ENSF 619-Fall 2020

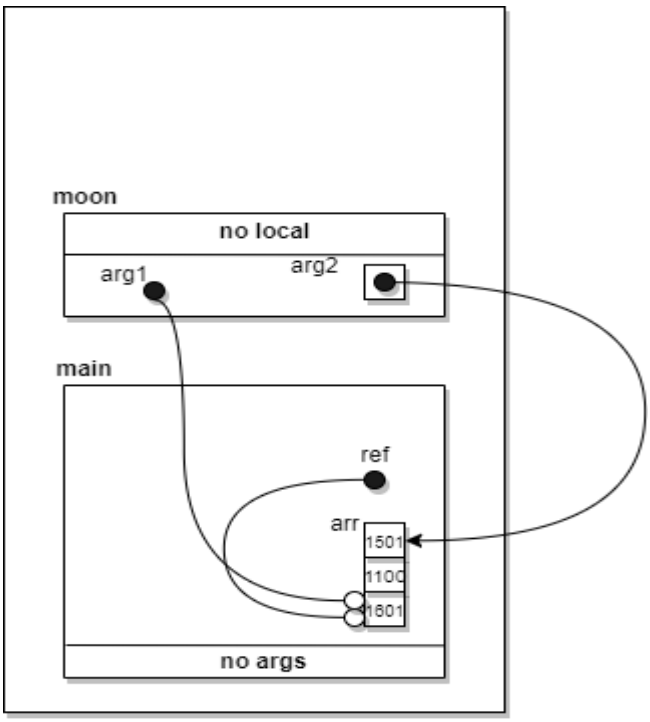
Lab# 3

Ziad Chemali

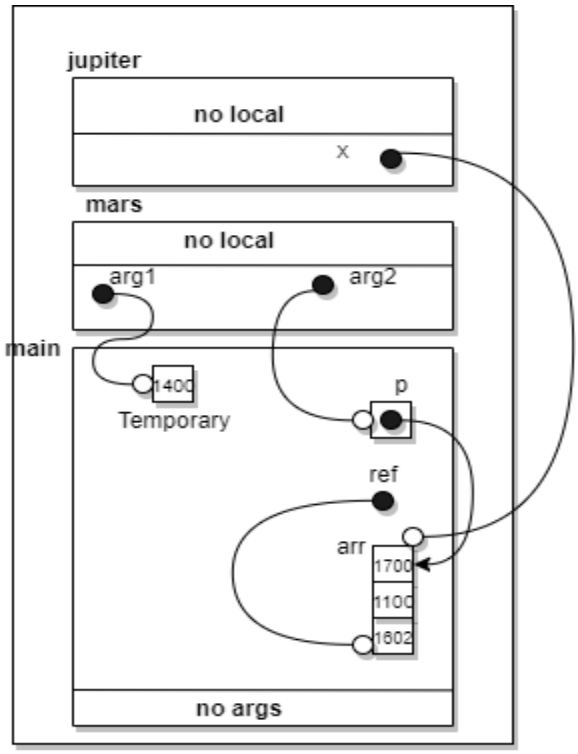
October 9, 2020

I] Exercise: A

1) Point-1

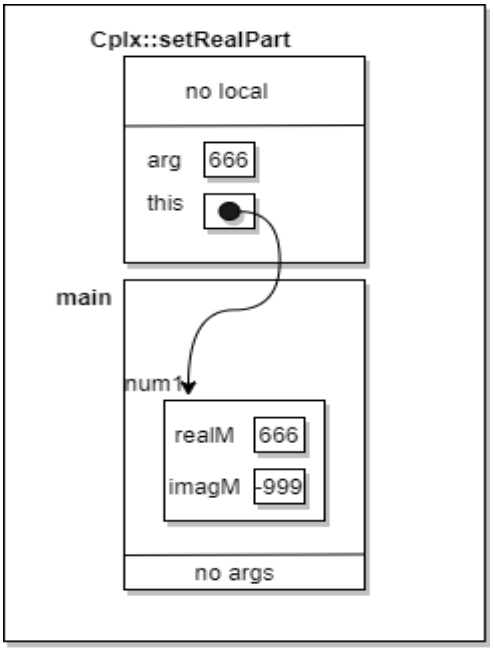


2) Point-2

