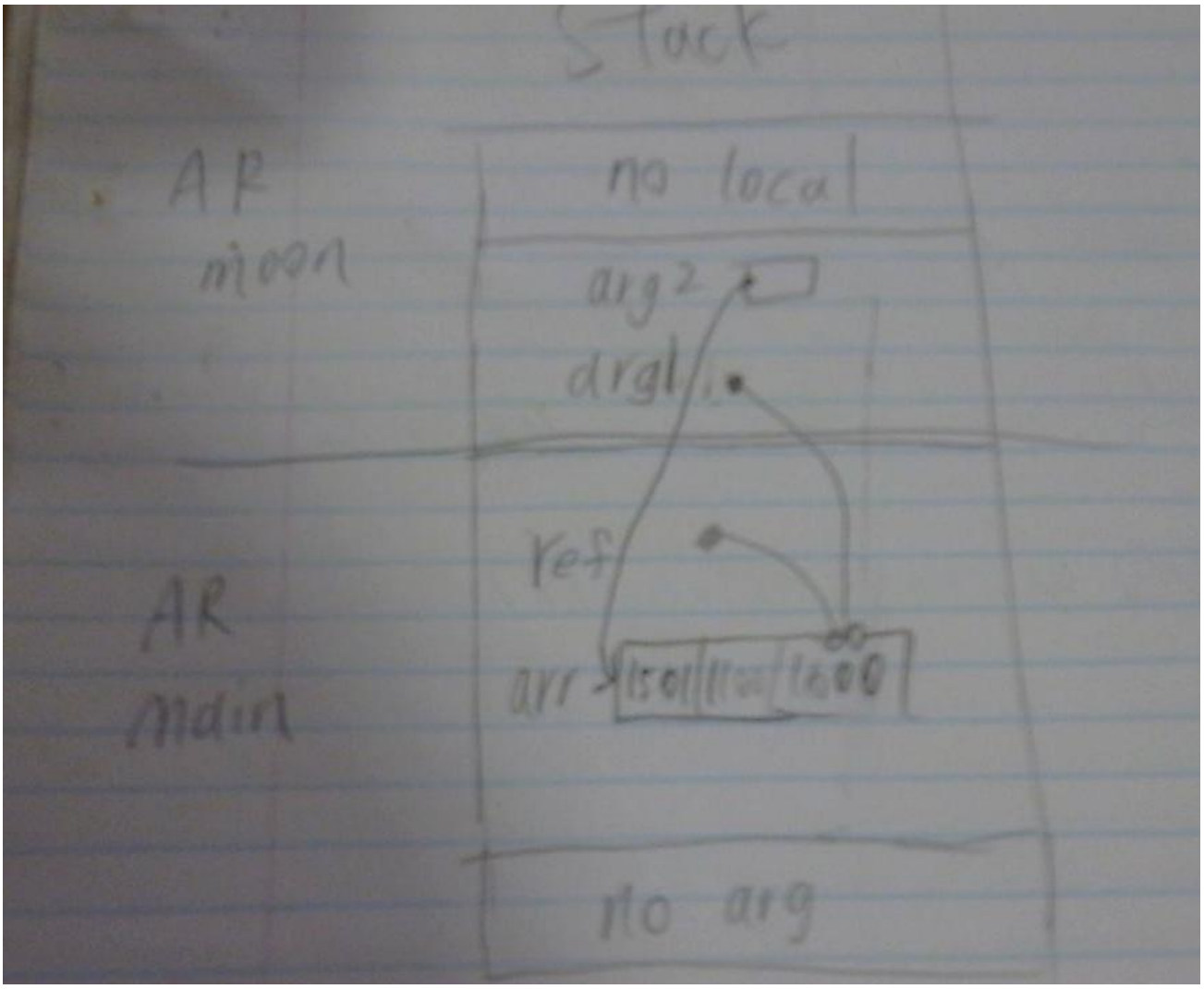
Course: ENSF694 – Summer 2025
Lab #: Lab 3
Instructor: Mahmood Moussavi
Student Name: John Zhou
Submission Date: July 24, 2025

I have been keeping all the files in github. I hope by providing this github link will help you a little bit.
https://github.com/JZ-Zhou-UofC/ENSF-604-assignment-repo

