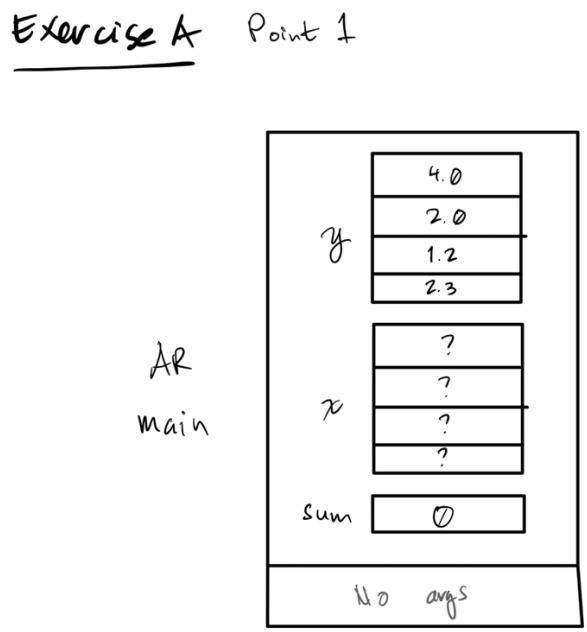
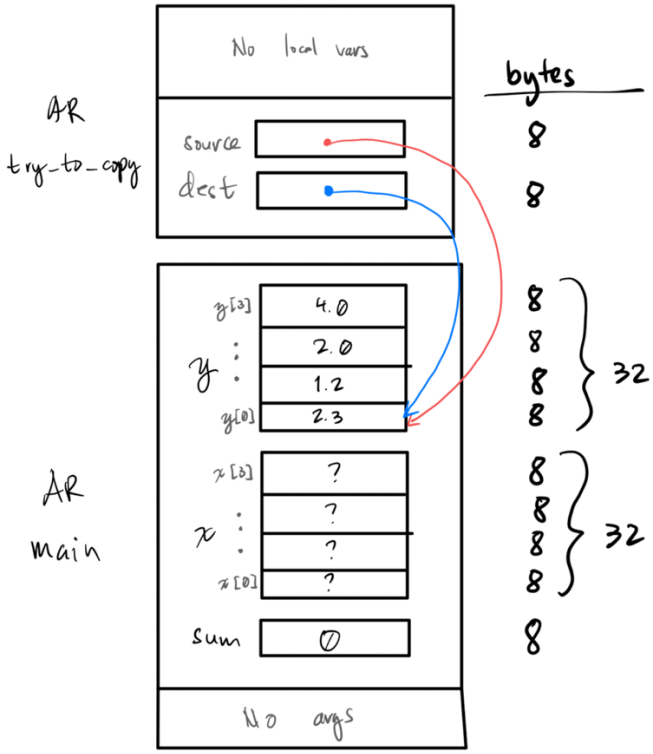
Course: ENSF 614 - Fall 2023
Lab #: Lab 2
Instructor: Mahmood Moussavi
Student Names: Redge Santillan,
Christian Valdez
Submission Date: September 27, 2023

