Course: ENSF 614 - Fall 2023

Lab B01: Lab 2

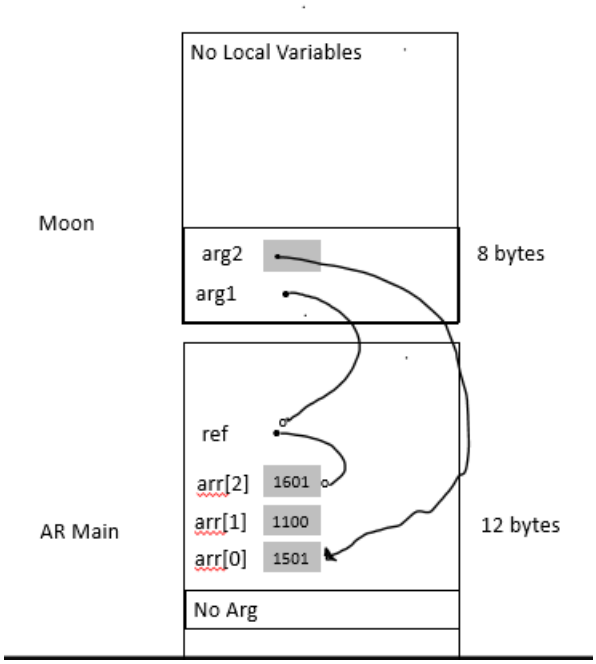
Instructor: Mahmood Moussavi

Student Name: Braden Tink

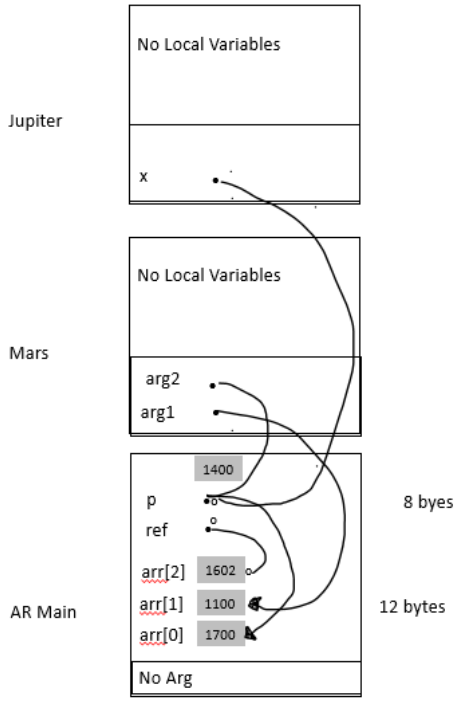
Submission Date: October 13, 2023

# Exercise A

## Point 1

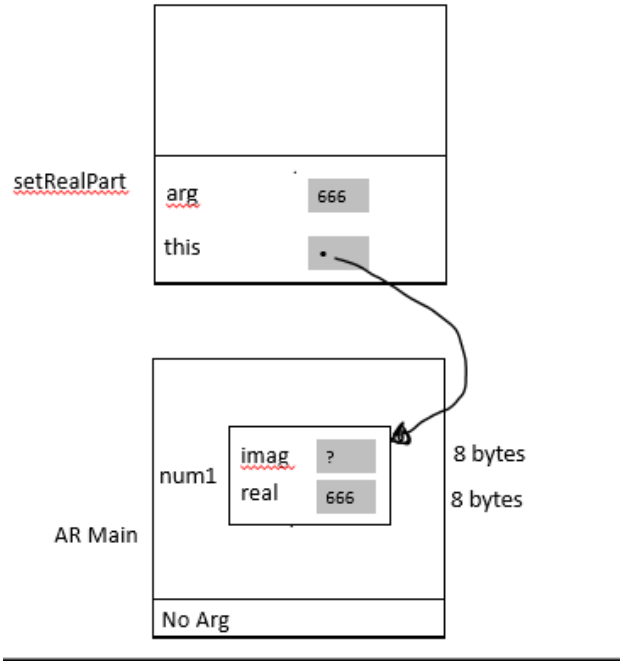



## Point 2

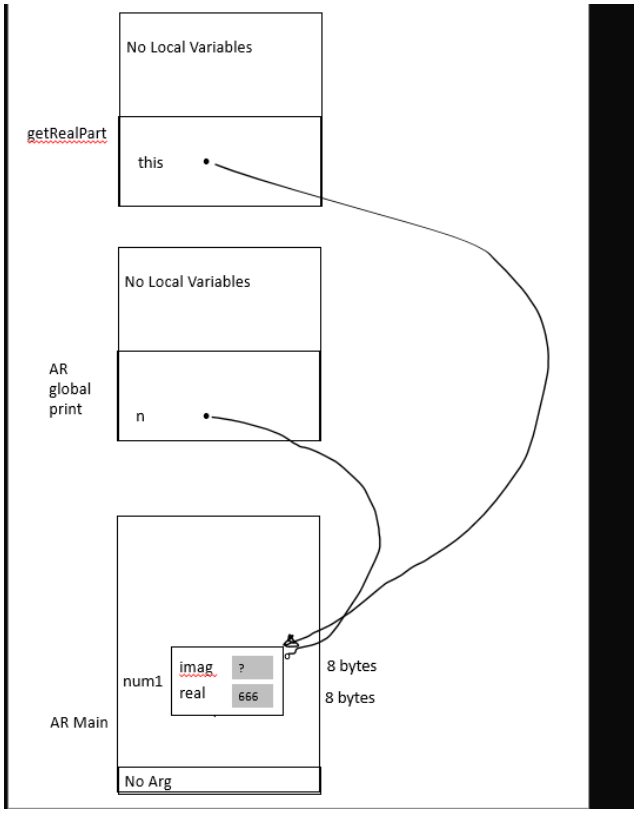


# Exercise B

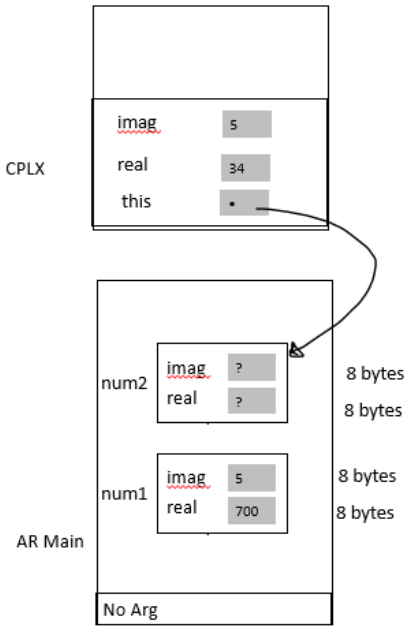
## Point 1




## Point 2



Point 3




# Exercise C

```
// lab3Clock.h
// ENSF 614 Lab 3 Exercise C

#ifndef LAB3CLOCK_H
#define LAB3CLOCK_H
class Clock {
public:
        Clock ();

        Clock (int tot);

        Clock (int second, int minute, int hour);


        int get_second() const;

        int get_minute() const;

        int get_hour() const;

        void set_second(int sec);

        void set_minute(int min);
```

```
        void set_hour(int hour);

        void add_seconds(int sec);

        void add_minute(int min);

        void increment();

        void decrement();



private:
        int second;  // the second part of the clock
    int minute;  // the minute part of the clock
        int hour;           // the hour part of the clock

        int hms_to_sec();
        void sec_to_hms(int sec);

};

#endif //LAB3_CLOCK_H


// lab3Clock.cpp
// ENSF 614 Lab 3 Exercise C

#include "lab3Clock.h"
#include <iostream>
using namespace std;


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

}

Clock::Clock(int tot){
        int total = tot;
        if(total < 0){
                set_hour(0);
                set_minute(0);
                set_second(0);
        }else{
                if(total >= 86400){
                        total -= 86400;
                }

        }
        sec_to_hms(total);
```

```
}

Clock::Clock(int hour, int minute, int second){
	if((hour < 0 || minute < 0 || second < 0) || (hour > 23 || minute >59 || second > 59)){
		set_second(0);
		set_minute(0);
		set_hour(0);
	}
	else{
		set_second(second);
		set_minute(minute);
		set_hour(hour);
	}
}

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