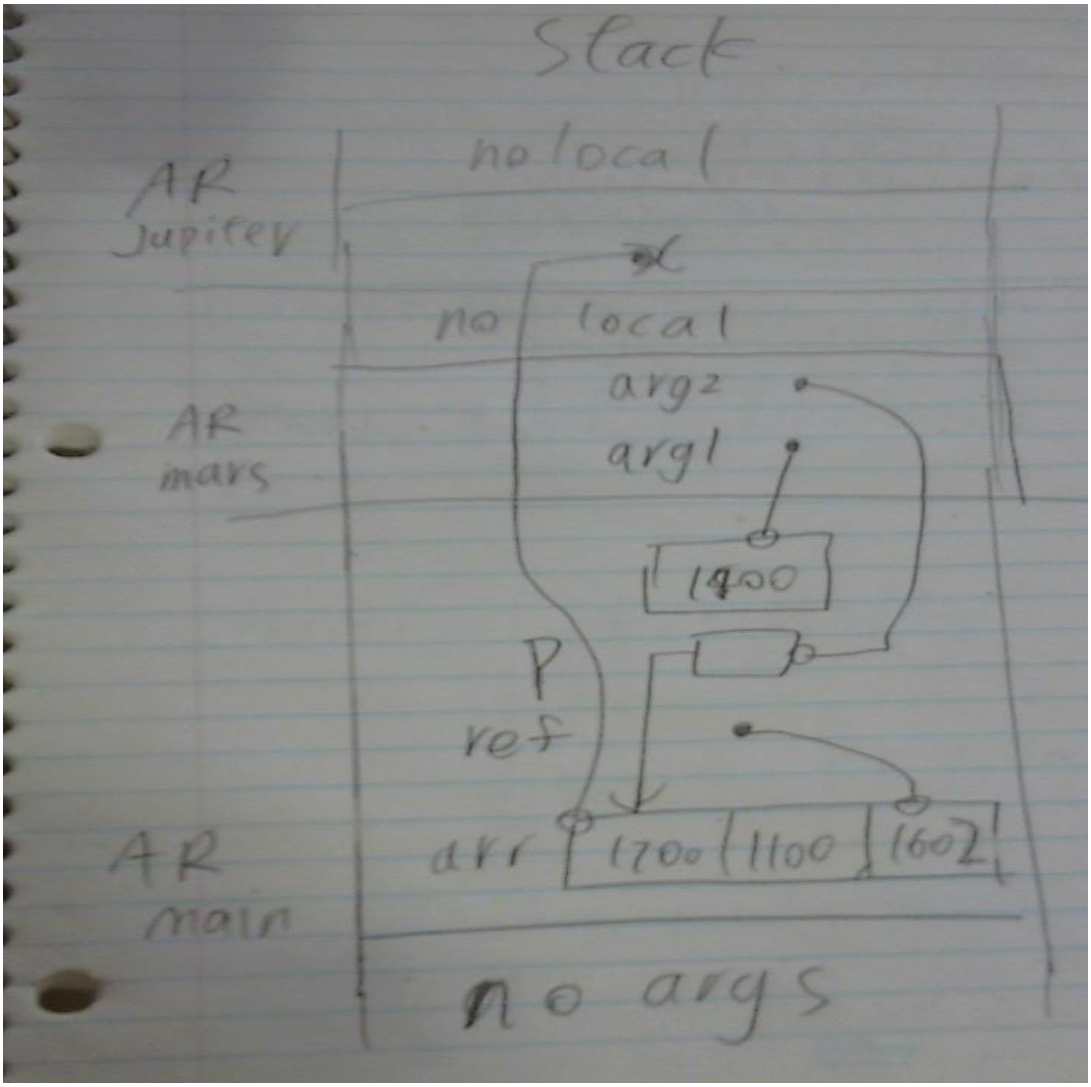
Exercise A

The static area only contains string constant like " " , "\nin mars: " etc.

