Course: ENSF 614 - Fall 2023

Lab B01: Lab 2

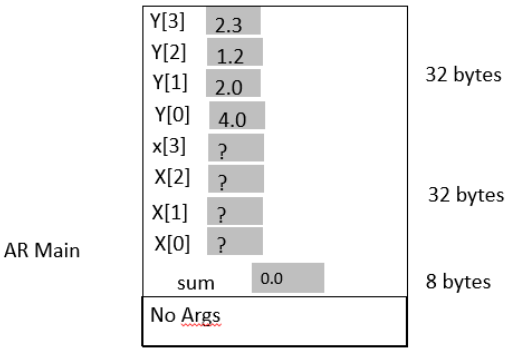
Instructor: Mahmood Moussavi

Student Name: Braden Tink

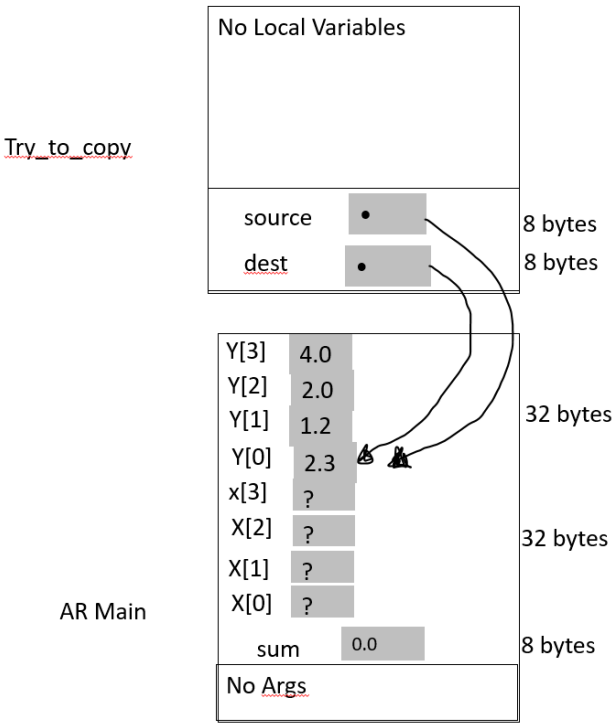
Submission Date: September , 2023

Exercise A

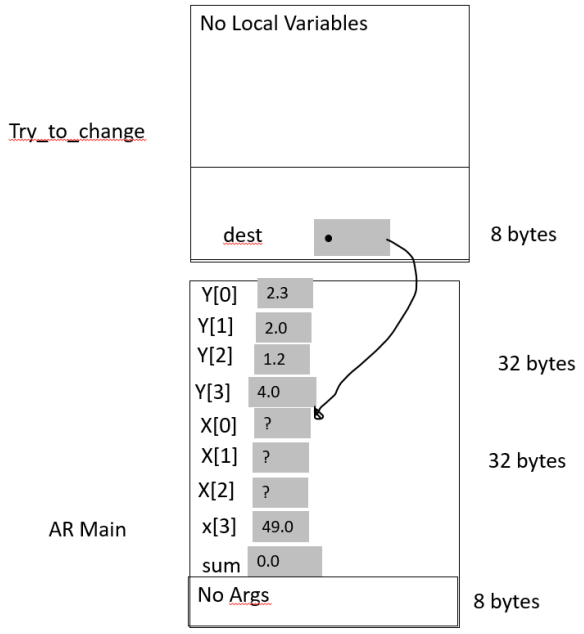
Point 1



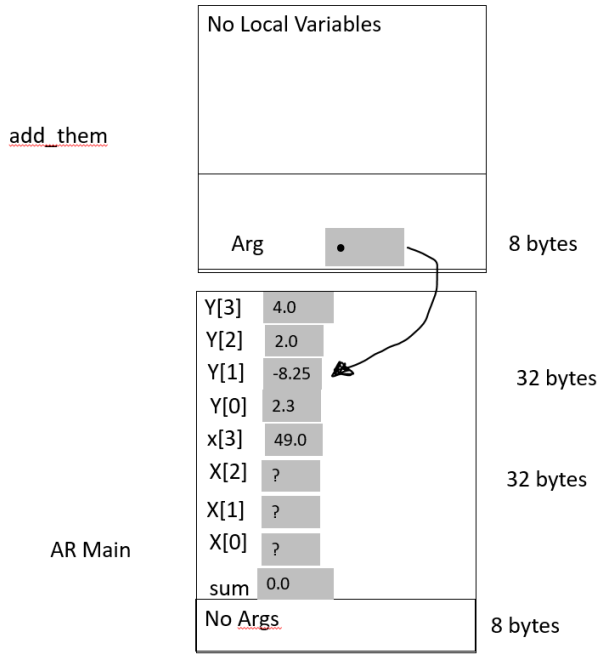
Point 2



Point 3



Point 4



Exercise B

```
/*
 *  lab2exe_B.cpp
 *  ENSF 614 Lab 2 Exercise B
 */

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 pos);
/*  Duplicates strncat from <cstring>, except return type is void.
 */

#include <iostream>
#include <cstring>
using namespace std;

int my_strlen(const char *s);
void my_strncat(char *dest, const char *source, int);
int  my_strcmp(const char *str1,const char *str2);

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;
```