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

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

void Clock::set_second(int sec){
	if(sec >= 60 || sec < 0 ){
		second = this->get_second();
	}
	else{
		second = sec;
	}
}

void Clock::set_minute(int min){
	if(min >= 60 || min < 0 ){
		minute = this->get_minute();
	}
	else{
		minute = min;
	}
}

void Clock::set_hour(int hr){
	if(hr > 24 || hr < 0 ){
		hour = this->get_hour();
	}
	else{
```

```
                hour = hr;
        }
}

void Clock::add_seconds(int sec){
        int total = 0;
        total = hms_to_sec();
        total = total + sec;

        cout << total;

        while(total >= 86400){
                total -= 86400;
        }
        sec_to_hms(total);
}

void Clock::increment(){

        int total;

        total = this -> get_second() + (this -> get_minute()*60) + (this -> get_hour()*3600);

        total += 1;

        if(total == 86400){
                this -> set_hour(0);
                this -> set_minute(0);
                this -> set_second(0);
        }
        else {

                sec_to_hms(total);
        }

}

void Clock::decrement(){
        int total;

        total = this -> get_second() + (this -> get_minute()*60) + (this -> get_hour()*3600);
        total -= 1;
        if (total == -1){
                this ->set_hour(23);
                this ->set_minute(59);
                this ->set_second(59);
        }
        else{
                this -> set_hour(total / 3600);
                total -= ((this -> get_hour())*3600);
```

```
                this -> set_minute(total/60);
                total -= ((this -> get_minute())*60);

                this -> set_second(total);
        }

}

int Clock::hms_to_sec(){
        int total = 0;
        total = this-> get_second() + (this->get_minute()*60) + (this->get_hour()*3600);
        return total;

}

void Clock::sec_to_hms(int sec){

        int temp = sec;

        this -> set_hour(sec/3600);

        temp -= hour*3600;

        this -> set_minute(temp/60);

        temp -= minute*60;

        this -> set_second(temp);

}
```