Point1



Point2


Stack
AR
Jupiter
no local
no local
arg2
arg1
AR
mars
1400
P
ref
AR
main
arr
1700
1100
1602
no args

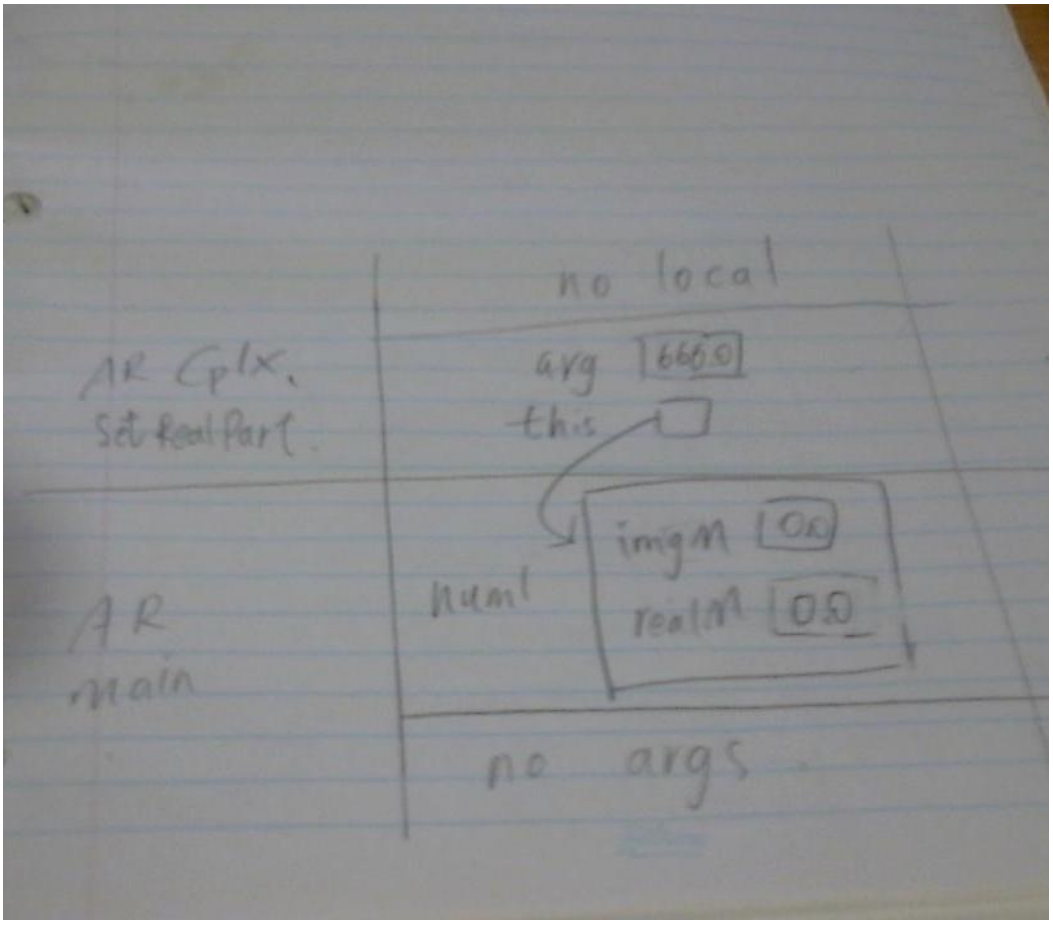
Exercise B

The static area only contains string constant like "\nYour complex number is: ("

, "\nTesting member functions add and subtract: \n" etc.

