ENSF 694 – Summer 2024

Principles of Software Development II

University of Calgary

Lab Assignment 3

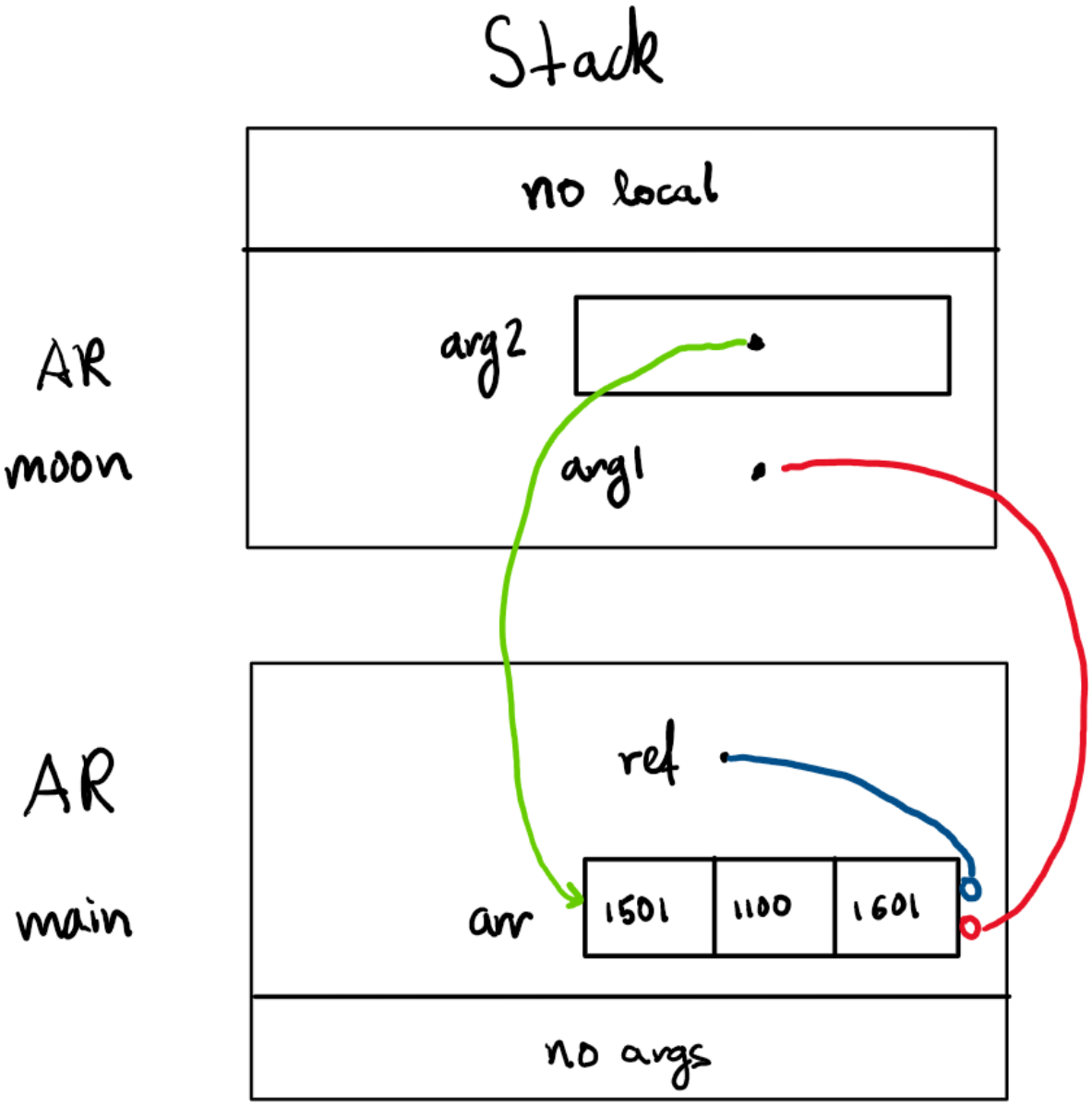
Student Name: Yael Gonzalez

