**Course: ENSF 614** - Fall 2023  
**Lab #:** Lab 2  
**Instructor:** Mahmood Moussavi  
**Student Names:** Redge Santillan, Christian Valdez  
**Submission Date:** September 27, 2023

**Exercise A**

Point 1

A drawing of a diagram

Description automatically generated

Point 2

A screenshot of a phone

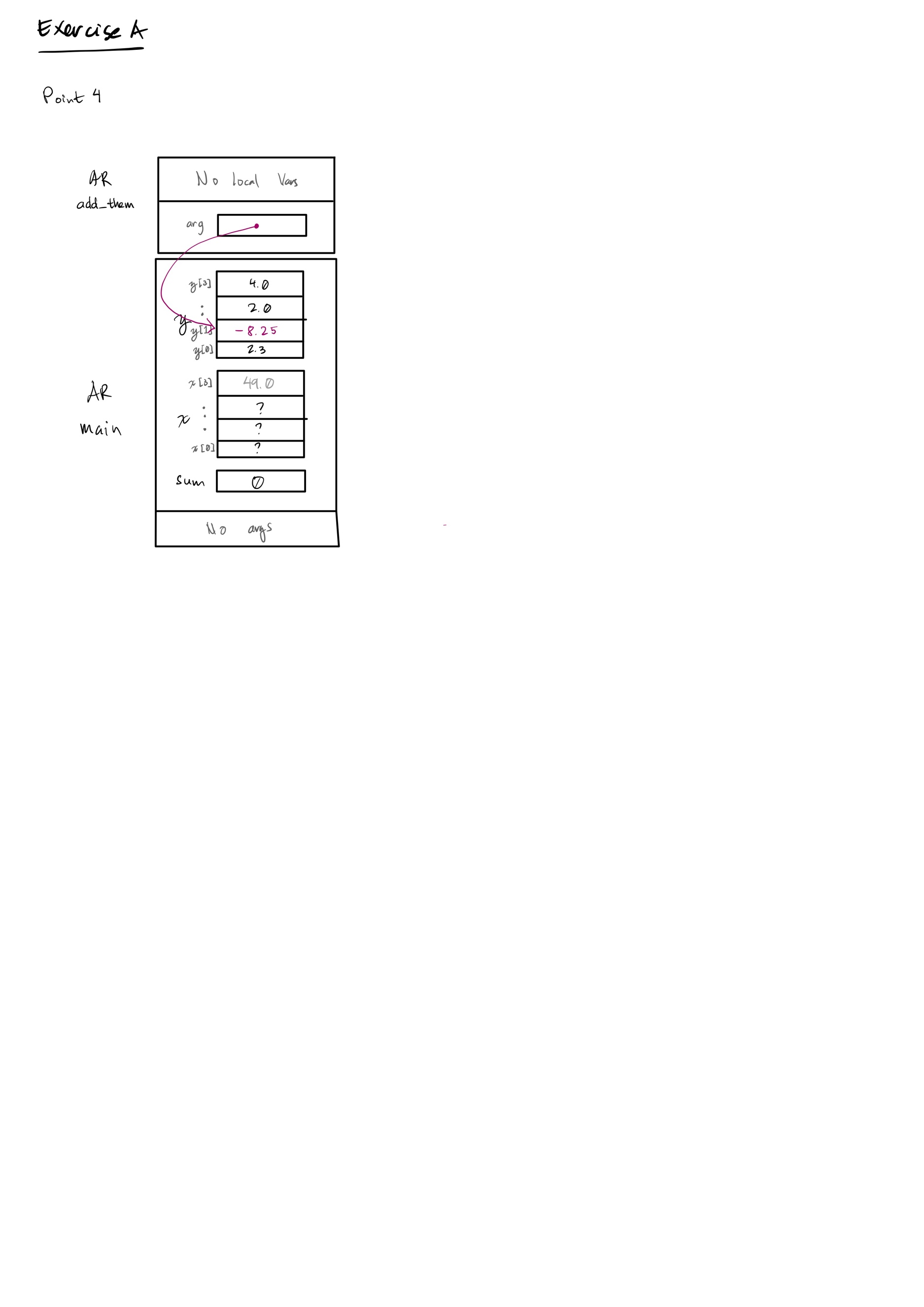
Description automatically generated

Point 3

A paper with writing on it

Description automatically generated

Point 4



**Exercise B**

/\*

\* Filename: my\_lab2exe\_B.cpp

\* Assignment: Lab 2 Exercise B

\* Section: B01

\* Completed by: Redge Santillan and Christian Valdez

\* Submission Date: Sep 27, 2023

\*/

**int** my\_strlen(**const** **char** \*s);

/\* Duplicates strlen from <cstring>, except return type is int.

\* REQUIRES

\* s points to the beginning of a string.

\* PROMISES

\* Returns the number of chars in the string, not including the

\* terminating null.

\*/

**void** my\_strncat(**char** \*dest, **const** **char** \*source, **int** n);

/\* Duplicates strncat from <cstring>, except return type is void.

\* REQUIRES

\* dest points to the beginning of a string

\* source points to the beginning of a string

\* n - integer, first n number of characters to copy from source to dest.

\* PROMISES

\* Appends the dest c-string with the first n characters of source c-string.

\*/

**int** my\_strcmp(**const** **char** \*str1, **const** **char** \*str2);

/\* Compares string1 and string2

\* REQUIRES

\* str1 points to the beginning of a string

\* str2 points to the beginning of a string

\* PROMISES

\* Returns 0 if str1 and str2 are identical.

\* Returns a positive integer if str1 > str2.

