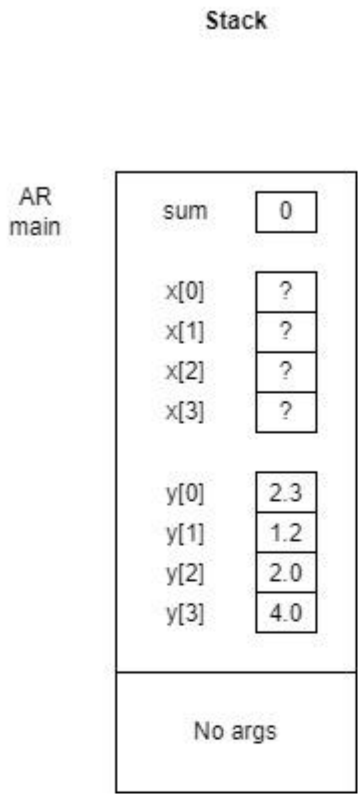
Course: ENSF 614 – Fall 2022

Assignment 2

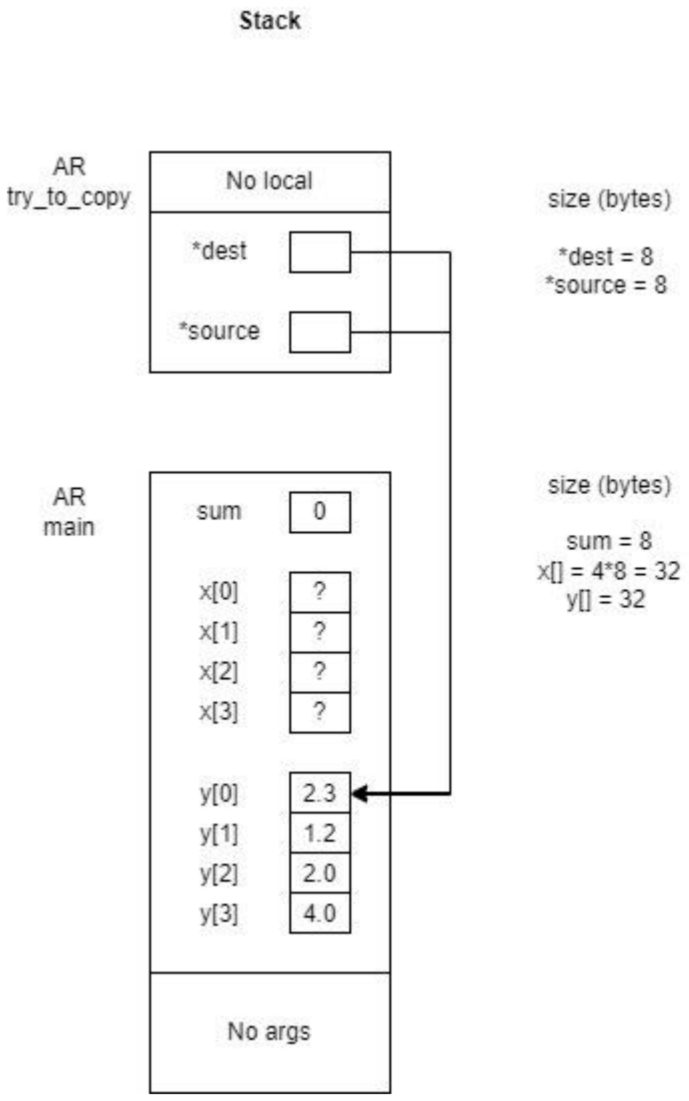
Student Name: Khoi Nguyen

Exercise A

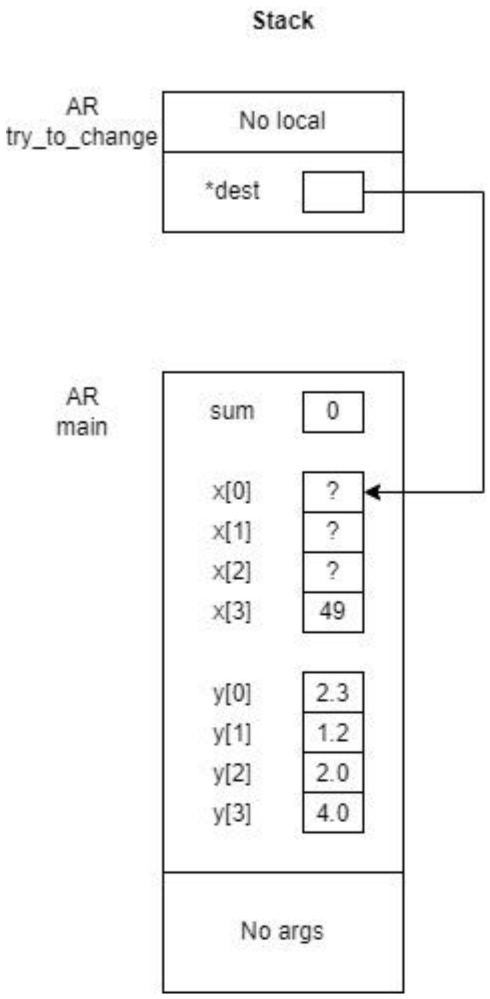
Point 1



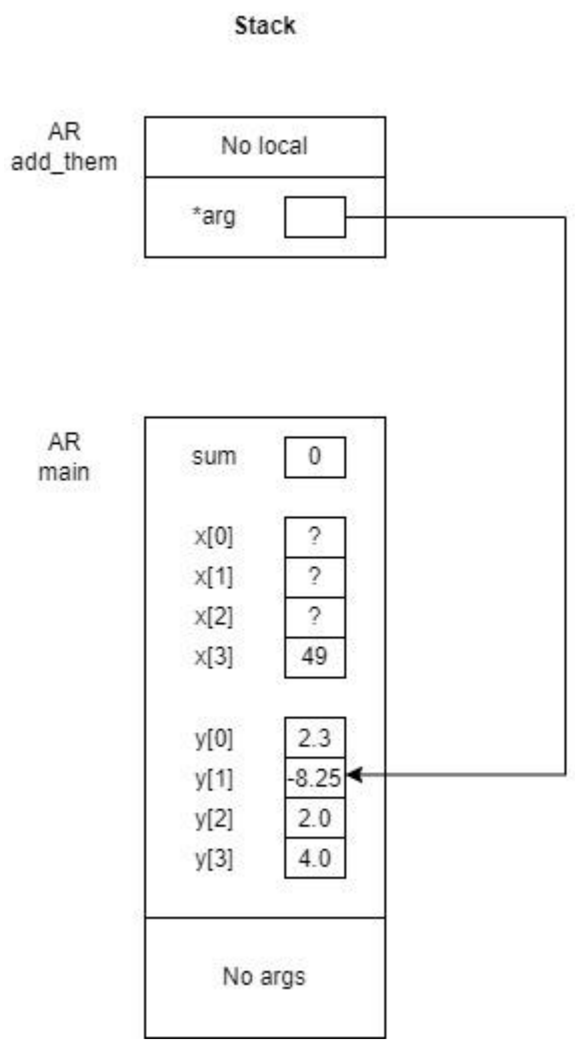
Point 2


Stack
AR
try_to_copy
No local
*dest
*source
size (bytes)
*dest = 8
*source = 8
AR
main
sum
0
x[0]
x[1]
x[2]
x[3]
?
?
?
?
y[0]
y[1]
y[2]
y[3]
2.3
1.2
2.0
4.0
size (bytes)
sum = 8
x[] = 4*8 = 32
y[] = 32
No args

Point 3


Stack
AR
try_to_change
No local
*dest
AR
main
sum
0
x[0]
?
x[1]
?
x[2]
?
x[3]
49
y[0]
2.3
y[1]
1.2
y[2]
2.0
y[3]
4.0
No args

