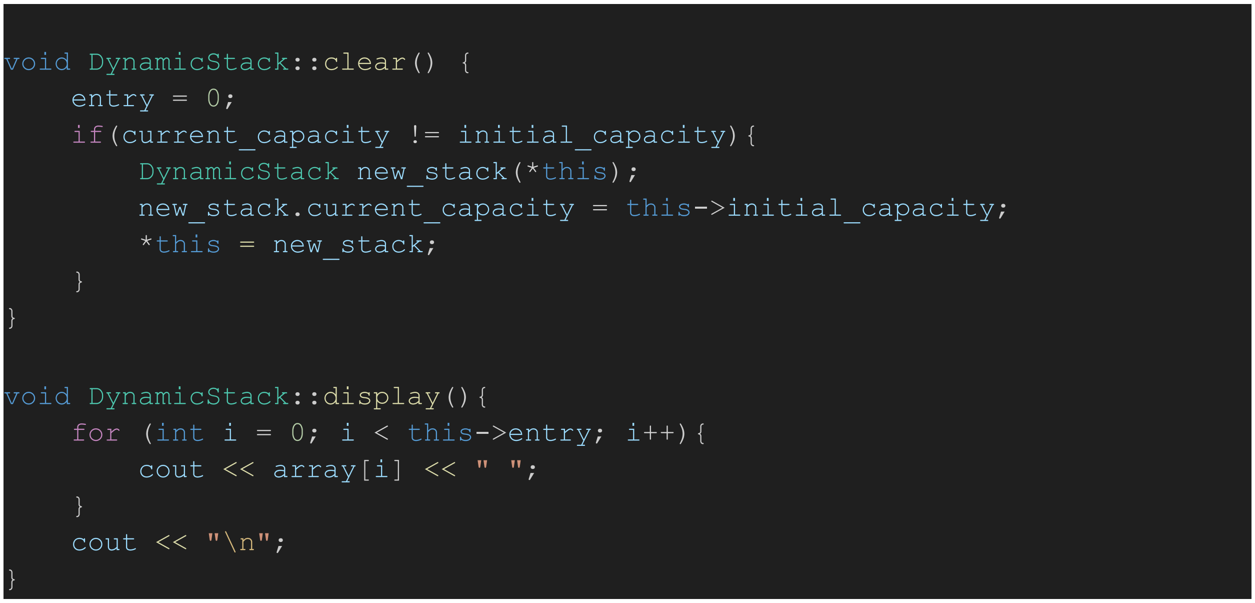
Exercise E

Code:

```
/*
 *  DynamicStack.cpp
 *  ENSF 694 Lab 3 - Exercise E
 *  Created by Mahmood Moussavi
 *  Completed by: Jeff Wheeler
 *  Submission date: July 17, 2024
 */

#include "DynamicStack.h"
DynamicStack::DynamicStack(int n): entry(0), initial_capacity(n),
current_capacity(n) {
    array = new int[n];
}

DynamicStack::DynamicStack(DynamicStack const &src) {
    this->entry = src.entry;
    this->initial_capacity = src.initial_capacity;
    this->current_capacity = src.current_capacity;
    array = new int[src.current_capacity];

    for (int i =0; i < src.current_capacity; i++){
        this->array[i] = src.array[i];
    }
}

DynamicStack::~DynamicStack() {
    delete[] array;
}

int DynamicStack::top() const {
    if (DynamicStack::empty())
        return -1; // if the stack is empty return -1
    return array[entry - 1];
}

int DynamicStack::size() const {
    return entry;
}

bool DynamicStack::empty() const {
    if (entry == 0)
        return true;
```