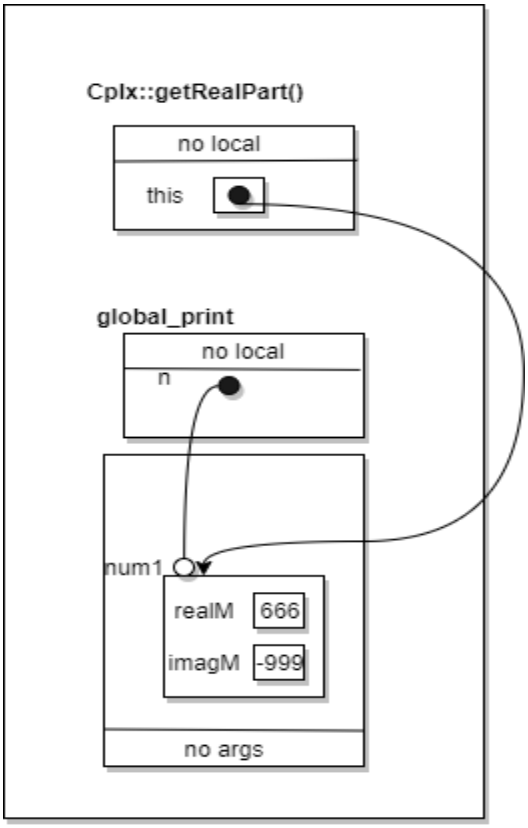


II] Exercise: B

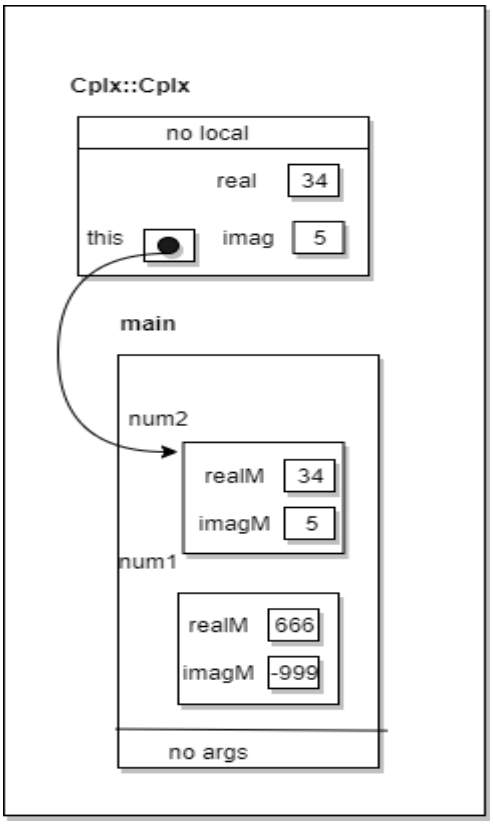
1) Point-1



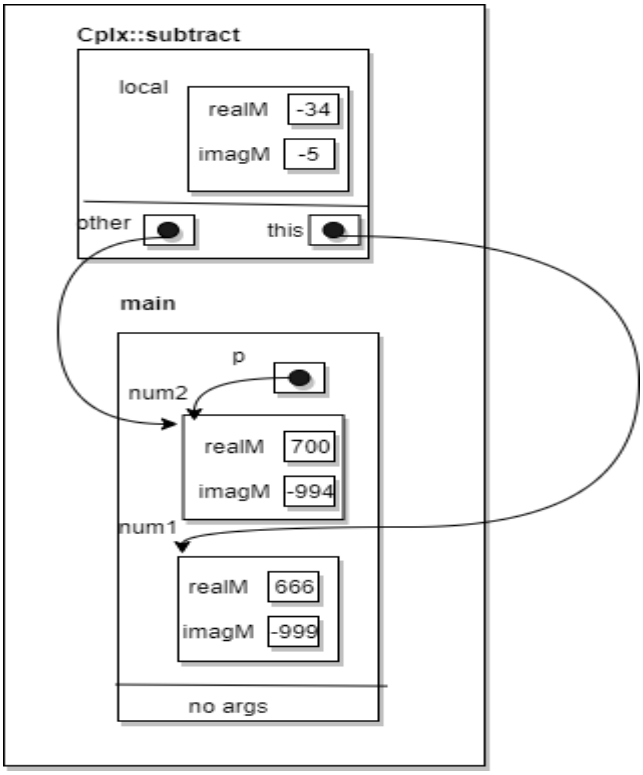
2) Point-2



3) Point-3



4) Point-4



III] Exercise: C

1) Code:

Header file:

```
/*
* File Name: lab3Clock.h
* Lab #3
* Completed by: Ziad Chemali
* Submission Date: October 9,2020

*/

#ifndef lab3_exe_C_Clock
#define lab3_exe_C_Clock
/* The following class definition is for Clock that represents 24 hour, and is able to
* create a clock, increment time by one second, or decrement time by one second.
*
 */

class Clock {
public:
    Clock();
    // PROMISES: initializes hours,minutes, and seconds to zero
    Clock(int s);
    // PROMISES: initialize the clock data members with values for hour,minute, and
second in this
    // argument
    Clock(int h, int m, int s);
    // PROMISES: initialize the data members hour, minute, and second with h,m, and s
    int get_hour() const;
    // PROMISES: returns hour
    int get_minute() const;
    // PROMISES: return minute
    int get_second() const;
    // PROMISES: return seconds
    void set_hour(int h);
    // PROMISES: updates hour data member with h
    void set_minute(int m);
    // PROMISES: updates the minute data member with m
    void set_second(int s);
    // PROMISES: updates the second data member with s
    void increment();
    // PROMISES: increments the value of clocks time by one second
    void decrement();
    // PROMISES: decrements the value of the clocks time by one second
    void add_seconds(int s);
    // PROMISES: adds the value of s to the value of current time
private:
    int hms_to_sec();
    // PROMISES: returns the total value of data members in a Clock
    void sec_to_hms(int s);
    // PROMISES: sets the data members of clock to the value of s
    int hour;
    int minute;
    int second;
};
```

```
#endif
```

Cpp file:

```
/*
* File Name: lab3Clock.cpp
* Lab #3
* Completed by: Ziad Chemali
* Submission Date: October 9,2020

*/

#include<iostream>
#include <iomanip>
# include "lab3Clock.h"

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

Clock::Clock(int s) {
	if (s < 0) {
		hour = 0;
		minute = 0;
		second = 0;
	}
	else {
		sec_to_hms(s);
	}
}

Clock::Clock(int h, int m, int s) {
	if ((h >= 0 && h <= 23) && (m >= 0 && m <= 60) && (s >= 0 && s <= 60)) {
		this->hour = h;
		this->minute = m;
		this->second = s;

	}
	else
	{
		hour = 0;
		minute = 0;
		second = 0;
	}


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

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

int Clock::get_second() const {
	return this->second;
```

```
}

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

}

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

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

void Clock::increment() {
	if (this->second < 59)
		this->second++;
	else if (this->second == 59)
	{
		this->second = 0;
		if (this->minute < 59)

			this->minute++;
		else if (this->minute == 59)
		{
			this-> minute = 0;
			if (this->hour < 23)
				this->hour++;
			else if (hour == 23)
			{
				hour = 0;


			}
		}

	}
}
void Clock::decrement() {
	if (this->second >0)
		this->second--;
	else if (this->second == 0)
	{
		this->second = 59;
		if (this->minute >0)

			this->minute--;
		else if (this->minute == 0)
		{
			this->minute = 59;
			if (this->hour >0)
				this->hour--;
			else if (hour == 0)
			{
```

```
                        hour = 23;


                    }
                }

        }


}
void Clock::add_seconds(int s) {
        if (s >= 0)
        {
                int local = hms_to_sec();
                local += s;

                sec_to_hms(local);
        }

}
void Clock::sec_to_hms(int s) {



                double local = double(s) / (24.0 * 60 * 60);
                int day = local;

                double h = local - day;

                h *= 24;

                this->hour = h;
                double min = h - int(h);



                min *= 60;

                this->minute = min;
                long double sec = min - int(min);



                second = sec*60;




}
int Clock::hms_to_sec() {
        std::cout << "hms to sec  " <<  (this->second) << " \n";

        return (second+60*minute+hour*3600);
}
```