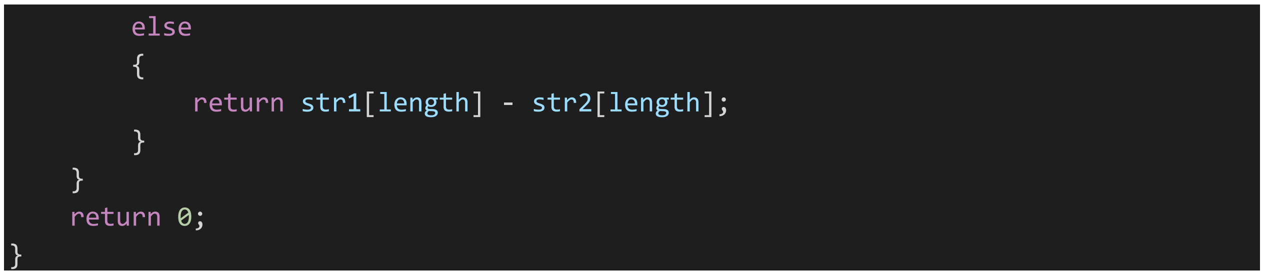
Point 4



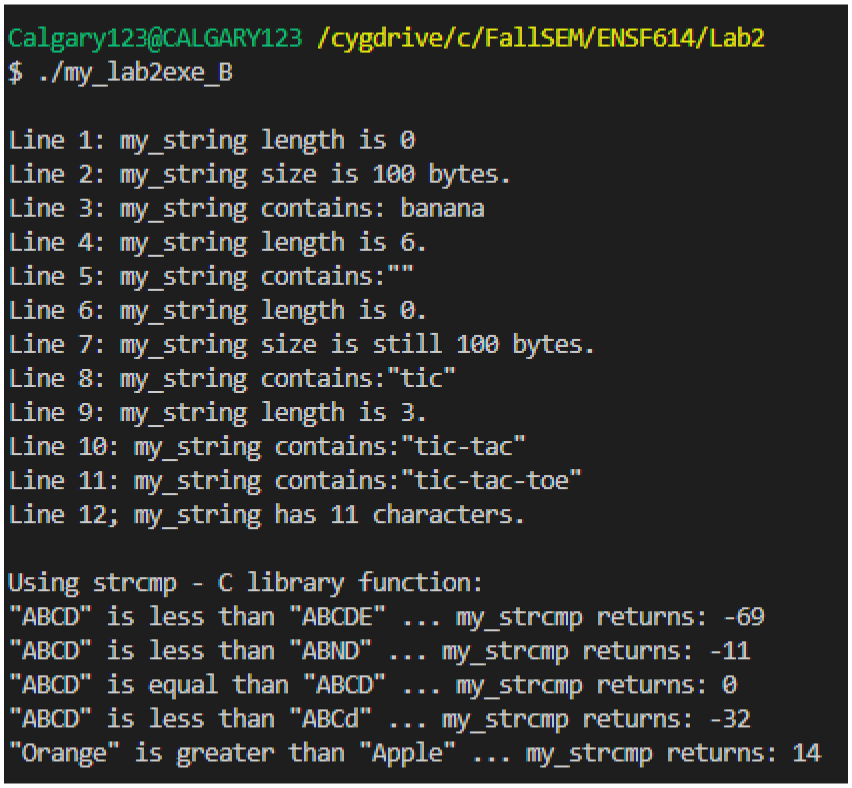
Exercise B

```
/*
 *  my_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);
/*  Duplicates strncat from <cstring>, except return type is void.
 *  REQUIRES
 *     dest points to the beginning of the first string.
 *     source points to the beginning of the second string.
 *     int indicates how many letters from the second string needs to be added to
string 1
 *  PROMISES
 *     Returns the concatenated string.
 */

int my_strcmp(const char *str1, const char *str2);
/*  Duplicates strcmp from <cstring>, except return type is void.
 *  REQUIRES
 *     str1 points to the beginning of the first string.
 *     str2 points to the beginning of the second string.
 *  PROMISES
 *     Returns an integer to indicate the difference between str1 and str2 if
they are different.
 *     0 indicates both strings are the same.
 *     Positive means str1 is larger than str2 and vice versa.
 */

#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 << "\"";

    /* my_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\" ... my_strcmp returns: " <<
    my_strcmp("ABCD", "ABCDE");

    cout << "\n\"ABCD\" is less than \"ABND\" ... my_strcmp returns: " <<
    my_strcmp("ABCD", "ABND");

    cout << "\n\"ABCD\" is equal than \"ABCD\" ... my_strcmp returns: " <<
    my_strcmp("ABCD", "ABCD");

    cout << "\n\"ABCD\" is less than \"ABCd\" ... my_strcmp returns: " <<
    my_strcmp("ABCD", "ABCd");

    cout << "\n\"Orange\" is greater than \"Apple\" ... my_strcmp returns: " <<
    my_strcmp("Orange", "Apple") << endl;
    return 0;
}

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

    while (s[length] != 0)
    {
        length++;
    }
    return length;
}

void my_strncat(char *dest, const char *source, int n)
{
    int end = 0;
    while (dest[end] != 0)
    {
        end++;
    }
    for (int i = 0; i < n; i++)
    {
        dest[end] = source[i];
        end++;
    }
    dest[end] = 0;
}

int my_strcmp(const char *str1, const char *str2)
{
    int length = 0;
    while (str1[length] != 0 || str2[length] != 0)
    {
        if (str1[length] == str2 [length])
        {
            length++;
        }
```