Instructor: M. Moussavi, PhD, Peng

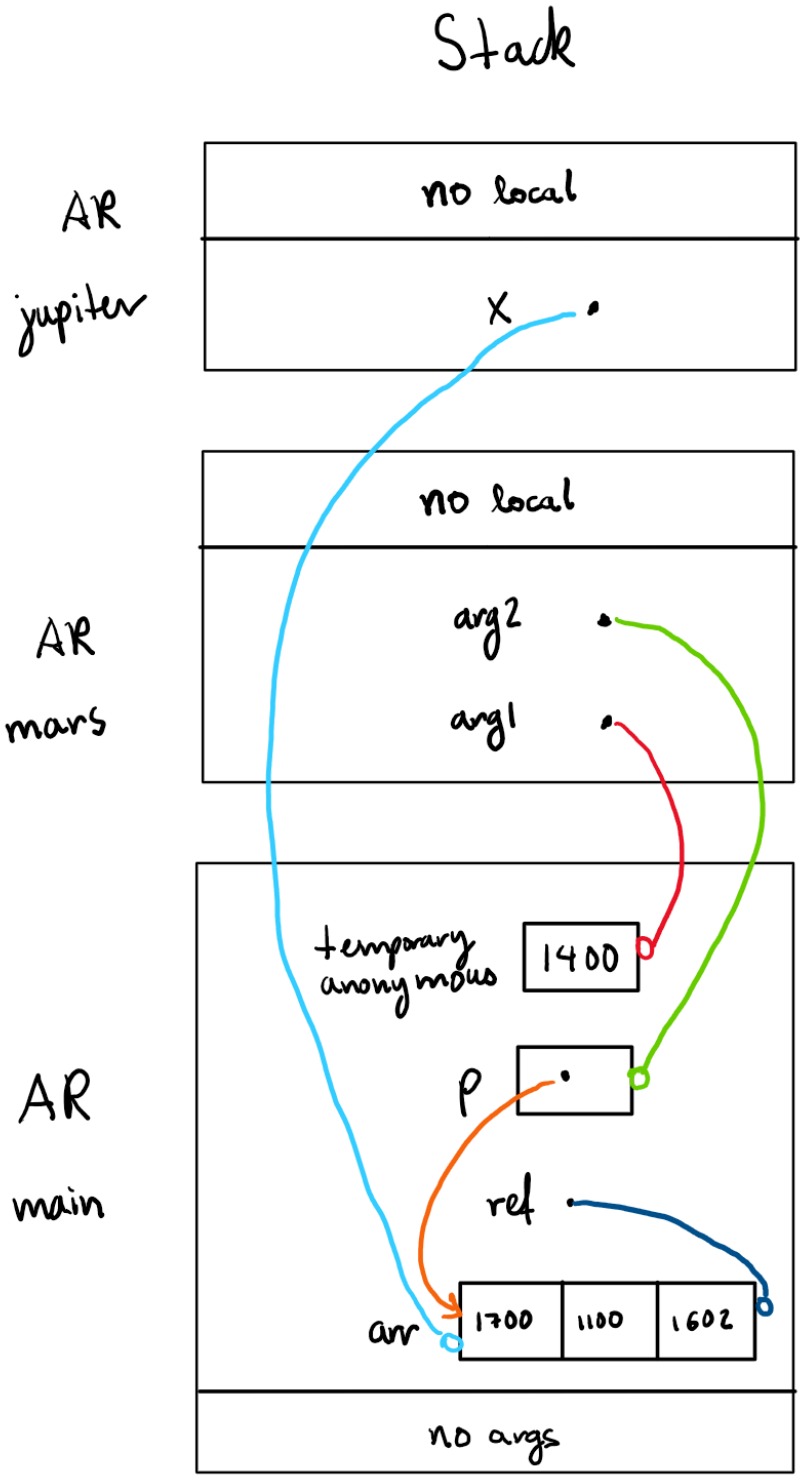
Submission Date: July 17, 2024

Exercise A

Point one

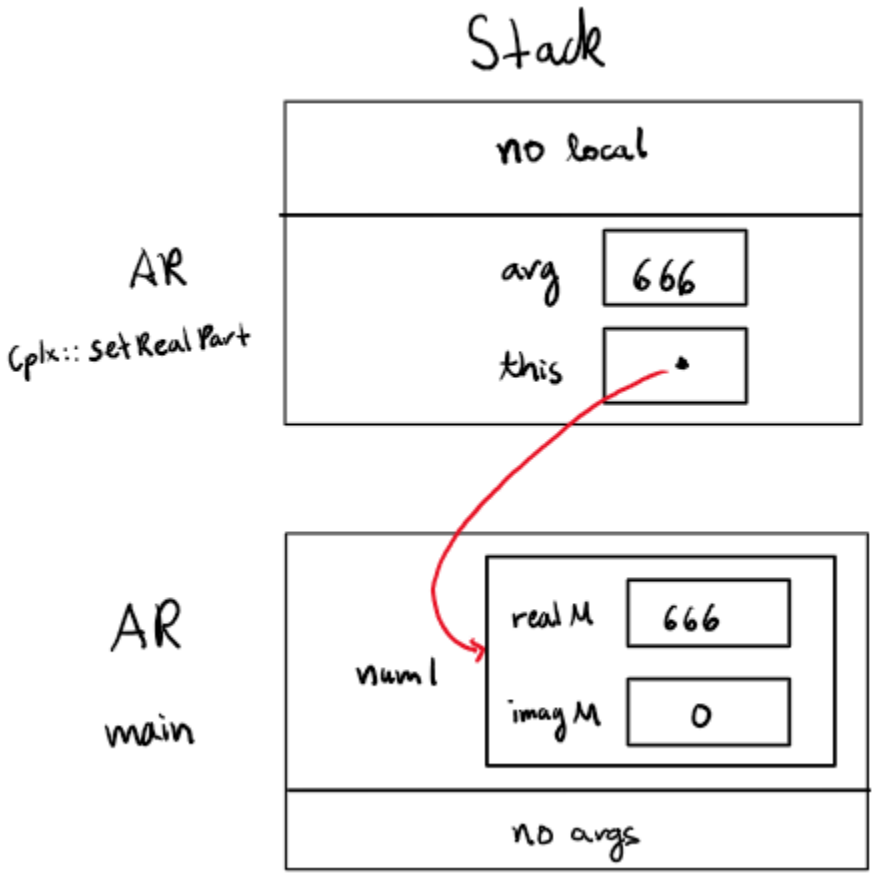


Point two