2) Output:

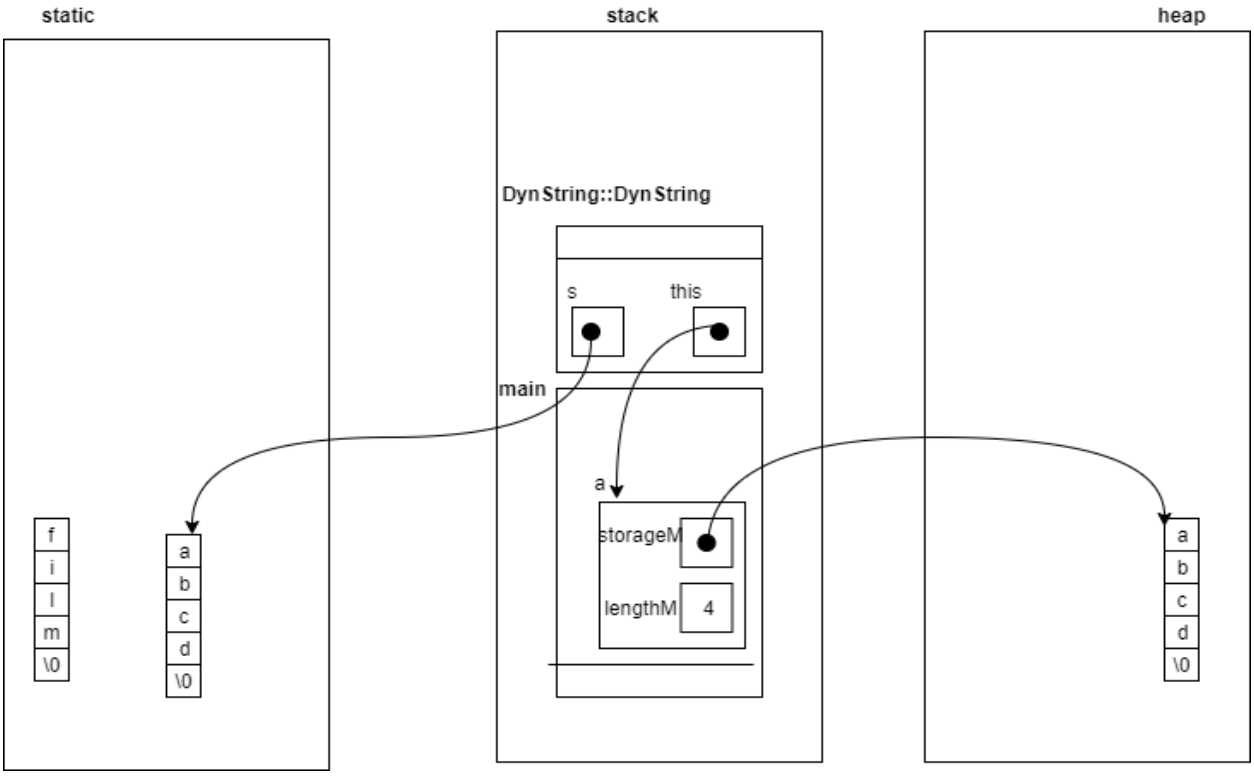
Microsoft Visual Studio Debug Console

```
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
hms to sec  50
Adding 2350 seconds to t2. Expected time is: 00:00:00
00:00:00
hms to sec  0
Adding 72000 seconds to t2. Expected time is: 20:00:00
20:00:00
hms to sec  0
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
```