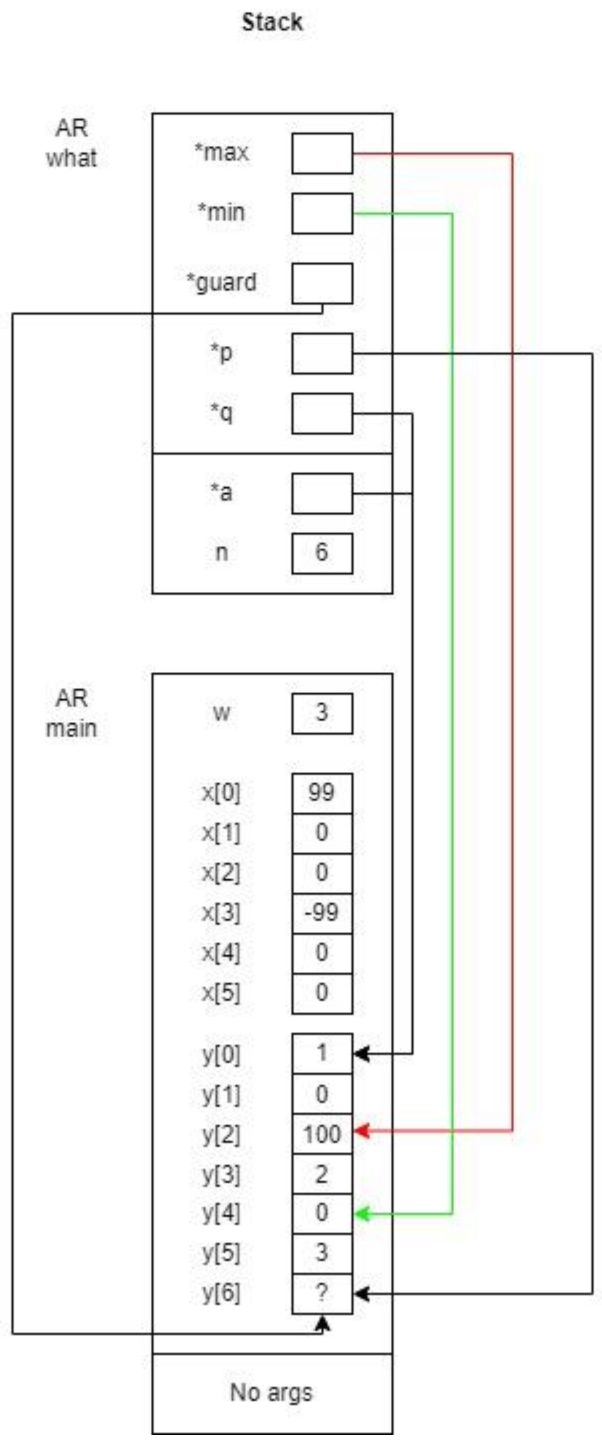


```c
        else
        {
            return str1[length] - str2[length];
        }
    }
    return 0;
}
```



```
Calgary123@CALGARY123 /cygdrive/c/FallSEM/ENSF614/Lab2
$ ./my_lab2exe_B

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" ... my_strcmp returns: -69
"ABCD" is less than "ABND" ... my_strcmp returns: -11
"ABCD" is equal than "ABCD" ... my_strcmp returns: 0
"ABCD" is less than "ABCd" ... my_strcmp returns: -32
"Orange" is greater than "Apple" ... my_strcmp returns: 14
```

Exercise C



Exercise E

```
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 *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;
}
```