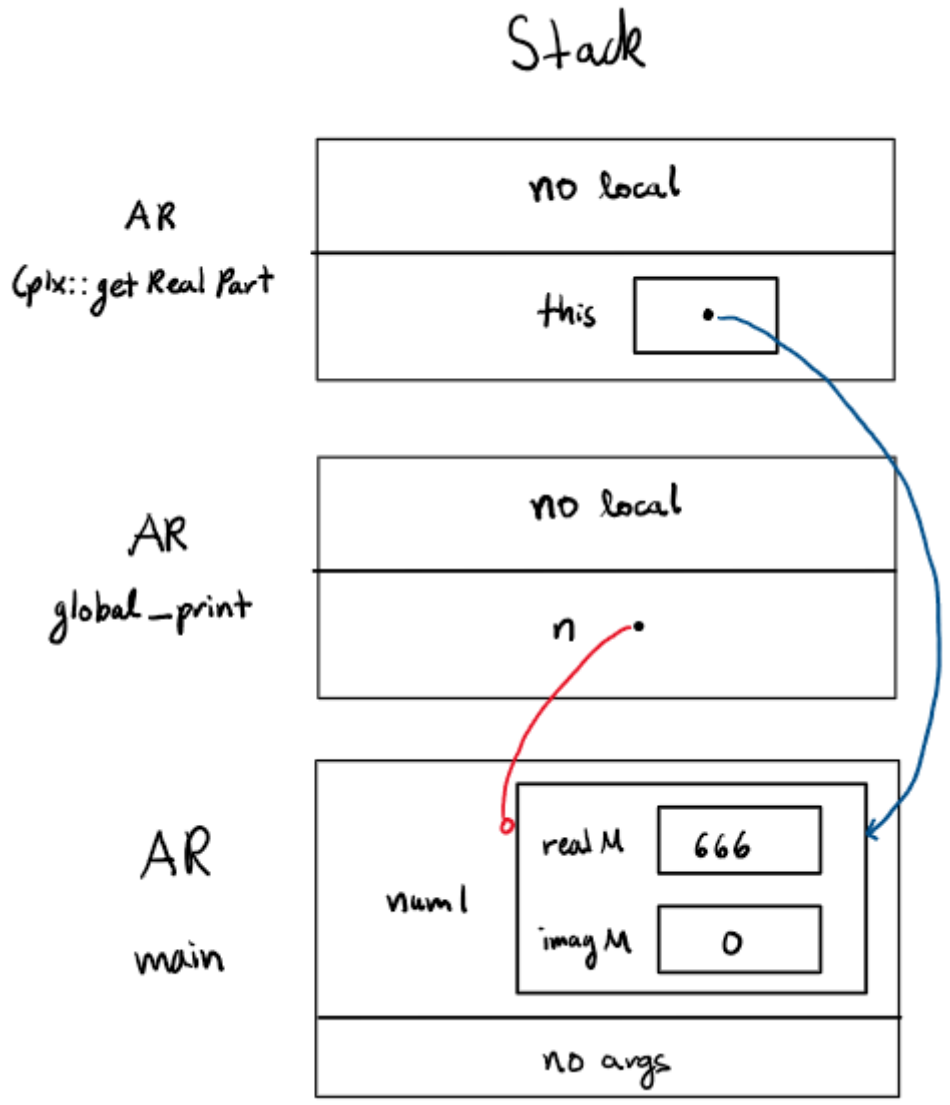

Stack
AR
jupiter
no local
x
no local
AR
mars
arg2
arg1
temporary
anonymous
1400
AR
main
p
ref
arr
1700
1100
1602
no args

Exercise B