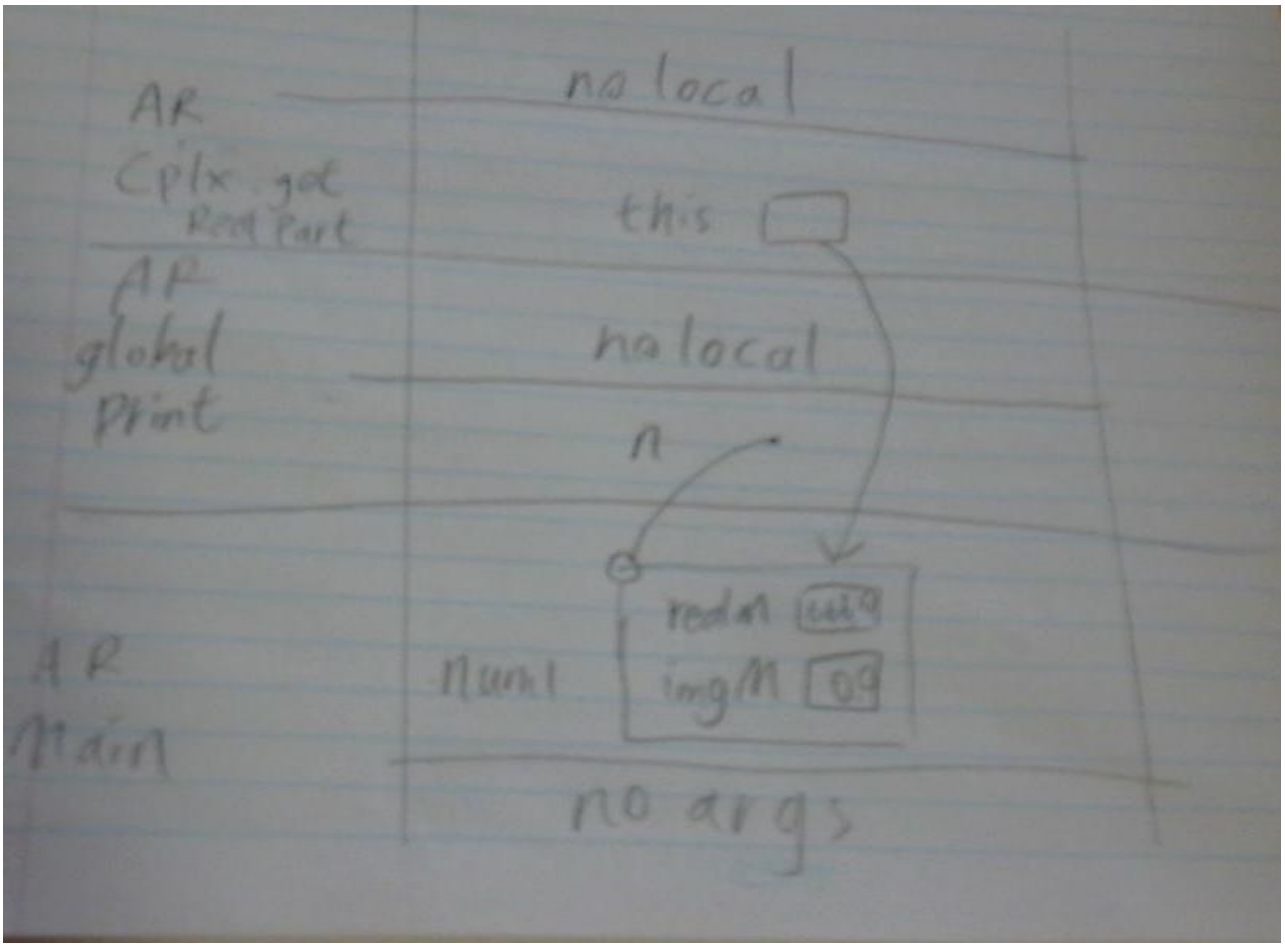
Point1

stack area

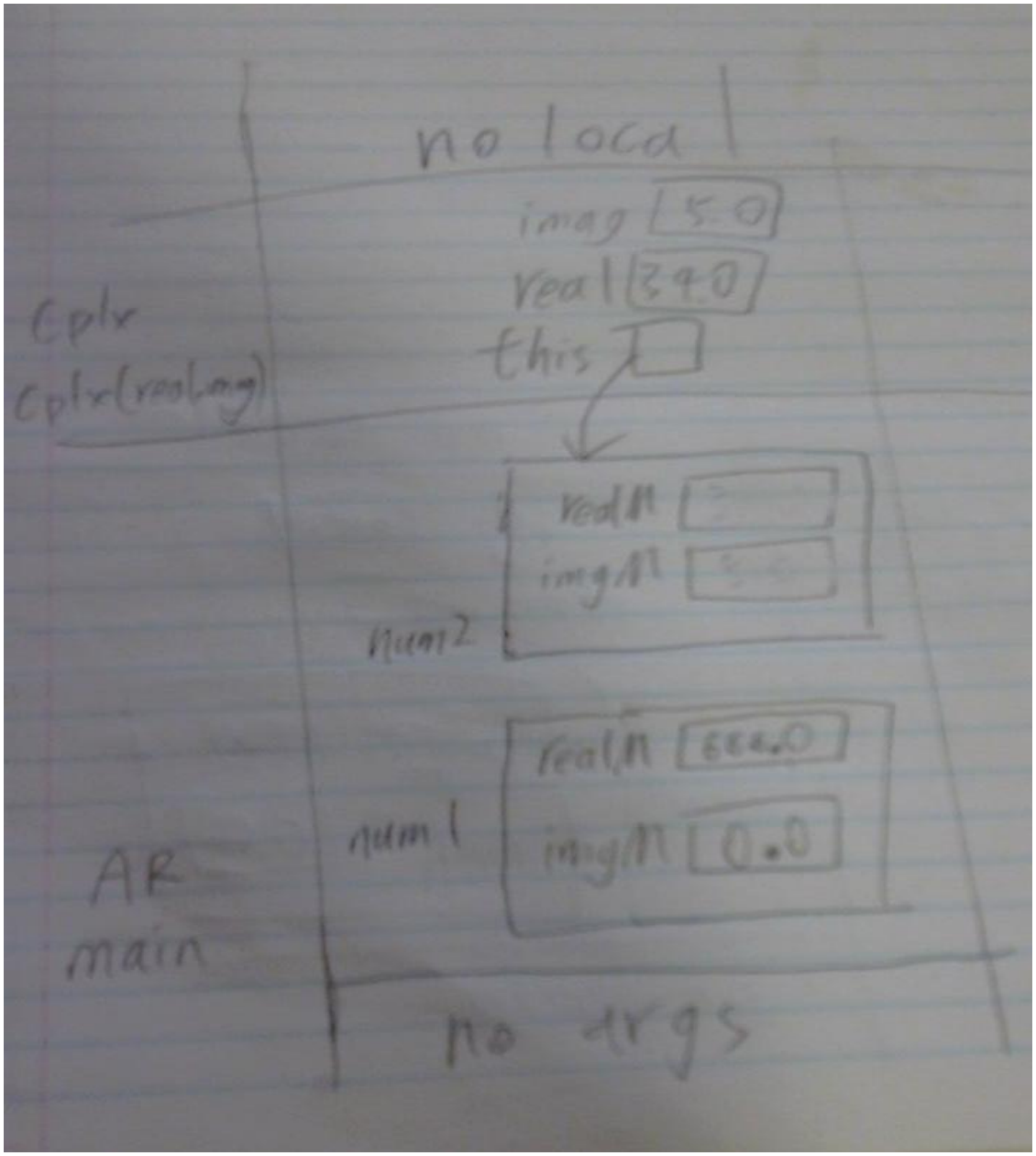


Point2

stack area



Point3
stack area



Exercise C

cpp

```
// lab3Clock.cpp

// ENSF 694 Summer 2025 LAB 3 - EXERCISE C

// Created by: John Zhou


#include "lab3Clock.h"

#include <iostream>

using namespace std;


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

{

}


Clock::Clock(int s)

{

   if (s < 0)

   {

      *this = Clock();

   }

   else

   {

      sec_to_hms(s);

   }

}

Clock::Clock(int h, int m, int s) : hour(h),

                                    minute(m), second(s)

{

   if (s < 0 || m < 0 || h < 0 || m > 59 || s > 59 || h > 23)
```

```
    {
        *this = Clock();
    }
    else
    {
        this->hour = h;
        this->minute = m;
        this->second = s;
    }
}

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

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

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

int Clock::get_time_in_seconds() const
{
    return hms_to_sec();
```

```
}

void Clock::set_hour(int h)
{
   if (h >= 0 && h < 24)
   {
      hour = h;
   }
}

void Clock::set_minute(int m)
{
   if (m >= 0 && m < 60)
   {
      minute = m;