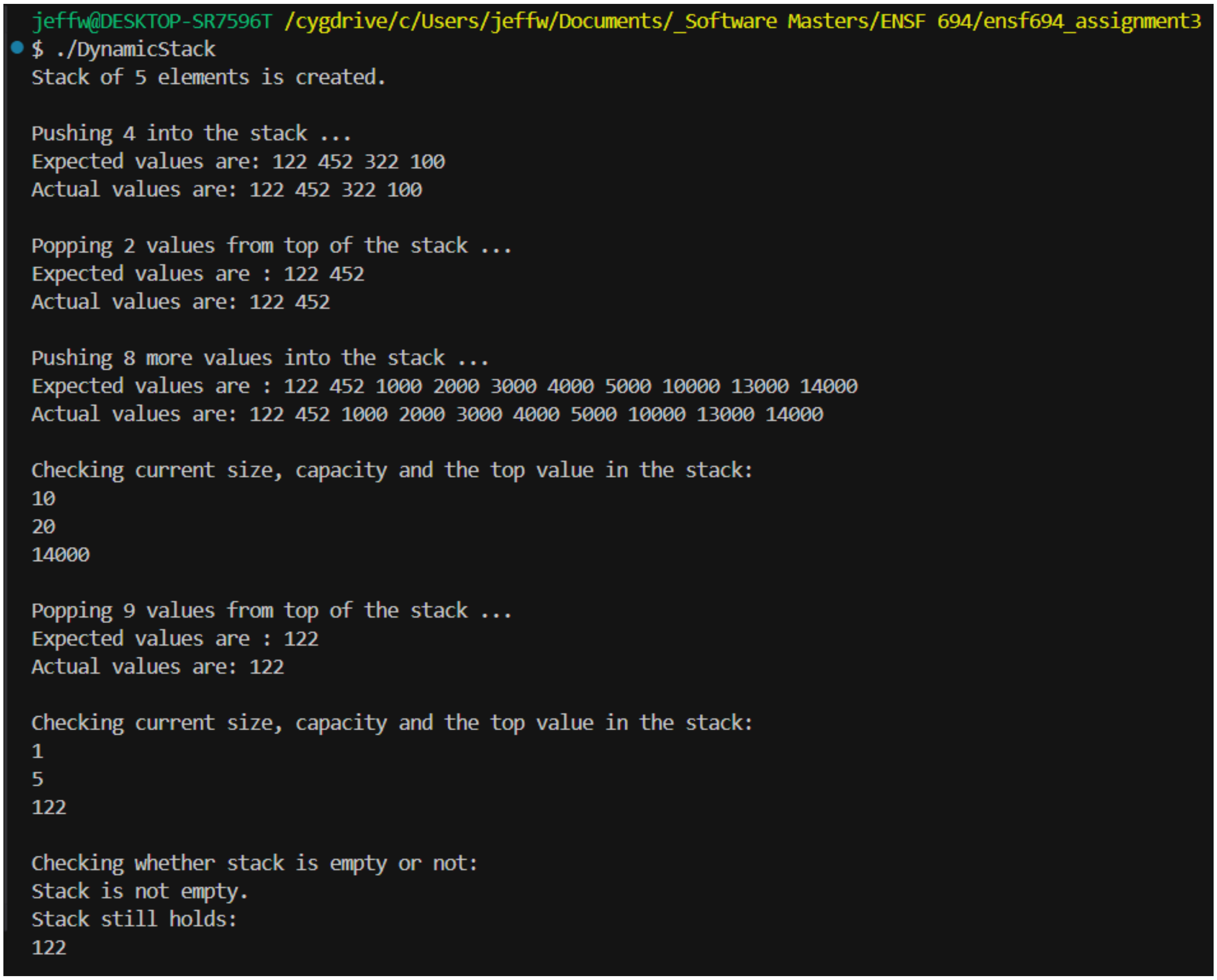


```
    return false;
}

int DynamicStack::capacity() const {
    return current_capacity;
}

DynamicStack &DynamicStack::operator=( DynamicStack const &rhs ) {
    if (this != &rhs){
        this->entry = rhs.entry;
        this->initial_capacity = rhs.initial_capacity;
        this->current_capacity = rhs.current_capacity;
        delete[] array;
        this->array = new int[rhs.current_capacity];

        for (int i = 0; i < rhs.entry; i++){
            this->array[i] = rhs.array[i];
        }
    }
    return *this;
}

void DynamicStack::push(const int  &obj) {
    array[entry] = obj;
    entry++;
    if(entry == current_capacity){
        DynamicStack new_stack(*this);
        new_stack.current_capacity = this->current_capacity * 2;
        *this = new_stack;
    }
}

void DynamicStack::pop() {
    if (entry == current_capacity / 4 && current_capacity >
initial_capacity){
        DynamicStack new_stack(*this);
        new_stack.current_capacity = this->current_capacity / 2;
        *this = new_stack;
    }
    entry--;
}
```



```
void DynamicStack::clear() {
    entry = 0;
    if(current_capacity != initial_capacity){
        DynamicStack new_stack(*this);
        new_stack.current_capacity = this->initial_capacity;
        *this = new_stack;
    }
}

void DynamicStack::display(){
    for (int i = 0; i < this->entry; i++){
        cout << array[i] << " ";
    }
    cout << "\n";
}
```