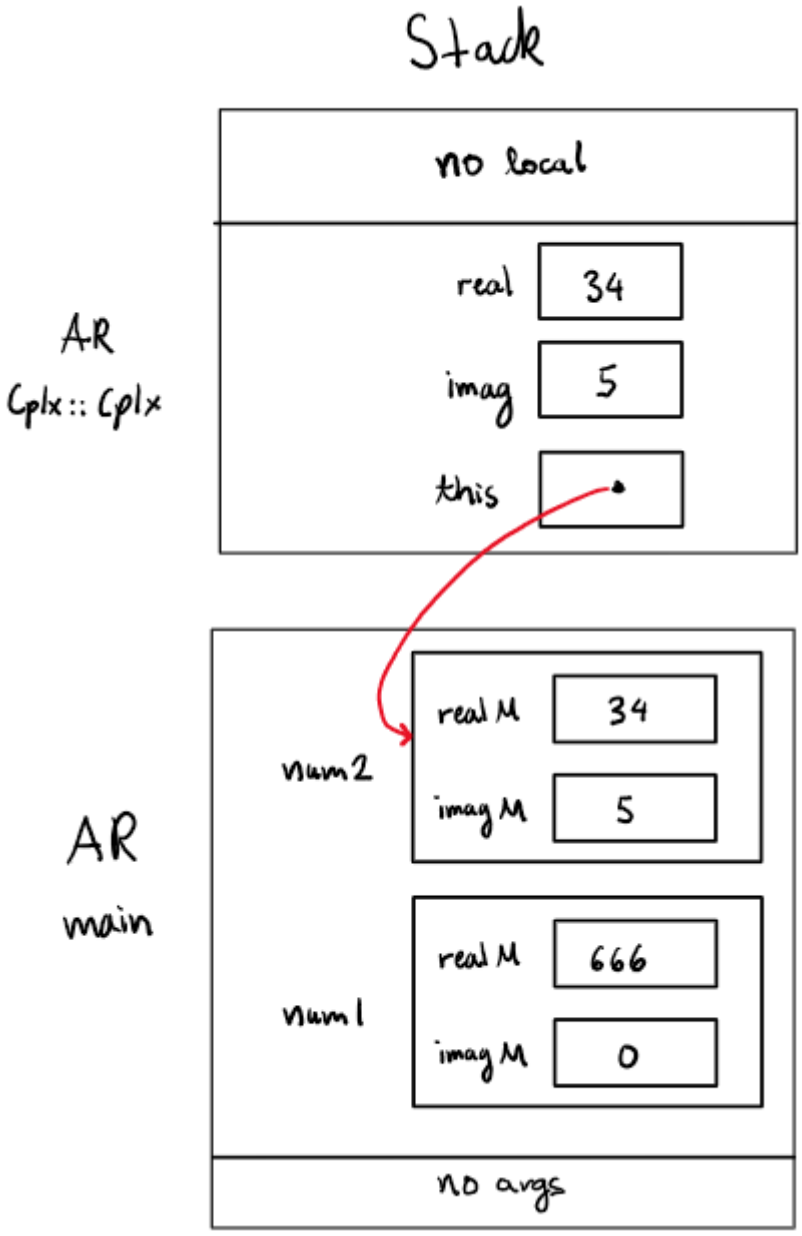
Point one



Point two


Stack
AR
Cplx::getRealPart
no local
this
AR
global_print
no local
n
AR
main
num1
realM
666
imagM
0
no args