Output

```
Braden@TBLaptop04 /cygdrive/c/users/braden/documents/school/ENSF 614/Assignments/Assignment 3
$ ./a.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
86400Adding 2350 seconds to t2. Expected time is: 00:00:00
00:00:00
72000Adding 72000 seconds to t2. Expected time is: 20:00:00
20:00:00
288000Adding 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

```
//

#include "MyArray.h"
#include <assert.h>
#include <iostream>
using namespace std;

MyArray::MyArray(){
	// Create empty array.
	// PROMISES: size() == 0.

	resize(0);
	storageM = new EType[0];

}


MyArray::MyArray(const EType *builtin, int sizeA){
	// Create object by copying a built-in array.
	// REQUIRES
	//   sizeA >= 0. Elements builtin[0] ... builtin[sizeA - 1] exist.
	// PROMISES
	//   size() == sizeA.
	//   For i from 0 to sizeA-1, element i of object == builtin[i].

	if(sizeA >= 0){
		sizeM = sizeA;
		storageM = new EType[sizeM];

		for(int i = 0; i < sizeM; i++){
			storageM[i] = builtin[i];
		}
	}

}

MyArray::MyArray(const MyArray& source){
	sizeM = source.size();
	storageM = new double[sizeM];
	assert(storageM!=0);

	for(int i = 0; i < sizeM; i++){
		storageM[i] = source.storageM[i];
	}


}
// copy .....
MyArray& MyArray::operator = (const MyArray& rhs){
```

```
	//return rhs;

	if(this != &rhs){
		delete[] storageM;
		sizeM = rhs.sizeM;
		storageM = new EType[sizeM];


		for(int i = 0;i<sizeM;i++){
			storageM[i] = rhs.storageM[i];
		}

	}
	return *this;
}


MyArray::~MyArray(){
	delete this;



}

int MyArray::size() const{

	return sizeM;

}

EType MyArray::at(int i) const{


	return storageM[i];
	// REQUIRES: 0 <= i && i < size().
	// PROMISES: Return value is a reference to element i of array.


}

void MyArray::set(int i, EType new_value){


	storageM[i] = new_value;
	// REQUIRES: 0 <= i && i < size().
	// PROMISES: assigns new_value to the ith element of storageM
}

void MyArray::resize(int new_size){

	EType* new_Stor = nullptr;
```

```
        int temp_size;

        if(sizeM < new_size){
                temp_size = sizeM;
        }
        else{
                temp_size = new_size;
        }


        if (new_size >= 0){
                new_Stor = new EType[new_size];

                for(int i = 0; i < temp_size; i++){
                        new_Stor[i] = storageM[i];
                }

                delete[] storageM;
                storageM = new_Stor;
                sizeM = new_size;


        }else{

        }
}
```

Output

```
Braden@TBLaptop04 /cygdrive/c/users/braden/documents/school/ENSF 614/Assignments/Assignment 3
$ ./a.exe
Elements of a:  0.5 1.5 2.5 3.5 4.5
(Expected:      0.5 1.5 2.5 3.5 4.5)

Elements of b after first resize:  10.5 11.5 12.5 13.5 14.5 15.5 16.5
(Expected:                         10.5 11.5 12.5 13.5 14.5 15.5 16.5)

Elements of b after second resize:  10.5 11.5 12.5
(Expected:                          10.5 11.5 12.5)

Elements of b after copy ctor check:  10.5 11.5 12.5
(Expected:                            10.5 11.5 12.5)

Elements of c after copy ctor check:  -1.5 11.5 12.5
(Expected:                            -1.5 11.5 12.5)

Elements of a after operator = check:  -10.5 1.5 2.5 3.5 4.5
(Expected:                             -10.5 1.5 2.5 3.5 4.5)

Elements of b after operator = check:  -11.5 1.5 2.5 3.5 4.5
(Expected:                             -11.5 1.5 2.5 3.5 4.5)

Elements of c after operator = check:  0.5 1.5 2.5 3.5 4.5
(Expected:                             0.5 1.5 2.5 3.5 4.5)
```