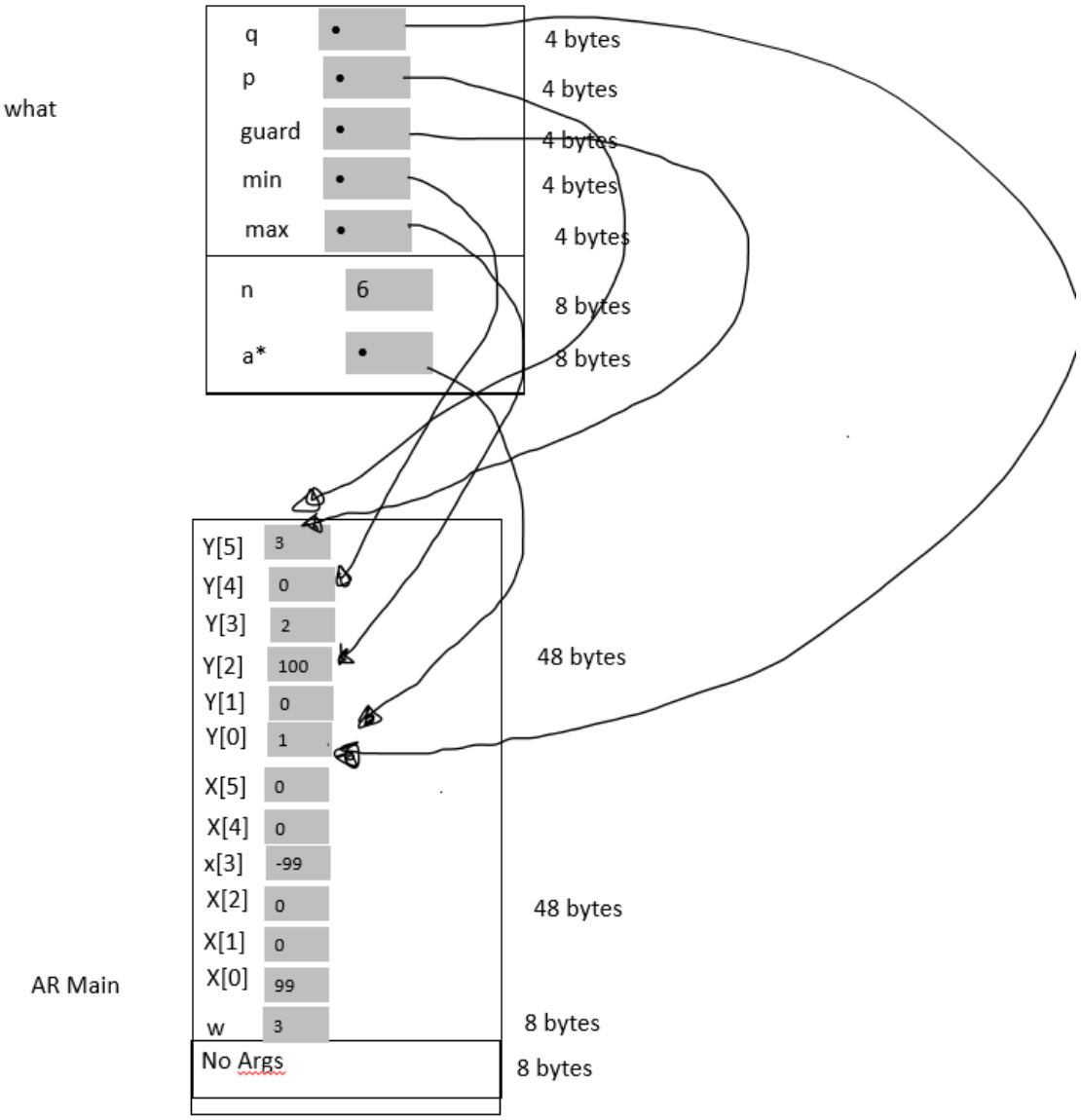
```
    length = (int) my_strlen(my_string);
    cout << "\nLine 4: my_string length is " << length << ".";

    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;
}

int my_strlen(const char *s){
        int count = 0;
```

```
	while(true){
		if (s[count] == '\0'){
			break;
		}
		count += 1;
	}
	return count;

}
void my_strncat(char *dest, const char *source, int pos){
	int count = 0;

	while(true){
		if (dest[count] == '\0'){

			break;
		}
		count += 1;
	}


	for(int i = 0; i < pos; i++){
		dest[count] = source[i];
		count += 1;
	}

	dest[count] = '\0';
}

int my_strcmp(const char *str1,const char *str2){
	int int1 = 0;
	int int2 = 0;
	int total = 0;
	int i = 0;
	while(true){

		if((str1[i]) == '\0' && (str2[i]) == '\0'){

			break;
		}
		else if((str1[i]) == (str2[i])){

			i += 1;
		}
		else {

			break;
		}

	}
	int1 = int(str1[i]);
```

```
        int2 = int(str2[i]);
        total = int1 - int2;

        return total;
}
```

```
Braden@TBLaptop04 /cygdrive/c/users/braden/documents/school/ENSF 614/Assignments
/Assignment 2