Point three


Stack
no local
real
34
imag
5
this
AR
Cplx :: Cplx
real M
34
num2
imag M
5
AR
main
real M
666
num1
imag M
0
no args

Exercise C

Source code

lab3Clock.h

```
/**
 *  File Name: lab3Clock.h
 *  Assignment: ENSF 694 Summer 2024 - Lab 3 Exercise C
 *  Created by: Mahmood Moussavi
 *  Completed by: Yael Gonzalez
 *  Submission Date: July 17, 2024
 */

#ifndef lab3_exe_C_Clock
#define lab3_exe_C_Clock

#include <assert.h>
#include <iostream>

/**
 * The following class definition represnets a clock and contains three
 * private data members hour, minute, and second, which respectively describe a
clock
 * in the format hours:minutes:seconds.
 */
class Clock
{
public:
    Clock();
    /**
     * PROMISES:
     *  Initializes the clock with hour, minute, and second each set to zero.
     */

    Clock(int seconds);
    /**
     * PROMISES:
     *  Initializes the clock hour, minute, and second members respectively with
the
     *  converted supplied value of seconds.
     *
     *  For example, if the argument value is 4205, the values of data members
hour,
     *  minute and second will be: 1, 10, and 5 respectively.
```

```
     *
     *  If the given argument value is negative the data members will all be
initialized to zeros.
     */

    Clock(int h, int min, int sec);
    /**
     * PROMISES:
     *  Initializes the clock hour, minute, and second members with the
respective supplied
     *  arguments.
     *
     *  The data members will all be initialized to zeroes if any of the
following illegal
     *  operations is attempted:
     *  - The given values for second or minute are greater than 59 or less than
zero.
     *  - The given value for hour is greater than 23 or less than zero.
     */

    int get_hour() const;
    /**
     * PROMISES:
     *  Retuns the hours of a clock.
     */

    int get_minute() const;
    /**
     * PROMISES:
     *  Retuns the minutes of a clock.
     */

    int get_second() const;
    /**
     * PROMISES:
     *  Retuns the seconds of a clock.
     */

    void set_hour(int h);
    /**
     * PROMISES:
     *  Sets a new value to the hours of a clock with the value of h.
     */

    void set_minute(int min);
```

```
    /**
     * PROMISES:
     *  Sets a new value to the minutes of a clock with the value of m.
     */

    void set_second(int sec);
    /**
     * PROMISES:
     *  Sets a new value to the seconds of a clock with the value of s.
     */

    // Implementer functions
    void increment();
    /**
     * PROMISES:
     *  Increments the value of the clock's time by one.
     *  For example, if the value of the time is 00:00:00 it will change to
00:00:01.
     */

    void decrement();
    /**
     * PROMISES:
     *  Decrements the value of the clock's time by one.
     *  For example, if the value of the time is 00:00:00 it will change to
23:59:59.
     */

    void add_seconds(int seconds);
    /**
     * REQUIRES:
     *  seconds > 0, i.e., argument is a positive integer of seconds.
     * PROMISES:
     *  Adds the value of supplied argument seconds to the value of the current
time.
     *  For example, if the clock's time is 23:00:00, and the given argument is
3601 seconds,
     *  the time will change to: 00:00:01.
     */

private:
    int hour;   // Cannot be less than 0 or more than 23
    int minute; // Cannot be less than 0 or more than 59
    int second; // Cannot be less than 0 or more than 59
```

```
    int hms_to_sec();
    /**
     * PROMISE:
     *  Returns the total value of the data members in a Clock object, in
seconds.
     *
     *  For example, if the time value of a Clock object is 01:10:10, returns
4210
     *  seconds.
     */

    void sec_to_hms(int seconds);
    /**
     * PROMISE:
     *  Sets the total values for the Clock object data members with the supplied
value of
     *  seconds.
     *
     *  For example, if the supplied argument is 4210 seconds, the data members
values will be:
     *  1, 10 and 10, respectively for hour, minute, and second.
     */
};

#endif
```