Output:

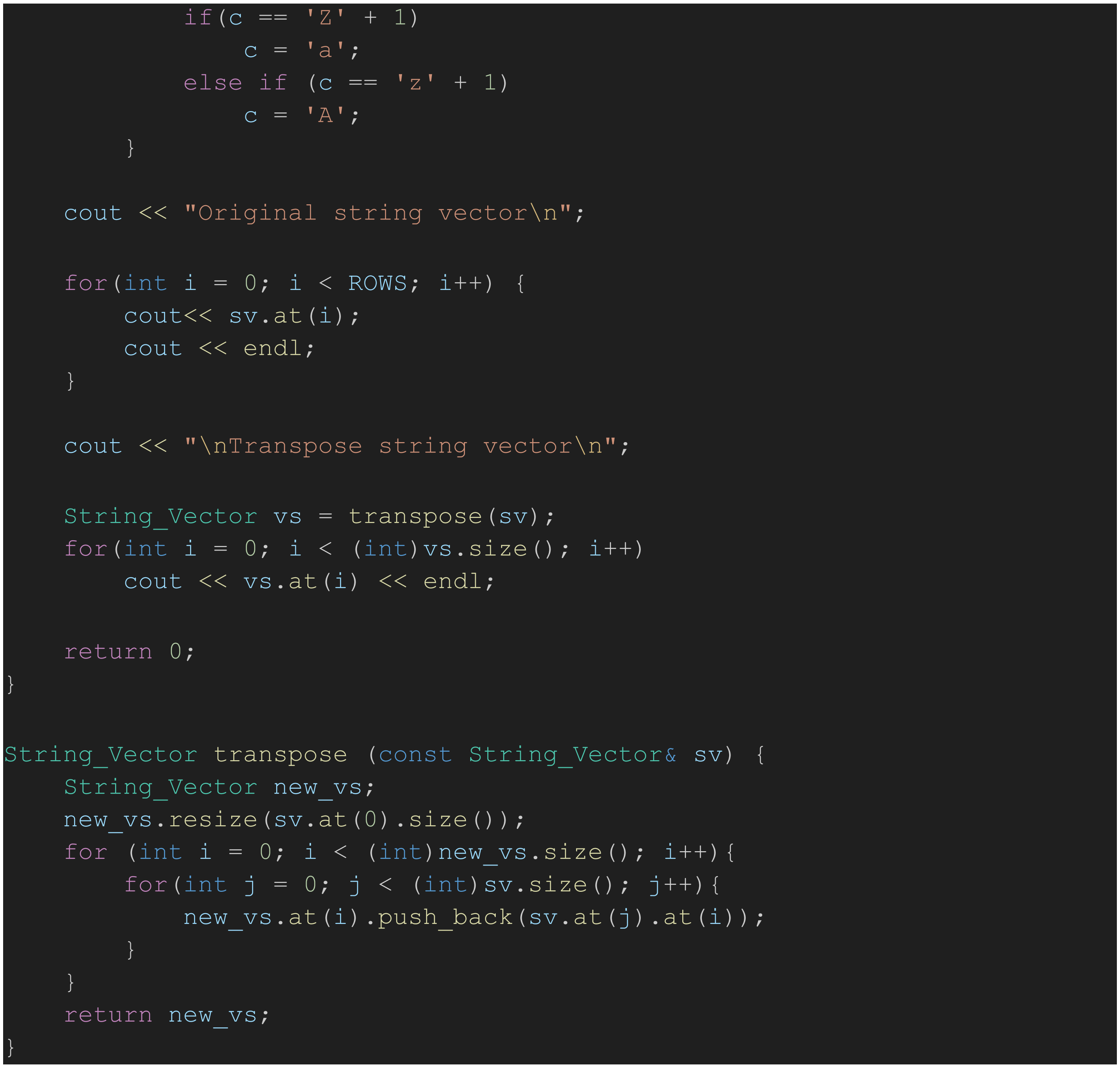


```
jeffw@DESKTOP-SR7596T /cygdrive/c/Users/jeffw/Documents/_Software Masters/ENSF 694/ensf694_assignment3
$ ./DynamicStack
Stack of 5 elements is created.

Pushing 4 into the stack ...
Expected values are: 122 452 322 100
Actual values are: 122 452 322 100

Popping 2 values from top of the stack ...
Expected values are : 122 452
Actual values are: 122 452

Pushing 8 more values into the stack ...
Expected values are : 122 452 1000 2000 3000 4000 5000 10000 13000 14000
Actual values are: 122 452 1000 2000 3000 4000 5000 10000 13000 14000

Checking current size, capacity and the top value in the stack:
10
20
14000

Popping 9 values from top of the stack ...
Expected values are : 122
Actual values are: 122

Checking current size, capacity and the top value in the stack:
1
5
122

Checking whether stack is empty or not:
Stack is not empty.
Stack still holds:
122
```