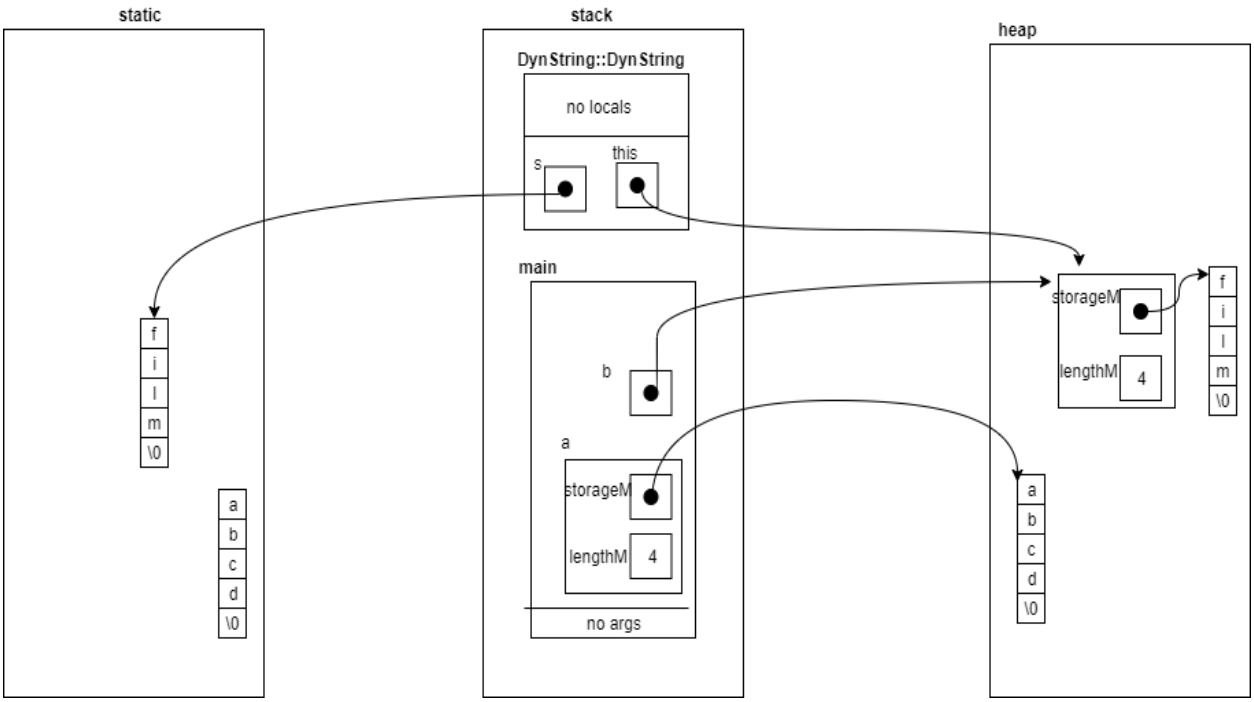
IV] Exercise: D

1) Part-One:

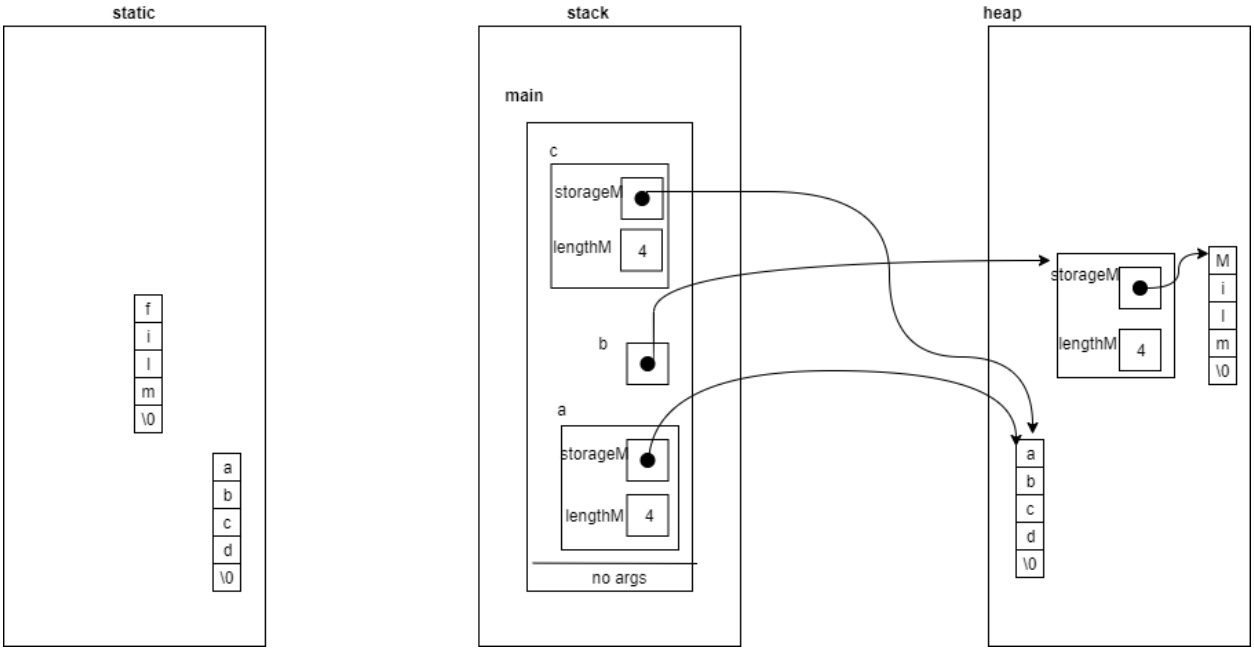
Point One: First Time



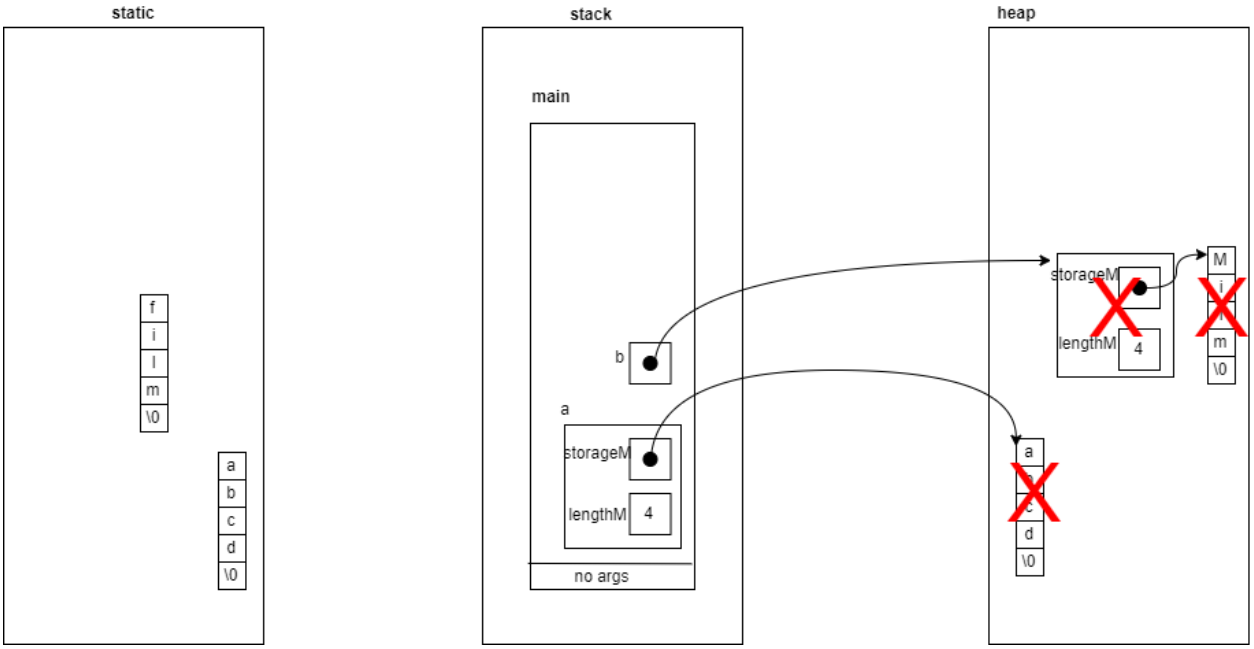
Point One: Second Time



Point Three:



Point Four:



Questions

1) At point 4 the constructor has been called two times
2) At point 4 the destructor has been called two times
3) Overall, the destructor has been called three times
4) Error is because the destructor at the end of main is trying to deallocate the char array “abcd” which already been deallocated when the inner bracket is finished.

2) Part-Two:

i) Code

```
void DynString::append(const DynString& tail) {

   char* updated = new char [this->lengthM + tail.lengthM+1];
   strcpy(updated, storageM);
   strcat(updated, tail.storageM);
   this->lengthM += tail.lengthM;
   delete[] this->storageM;
   storageM = updated;

}
```

ii) Output