   }
}

void Clock::set_second(int s)
{
   if (s >= 0 && s < 60)
   {
      second = s;
   }
}

void Clock::set_time(int h, int m, int s)
{
   set_hour(h);
```

```
    set_minute(m);
    set_second(s);
}

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

void Clock::sec_to_hms(int s)
{
    int hour = s / 3600; // Calculate hours
    hour = hour % 24;    // Ensure hours are within a 24-hour range

    this->hour = hour;
    this->minute = (s % 3600) / 60;
    this->second = s % 60;
}

void Clock::increment()
{
    second++;
    if (second == 60)
    {
        second = 0;
        minute++;
        if (minute == 60)
        {
            minute = 0;
```

```
            hour++;
            if (hour == 24)
            {
                hour = 0;
            }
        }
    }
}
void Clock::decrement()
{
    if (hms_to_sec() == 0)
    {
        set_time(23, 59, 59);
    }
    else
    {
        sec_to_hms(hms_to_sec() - 1);
    }
}

void Clock::add_seconds(int s)
{
    if (s < 0)
    {
        cout << "Error: Seconds must be a positive integer" << endl;
        return;
    }

    int total_seconds = hms_to_sec() + s;
```

```
    total_seconds = total_seconds % (24 * 60 * 60);
    sec_to_hms(total_seconds);
}
```