lab3Clock.cpp

```
/**
 *  File Name: lab3Clock.cpp
 *  Assignment: ENSF 694 Summer 2024 - Lab 3 Exercise C
 *  Created by: Mahmood Moussavi
 *  Completed by: Yael Gonzalez
 *  Submission Date: July 17, 2024
 */

#include "lab3Clock.h"

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

Clock::Clock(int seconds) : hour(0), minute(0), second(0)
{
    if (seconds >= 0)
    {
```

```
        sec_to_hms(seconds);
    }
}

Clock::Clock(int h, int min, int sec) : hour(h), minute(min), second(sec)
{
    if (h < 0 || h > 23 || min < 0 || min > 59 || sec < 0 || sec > 59)
    {
        hour = 0;
        minute = 0;
        second = 0;
    }
}

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 (0 <= h && h <= 23)
    {
        hour = h;
    }
}

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

```cpp
void Clock::set_second(int sec)
{
    if (0 <= sec && sec <= 59)
    {
        second = sec;
    }
}

void Clock::increment()
{
    int curr_secs = hms_to_sec(); // Current clock in seconds
    sec_to_hms(curr_secs + 1);    // Increment clock by 1 second
}

void Clock::decrement()
{
    int curr_secs = hms_to_sec(); // Current clock in seconds
    if (curr_secs == 0)           // 00:00:00
    {
        sec_to_hms(86399); // Decrement to 23:59:59
    }
    else
    {
        sec_to_hms(curr_secs - 1); // Decrement clock by 1 second
    }
}

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

void Clock::sec_to_hms(int seconds)
{
    hour = (seconds / 3600) % 24;
    seconds %= 3600;
    minute = seconds / 60;
    second = seconds % 60;
}

void Clock::add_seconds(int seconds)
{
    assert(seconds > 0);
    int curr_secs = hms_to_sec();    // Current clock in seconds
    sec_to_hms(curr_secs + seconds); // Increment clock by specified seconds
```

```
}
```

Program output

```
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_C> g++ -Wall .\lab3Clock.cpp .\lab3exe_C.cpp -o MyProgram
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_C> .\MyProgram.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
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_C>
```

Exercise D

Source code

CircularQueue.h

```
/**
 *  File Name: CircularQueue.h
 *  Assignment: ENSF 694 Summer 2024 - Lab 3 Exercise D
 *  Created by: Mahmood Moussavi on 2024-04-09.
 *  Completed by: Yael Gonzalez
```

```
 *  Submission Date: July 17, 2024
 */

#ifndef CircularQueue_h
#define CircularQueue_h

#include <iostream>
using namespace std;
#define SIZE 10 // Set to 4 to test data1.txt. Set to 10 to test data2.txt.
typedef int TYPE;

class CircularQueue
{
private:
    TYPE head;      // position of head
    TYPE tail;      // position of tail
    TYPE arr[SIZE]; // a queue array with maximum SIZE elements
    TYPE count;     // keeps track of number of valid data in the queue
public:
    CircularQueue();
    /* PROMISES: set initial values for head, tail and count. Also, initializes
Queue (array) with zeroes */
    bool isFull() const;
    /* PROMISES: return value is true if queue is full */
    bool isEmpty() const;
    /* PROMISES: return value is true if queue is empty */
    TYPE enqueue(int v);
    /* PROMISES: adds value v to the tail, increments count of values and returns
the new position of the tail */
    TYPE dequeue();
    /* PROMISES: returns the position of the element representing, eliminates its
value and decrements count */
    void displayQueue() const;
    /* PROMISES: dispalys the existing values in the queue */
    TYPE counter() const;
    /* PROMISES: returns the number of values in the queue */
    const TYPE *get_arr() const;
    /* PROMISES: returns the address of the array arr */
};
#endif /* CircularQueue_h */
```

CircularQueue.cpp

```
/**
```

```
 *  File Name: CircularQueue.cpp
 *  Assignment: ENSF 694 Summer 2024 - Lab 3 Exercise D
 *  Created by: Mahmood Moussavi on 2024-04-09.
 *  Completed by: Yael Gonzalez
 *  Submission Date: July 17, 2024
 */