Exercise A

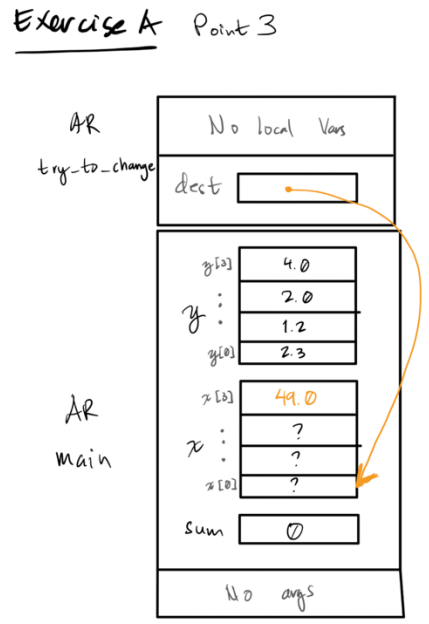
Point 1



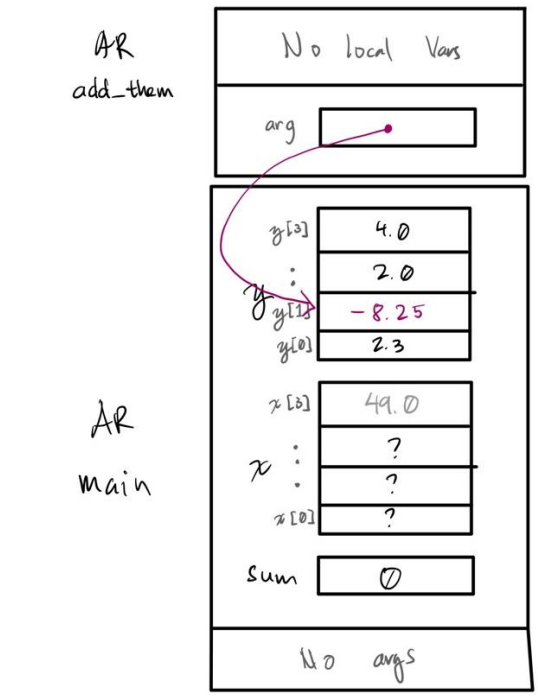
Point 2



Point 3



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;
}
```

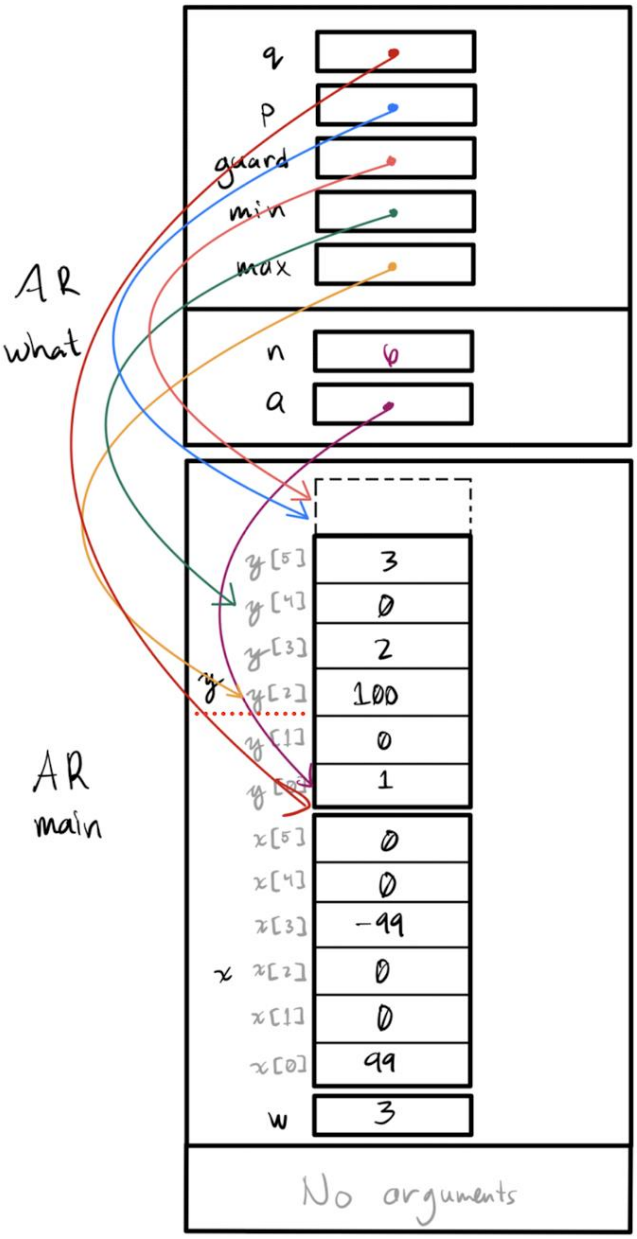
Sample output:

```
Line 1: my_string length is 0
Line 2: my_string size is 100 bytes.
Line 3: my_string contains: banana
Line 4: my_string length is 6.
Line 5: my_string contains:""
Line 6: my_string length is 0.
Line 7: my_string size is still 100 bytes.
Line 8: my_string contains:"tic"
Line 9: my_string length is 3.
Line 10: my_string contains:"tic-tac"
Line 11: my_string contains:"tic-tac-toe"
Line 12; my_string has 11 characters.

Using strcmp - C library function:
"ABCD" is less than "ABCDE" ... strcmp returns: -69
"ABCD" is less than "ABND" ... strcmp returns: -11
"ABCD" is equal than "ABCD" ... strcmp returns: 0
"ABCD" is less than "ABCd" ... strcmp returns: -32
"Orange" is greater than "Apple" ... strcmp returns: 14
```

Exercise C

Point 1 – second function call.



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);
}
```