$ g++ -Wall my_lab2exe_B.cpp

Braden@TBLaptop04 /cygdrive/c/users/braden/documents/school/ENSF 614/Assignments/Assignment 2
$ ./a.exe

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

Braden@TBLaptop04 /cygdrive/c/users/braden/documents/school/ENSF 614/Assignments/Assignment 2
$
```

Exercise C


what
q
p
guard
min
max
4 bytes
4 bytes
4 bytes
4 bytes
4 bytes
n
6
8 bytes
a*
8 bytes
Y[5] 3
Y[4] 0
Y[3] 2
Y[2] 100
Y[1] 0
Y[0] 1
48 bytes
X[5] 0
X[4] 0
x[3] -99
X[2] 0
X[1] 0
X[0] 99
48 bytes
AR Main
w 3
8 bytes
No Args
8 bytes

Exercise D (Omitted)

Exercise E

Add

```
cplx cplx_add(cplx z1, cplx z2)
{
  cplx result;

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

  return result;
}
```

Subtract

```
void cplx_subtract(cplx z1, cplx z2,  cplx *difference)
{

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

}
```

Multiply

```
void cplx_multiply(const cplx *pz1, const cplx *pz2, cplx *product)
{

        product -> real = (pz1 -> real * pz2 ->real) - (pz1->imag * pz2->imag);
        product -> imag = (pz1 -> real * pz2 ->imag) + (pz1->imag * pz2->real);

}
```