H file

```
// lab3Clock.h
// ENSF 694 Summer 2025 LAB 3 - EXERCISE C
// Created by: John Zhou

#ifndef lab3_exe_C_Cplx
#define lab3_exe_C_Cplx
/* The following class definition represents a clock and contains three
 * private data members called hour, minute, and second.
 */

class Clock
{
public:
    // Default constructor
    Clock();
    /* PROMISES: Initializes the clock with default values (00:00:00). */

    Clock(int s);
    /* PROMISES: Initializes the clock based on the total number of seconds (since 00:00:00).
       REQUIRES: s to be a non-negative integer representing the total seconds. */

    Clock(int hours, int minutes, int seconds);
    /* PROMISES: Initializes the clock with the given hours, minutes, and seconds.
       REQUIRES: hours (0-23), minutes (0-59), and seconds (0-59). If out of range, resets to 00:00:00. */
```

```
int get_hour() const;
/* PROMISES: Returns the current hour of the clock (0-23). */

int get_minute() const;
/* PROMISES: Returns the current minute of the clock (0-59). */

int get_second() const;
/* PROMISES: Returns the current second of the clock (0-59). */

int get_time_in_seconds() const;
/* PROMISES: Returns the current time in seconds since 00:00:00. */

void set_hour(int h);
/* REQUIRES: h to be in the range of 0 to 23.
   PROMISES: Sets the hour of the clock to the specified value (0-23). */

void set_minute(int m);
/* REQUIRES: m to be in the range of 0 to 59.
   PROMISES: Sets the minute of the clock to the specified value (0-59). */

void set_second(int s);
/* REQUIRES: s to be in the range of 0 to 59.
   PROMISES: Sets the second of the clock to the specified value (0-59). */

void increment();
/* PROMISES: Increments the clock by one second, updating the time accordingly. */

void decrement();
```

```
    /* PROMISES: Decrements the clock by one second, updating the time accordingly. */


    void add_seconds(int s);

    /* REQUIRES: s to be a non-negative integer. If negative, prints an error message.

       PROMISES: Adds the specified number of seconds to the clock. If the total exceeds 24 hours, it wraps
around. */

    void set_time(int h, int m, int s);

private:

    int hour;

    int minute;

    int second;


    // Converts the time to seconds

    int hms_to_sec() const;

    /* PROMISES: Converts the current hour, minute, and second into total seconds. */


    // Converts seconds to hour, minute, second format

    void sec_to_hms(int s);

    /* REQUIRES: s to be a non-negative integer representing total seconds.

       PROMISES: Converts the given total seconds to hour, minute, and second format. */

};


#endif
```

Execution output

```
john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo/assignment3
$ ./eC.exe
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

```
//
//  CircularQueue.cpp
//  Circular Queue
// ENSF 694 Summer 2025 LAB 3 - EXERCISE D
//  Created by Mahmood Moussavi on 2024-04-09.
// implemented by: John Zhou