Exercise F

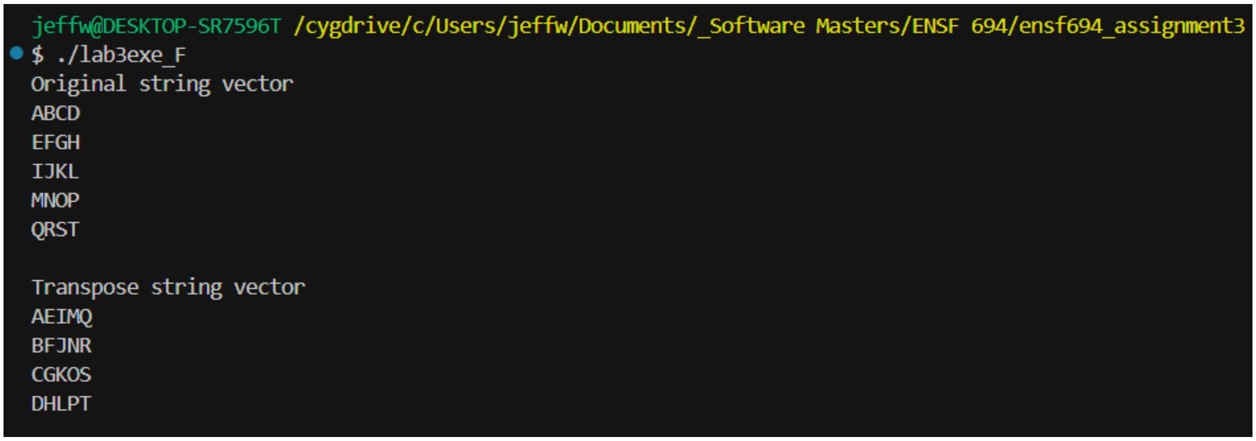
Code:

```
/*
 *  lab3exe_F.cpp
 *  ENSF 694 Lab 3 - Exercise F
 *  Created by Mahmood Moussavi
 *  Completed by Jeff Wheeler
 *  Submission date: July 17, 2024
 */

#include<vector>
#include<string>
#include <iostream>
using std::cout;
using std::cerr;
using std::endl;
using std::vector;
using std::string;

typedef vector<string> String_Vector;

// REQUIRES:
//    sv.size() >= 1
//    All the strings in sv are the same length, and that length is >= 1.
// PROMISES:
//    Return value is the "transpose" of sv, as defined in the Exercise B
//    instructions.
String_Vector transpose(const String_Vector& sv);

int main() {

    const int ROWS = 5;
    const int COLS = 4;

    char c = 'A';
    String_Vector sv;
    sv.resize(ROWS);




    for(int i = 0; i < ROWS; i++)
        for(int j = 0; j < COLS; j++) {
            sv.at(i).push_back(c);
            c++;
```



```
            if(c == 'Z' + 1)
                c = 'a';
            else if (c == 'z' + 1)
                c = 'A';
        }

    cout << "Original string vector\n";

    for(int i = 0; i < ROWS; i++) {
        cout<< sv.at(i);
        cout << endl;
    }

    cout << "\nTranspose string vector\n";

    String_Vector vs = transpose(sv);
    for(int i = 0; i < (int)vs.size(); i++)
        cout << vs.at(i) << endl;

    return 0;
}

String_Vector transpose (const String_Vector& sv) {
    String_Vector new_vs;
    new_vs.resize(sv.at(0).size());
    for (int i = 0; i < (int)new_vs.size(); i++){
        for(int j = 0; j < (int)sv.size(); j++){
            new_vs.at(i).push_back(sv.at(j).at(i));
        }
    }
    return new_vs;
}
```

Output:



```
jeffw@DESKTOP-SR7596T /cygdrive/c/Users/jeffw/Documents/_Software Masters/ENSF 694/ensf694_assignment3
$ ./lab3exe_F
Original string vector
ABCD
EFGH
IJKL
MNOP
QRST

Transpose string vector
AEIMQ
BFJNR
CGKOS
DHLPT
```