\* Returns a negative integer if str1 < str2.

\*/

#include <iostream>

#include <cstring>

**using** **namespace** std;

**int** main(**void**)

{

**char** str1[7] = "banana";

**const** **char** str2[] = "-tacit";

**const** **char**\* str3 = "-toe";

/\* point 1 \*/

**char** str5[] = "ticket";

**char** my\_string[100]="";

**int** bytes;

**int** length;

/\* using strlen libarary function \*/

length = (**int**) my\_strlen(my\_string);

cout << "\nLine 1: my\_string length is " << length;

/\* using sizeof operator \*/

bytes = **sizeof** (my\_string);

cout << "\nLine 2: my\_string size is " << bytes << " bytes.";

/\* using strcpy libarary function \*/

strcpy(my\_string, str1);

cout << "\nLine 3: my\_string contains: " << my\_string;

length = (**int**) my\_strlen(my\_string);

cout << "\nLine 4: my\_string length is " << length << ".";

my\_string[0] = '\0';

cout << "\nLine 5: my\_string contains:\"" << my\_string << "\"";

length = (**int**) my\_strlen(my\_string);

cout << "\nLine 6: my\_string length is " << length << ".";

bytes = **sizeof** (my\_string);

cout << "\nLine 7: my\_string size is still " << bytes << " bytes.";

/\* strncat append the first 3 characters of str5 to the end of my\_string \*/

my\_strncat(my\_string, str5, 3);

cout << "\nLine 8: my\_string contains:\"" << my\_string << "\"";

length = (**int**) my\_strlen(my\_string);

cout << "\nLine 9: my\_string length is " << length << ".";

my\_strncat(my\_string, str2, 4);

cout << "\nLine 10: my\_string contains:\"" << my\_string << "\"";

/\* strncat append ONLY up ot '\0' character from str3 -- not 6 characters \*/

my\_strncat(my\_string, str3, 6);

cout << "\nLine 11: my\_string contains:\"" << my\_string << "\"";

length = (**int**) my\_strlen(my\_string);

cout << "\nLine 12; my\_string has " << length << " characters.";

cout << "\n\nUsing strcmp - C library function: ";

cout << "\n\"ABCD\" is less than \"ABCDE\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABCDE");

cout << "\n\"ABCD\" is less than \"ABND\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABND");

cout << "\n\"ABCD\" is equal than \"ABCD\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABCD");

cout << "\n\"ABCD\" is less than \"ABCd\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABCd");

cout << "\n\"Orange\" is greater than \"Apple\" ... strcmp returns: " <<

my\_strcmp("Orange", "Apple") << endl;

**return** 0;

}

/\* Duplicates strlen from <cstring>, except return type is int.

\* Counts the number of non-'\0' characters in a char array.

\* Returns the number of non-'\0' characters in a char array.

\*/

**int** my\_strlen(**const** **char** \*s){

**bool** endOfArray = **false**;

**int** counter = 0;

**while** (!endOfArray) {

**if** (s[counter] == '\0'){

endOfArray = **true**;

} **else** {

counter++;

}

}

**return** counter;

}

/\* Appends the first n characters of string source to string dest, and returns a char\* to dest. If the length of the C-string in source is less than n, only the content up to the terminating null character '\0' is copied.

\*/

**void** my\_strncat(**char** \*dest, **const** **char** \*source, **int** n){

// If given n > strlen(source), only copy strlen(source)

**int** sourceLength = my\_strlen(source);

**int** destLength = my\_strlen(dest);

**if** (n > sourceLength){

n = sourceLength;

}

// Look for the first '\0' in dest - this will be n + 1. Loop thru

**for** (**int** i = 0; i < n; i++){

dest[i + destLength] = source[i];

}

dest[n + destLength] = '\0';

}

/\*\* Compares 2 c-strings.

Returns 0 if str1 and str2 are identical.

Returns a positive integer if str1 > str2.

Returns a negative integer if str1 < str2.

\*\**/*

**int** my\_strcmp(**const** **char** \*str1, **const** **char** \*str2){

// as soon as you find the difference until you subtract - don't need the lengths.

**int** result = 0;

// The while condition ensures that as soon as str1 and str2 are pointing to values that are NOT the same, the program will exit the loop

// Check if str1 is pointing to a '\0' value to ensure that neither pointers will point to inaccessible memory

**while** ((\*str1 == \*str2) && \*str1 != '\0') {

str1++;

str2++;

}

result = \*str1 - \*str2;

**return** result;

}

**Sample output:**

**A computer screen shot of a computer program

Description automatically generated**

**Exercise C**

Point 1 – second function call.

A diagram of a diagram

Description automatically generated

**Exercise E**

/\*

\* Filename: lab2exe\_E.cpp

\* Implementation file for complex number module

\* Assignment: Lab 2 Exercise E

\* Section: B01

\* Completed by: Christian Valdez and Redge Santillan

\* Submission date: Sep 27, 2023

\*/

#include "lab2exe\_E.h"

cplx cplx\_add(cplx z1, cplx z2) {

cplx result;

result.real = z1.real + z2.real;

result.imag = z1.imag + z2.imag;

return result;

}

void cplx\_subtract(cplx z1, cplx z2, cplx\* difference) {

difference->real = z1.real - z2.real;

difference->imag = z1.imag - z2.imag;

}

void cplx\_multiply(const cplx\* z1, const cplx\* z2, cplx\* difference) {

difference->real = (z1->real \* z2->real) - (z1->imag \* z2->imag);

difference->imag = (z1->real \* z2->imag) + (z1->imag \* z2->real);

}