#include "CircularQueue.h"
#include <stdexcept>

CircularQueue::CircularQueue() : head(0), tail(0), count(0)
{
}

bool CircularQueue::isFull() const
{

    return count == (SIZE-1);
}

bool CircularQueue::isEmpty() const
{

    return count == 0;
```

```
}

int CircularQueue::enqueue(int v)
{
   if (isFull())
   {
      throw std::overflow_error("Queue is full. Cannot enqueue.");
   }
   arr[tail] = v;
   int inserted_index = tail;
   tail = (tail + 1) % SIZE;
   count++;
   return inserted_index;
}

int CircularQueue::dequeue()
{
   if (isEmpty())
   {
      throw std::underflow_error("Queue is empty. Cannot dequeue.");
   }
   int removed_index = head;
   head = (head + 1) % SIZE;
   count--;
   return removed_index;
}
int CircularQueue::counter() const
{
```

```
    return count;
}

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

void CircularQueue::displayQueue() const
{
    if (isEmpty())
    {
        std::cout << "Queue is empty." << std::endl;
        return;
    }
    int index = head;
    for (int i = 0; i < counter(); i++)
    {
        std::cout << arr[index] << std::endl;
        index = (index + 1) % SIZE;
    }
}
```

Exercise E

```
//
//  DynamicStack.cpp
//  Dynamic Stack

// ENSF 694 Summer 2025 LAB 3 - EXERCISE E
//  Created by Mahmood Moussavi on 2024-04-09.
//  implemented by: John Zhou

#include "DynamicStack.h"
#include <stdexcept>

DynamicStack::DynamicStack(int n)
{
   entry = 0;
   initial_capacity = n;
   current_capacity = n;
   array = new int[current_capacity];
}

DynamicStack::DynamicStack(DynamicStack const &stack)
{
   entry = stack.entry;
   initial_capacity = stack.initial_capacity;
   current_capacity = stack.current_capacity;
   array = new int[current_capacity];
   for (int i = 0; i < entry; ++i)
```

```
    {
        array[i] = stack.array[i];
    }
}

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

int DynamicStack::top() const
{
    if (entry == 0)
    {
        throw std::underflow_error("stack empty. no top");
    }
    return array[entry - 1];
}

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

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

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

DynamicStack &DynamicStack::operator=(DynamicStack const &rhs)
{
   if (this != &rhs)
   {
      delete[] array;
      entry = rhs.entry;
      initial_capacity = rhs.initial_capacity;
      current_capacity = rhs.current_capacity;
      array = new int[current_capacity];
      for (int i = 0; i < entry; ++i)
      {
         array[i] = rhs.array[i];
      }
   }
   return *this;
}

void DynamicStack::push(const int &obj)
{
   if (entry == current_capacity)
   {
      int new_capacity = current_capacity * 2;
      int *new_array = new int[new_capacity];
      for (int i = 0; i < entry; ++i)
```

```
        {
            new_array[i] = array[i];
        }
        delete[] array;
        array = new_array;
        current_capacity = new_capacity;
    }
    array[entry++] = obj;
}

int DynamicStack::pop()
{
    if (entry == 0)
    {
        throw std::underflow_error("stack empty. can't pop");
    }
    entry--;
    return array[entry];
}

void DynamicStack::clear()
{
    entry = 0;
    if (current_capacity != initial_capacity)
    {

        current_capacity = initial_capacity;
    }
    delete[] array;
```

```
    array = new int[initial_capacity];
}


void DynamicStack::display()
{
    if (entry == 0)
    {
        cout << "Stack is empty." << endl;
    }
    else
    {
        for (int i = 0 ; i < entry; i++)
        {
            cout << array[i] << " ";
        }
        cout << endl;
    }
}
```

Exercise F

```
String_Vector transpose(const String_Vector &sv)
{
    String_Vector vs;

    int rows = sv.size();
    int cols = sv[0].size();

    vs.resize(cols);

    for (int i = 0; i < cols; ++i)
    {
        for (int j = 0; j < rows; ++j)
        {
            vs[i].push_back(sv[j][i]);
        }
    }

    return vs;
}
```

Program output

```
john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo/assignment3
$ g++ -Wall lab3exe_F.cpp -o ef

john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo/assignment3
$ ./ef.exe
ABCD
EFGH
IJKL
MNOP
QRST
AEIMQ
BFJNR
CGKOS
DHLPT
```