#include "CircularQueue.h"

CircularQueue::CircularQueue() : head(1), tail(1), count(0) {}

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

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

int CircularQueue::enqueue(int element)
{
    if (isFull())
    {
        cout << "Error: Queue is full" << endl;
    }
    else
    {
        tail = (tail + 1) % SIZE;
        arr[tail] = element;
        count++;
    }

    return tail;
}

int CircularQueue::dequeue()
{
    if (isEmpty())
    {
        cout << "Error: Queue is empty" << endl;
    }
    else
    {
```

```
        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
{
    int idx = (head + 1) % SIZE;

    if (isEmpty())
    {
        cout << "Queue is empty";
    }
    else
    {
        for (int i = 0; i < count; i++)
        {
            cout << arr[idx] << " ";
            idx++;
        }
    }

    cout << endl;
}
```

Program output

Testing data1.txt

```
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_D> g++ -Wall .\CircularQueue.cpp .\CircularQueue_tester.cpp -o myCircularQueue
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_D> .\myCircularQueue.exe
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
Exiting...
Here is the content of the circular queue at the end of program:
Queue is empty
Finishing Test Run

Program Ended ....
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_D>
```

Testing data2.txt

```
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_D> g++ -Wall .\CircularQueue.cpp .\CircularQueue_tester.cpp -o myCircularQueue
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_D> .\myCircularQueue.exe
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:
1000 2000 3000
Finishing Test Run

Program Ended ....
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_D>
```

Exercise E

Source code

DynamicStack.h

```
/**
 *  File Name: DynamicStack.h
 *  Assignment: ENSF 694 Summer 2024 - Lab 3 Exercise E
 *  Created by: Mahmood Moussavi on 2024-04-09.
 *  Completed by: Yael Gonzalez
 *  Submission Date: July 17, 2024
 */

#ifndef DynamicStack_H
#define DynamicStack_H

#include <iostream>
using namespace std;

class DynamicStack
{
private:
    int entry;
    int initial_capacity;
    int current_capacity;
    int *array;

public:
    DynamicStack(int n = 5);
    /* PROMISES: Sets value of entry to zero, initial_capacity and
current_capacity to n,
     * and allocates memory space for array */

    /* copy ctor*/
    DynamicStack(DynamicStack const &);
    /* PROMISES: A copy of the stack initialized with its current members */

    ~DynamicStack();
    /* PROMISES: Deallocates memory space allocated for array*/

    int top() const;
    /* PROMISES: Returns the value at the top of the stack */

    int size() const;
```

```
    /* PROMISES: Returns the number of values stored in the stack */

    bool empty() const;
    /* PROMISES: Returns true if stack is empty */

    int capacity() const;
    /* PROMISES: Returns current capacity of the array in the stack */

    DynamicStack &operator=(const DynamicStack &);
    /* PROMISES: Handles assignment operation for a DynamicStack object */

    void push(const int &v);
    /* PROMISES: The value of v is added at the top of the stack. If the array is
full,
     * the current capacity of the array is doubled, then the value of v is
added.
     */

    int pop();
    /* PROMISES: Removes the element at the top of the stack. If, only one fourth
of
     * the array is full,  and the array's capacity is greater than it's initial
capacit, the
     * capacity of the array is halved.
     */

    void clear();
    /* PROMISES: Removes all the entry values from the stack and if necessary
resizes
     * the array to its initial capacity.
     */

    void display();
    /* PROMISES: Displays all the existing values in the stack */
};

#endif
```

DynamicStack.cpp

```
/**
 *  File Name: DynamicStack.cpp
 *  Assignment: ENSF 694 Summer 2024 - Lab 3 Exercise E
 *  Created by: Mahmood Moussavi on 2024-04-09.
```

```
 *  Completed by: Yael Gonzalez
 *  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 &stack) : entry(stack.entry),
                                                        initial_capacity(stack.in
itial_capacity),
                                                        current_capacity(stack.cu
rrent_capacity),
                                                        array(new
int[stack.current_capacity])
{
    if (stack.entry > 0) // If no values in the array, don't copy
    {
        std::copy(stack.array, stack.array + stack.entry, this->array); // Copy
of the array
    }
}

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

int DynamicStack::top() const
{
    if (empty())
    {
        std::cout << "Cannot peek: Stack is empty." << endl;
        return -1;
    }

    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) // avoid self copy
    {
        delete[] array;
        this->entry = rhs.entry;
        this->initial_capacity = rhs.initial_capacity;
        this->current_capacity = rhs.current_capacity;
        this->array = new int[rhs.current_capacity];
        if (rhs.entry > 0) // If no values in the array, don't copy
        {
            std::copy(rhs.array, rhs.array + rhs.entry, this->array); // Copy of
the array
        }
    }

    return *this;
}

void DynamicStack::push(const int &obj)
{
    if (entry == current_capacity)
    {
        current_capacity *= 2;
        DynamicStack temp(*this); // temp value using copy ctor
        *this = temp;             // assignment op
    }

    entry++;
    array[entry - 1] = obj;
}

int DynamicStack::pop()
```

```
{
    if (empty())
    {
        std::cout << "Cannot pop: Stack is empty." << endl;
        return -1;
    }

    int popped_value = array[entry - 1];
    entry--;

    if (entry == current_capacity / 4 && current_capacity > initial_capacity)
    {
        current_capacity /= 2;
        DynamicStack temp(*this); // temp value using copy ctor
        *this = temp;             // assignment op
    }

    return popped_value;
}

void DynamicStack::clear()
{
    DynamicStack temp(initial_capacity); // temp value using default ctor
    *this = temp;                        // assignment op
}

void DynamicStack::display()
{
    for (int i = 0; i < entry; i++)
    {
        std::cout << array[i] << " ";
    }

    std::cout << std::endl;
}
```

Program output

There is a small mistake in the DynamicStack_tester.cpp, it is pushing 8 values 1000, 2000, 3000, 4000, **5000**, 10000, 13000, and 14000 into the stack, but the legend of expected values to be displayed says "*Expected vlues are : 122 452 1000 2000 3000* ***40000*** *4000 10000 13000 14000*". For that reason, the corresponding line in the DynamicStack_tester.cpp was modified in my local copy to display the correct value of ***40000****, as follows:*

```
std::cout << "\nPushing 8 more values into the stack ..." << endl;
stack->push(1000);
stack->push(2000);
stack->push(3000);
stack->push(40000);  Line added
stack->push(4000);
// stack->push(5000);  Line commented out
stack->push(10000);
stack->push(13000);
stack->push(14000);
std::cout << "Expected vlues are : 122 452 1000 2000 3000 40000 4000 10000 13000 14000 " << endl;
std::cout << "Actual values are: ";
stack->display();
```

Output is:

```
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_E> g++ -Wall .\DynamicStack.cpp .\DynamicStack_tester.cpp -o myDynamicStack
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_E> .\myDynamicStack.exe
Stack of 5 elements is created.

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

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

Pushing 8 more values into the stack ...
Expected vlues are : 122 452 1000 2000 3000 40000 4000 10000 13000 14000
Actual values are: 122 452 1000 2000 3000 40000 4000 10000 13000 14000

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

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

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

Checking whether stack is emoty or not:
Stack is not empty.
Stack still holds:
122
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_E>
```

Exercise F

Source code

```
/**
 *  File Name: lab3exe_F.cpp
 *  Assignment: ENSF 694 Summer 2024 - Lab 3 Exercise F
 *  Created by: Mahmood Moussavi on 2024-04-09.
 *  Completed by: Yael Gonzalez
 *  Submission Date: July 17, 2024
 */
```

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

typedef vector<string> String_Vector;

String_Vector transpose(const String_Vector &sv);
// 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.

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 Matrix:" << endl;

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

```
        cout << endl;
    }

    cout << "\nTranspose Matrix:" << endl;

    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 vs;                   // transposed matrix
    const int rows = (int)sv[0].size(); // vs(#rows) = sv(#cols)
    const int cols = (int)sv.size();    // vs(#cols) = sv(#rows)
    vs.resize(rows);

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

    return vs;
}
```

Program output

```
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_F> g++ -Wall .\lab3exe_F.cpp -o .\lab3exe_F
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_F> .\lab3exe_F.exe
Original Matrix:
ABCD
EFGH
IJKL
MNOP
QRST

Transpose Matrix:
AEIMQ
BFJNR
CGKOS
DHLPT
PS C:\Users\Owner\Desktop\Calgary\ENSF694\assignments\a3-ensf694\ex_F>
```