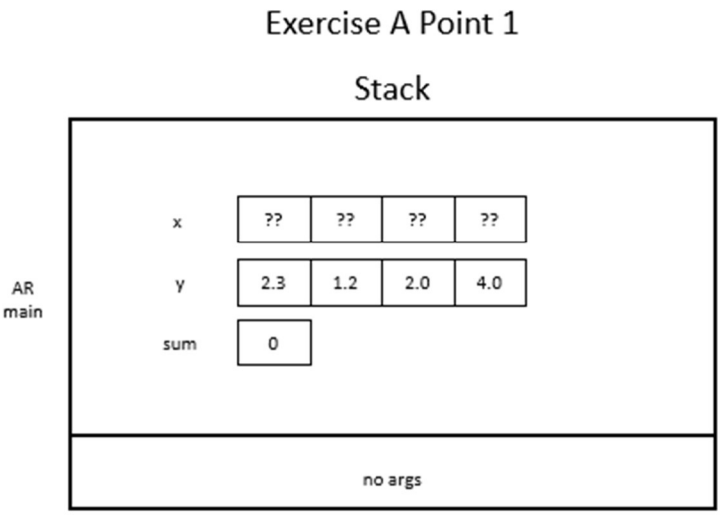
Course: ENSF 614–Fall2021

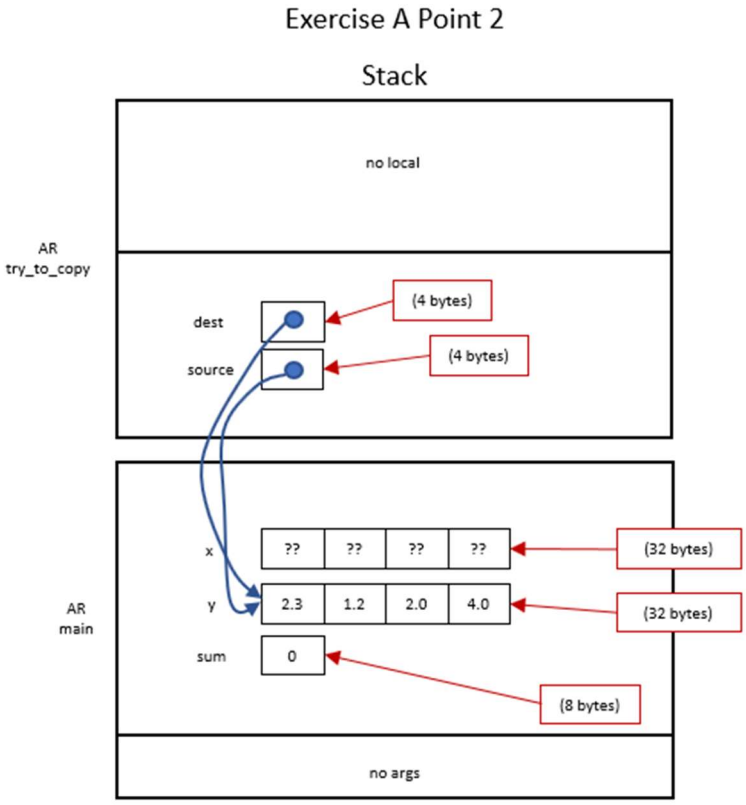
Lab #: Lab 2

Student Names: Graydon Hall, Jared Kraus

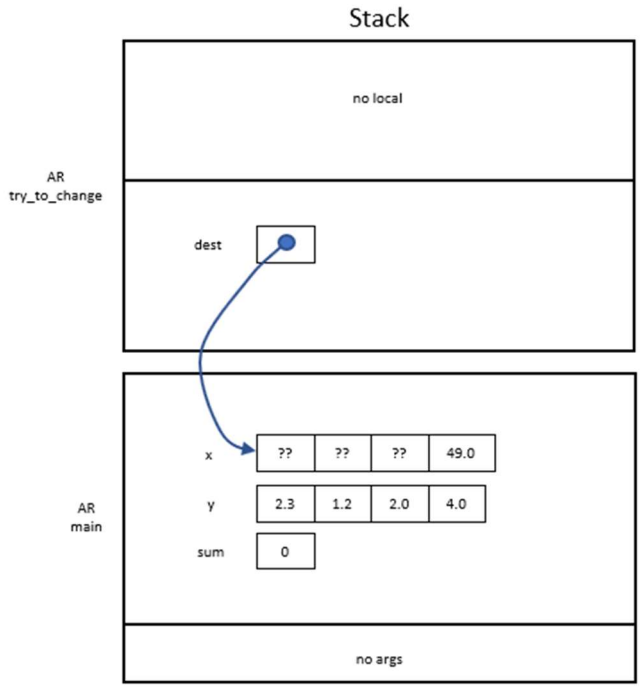
Submission Date: 2021-09-27

Exercise A

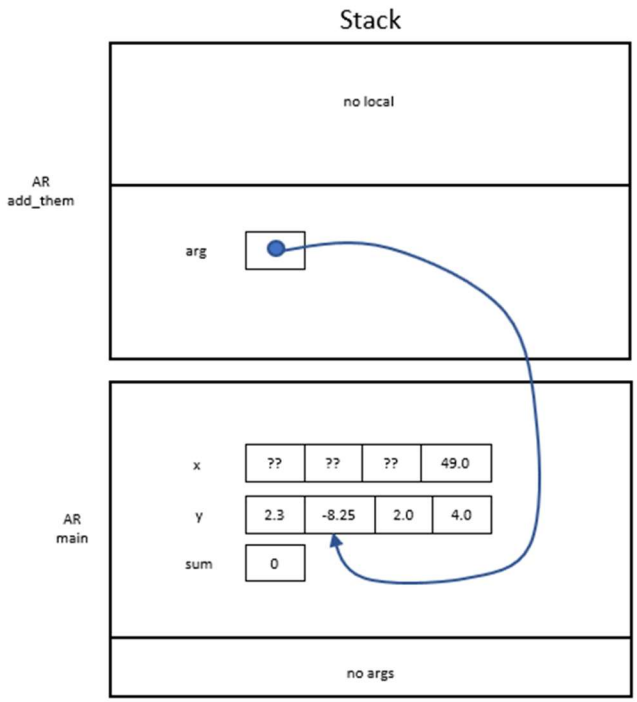

Exercise A Point 1
Stack
AR
main
x
??
??
??
??
y
2.3
1.2
2.0
4.0
sum
0
no args


Exercise A Point 2
Stack
no local
AR
try_to_copy
dest
source
(4 bytes)
(4 bytes)
AR
main
x
??
??
??
??
(32 bytes)
y
2.3
1.2
2.0
4.0
(32 bytes)
sum
0
(8 bytes)
no args

Exercise A Point 3



Exercise A Point 4



Exercise B

```c
/* File Name: my_lab2exe_B.c
* Lab # and Assignment #: Lab #2 Exercise B
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-09-27
*/

int my_strlen(const char *s);
/*  Duplicates my_strlen from <string.h>, 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 my_strncat from <string.h>, except return type is void.
 *  REQUIRES
 *     source and dest point to the beginning of the strings.
 *  PROMISES
 *     Appends sttring pointed to by source, to the string pointed to by dest
 *     up to n characters from source.
 */


int my_strcmp(const char* str1, const char* str2);
/*  Duplicates my_strcmp from <string.h>, except return type is void.
 *  REQUIRES
 *     str1 and str2 point to the beginning of the strings.
 *  PROMISES
 *     to return 0 when str1 and str2 are identical. Otherwise, returns the ASCII
 *     value differences of the first two characters that are different.
 *
 */

#include <stdio.h>
#include <string.h>

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;
```

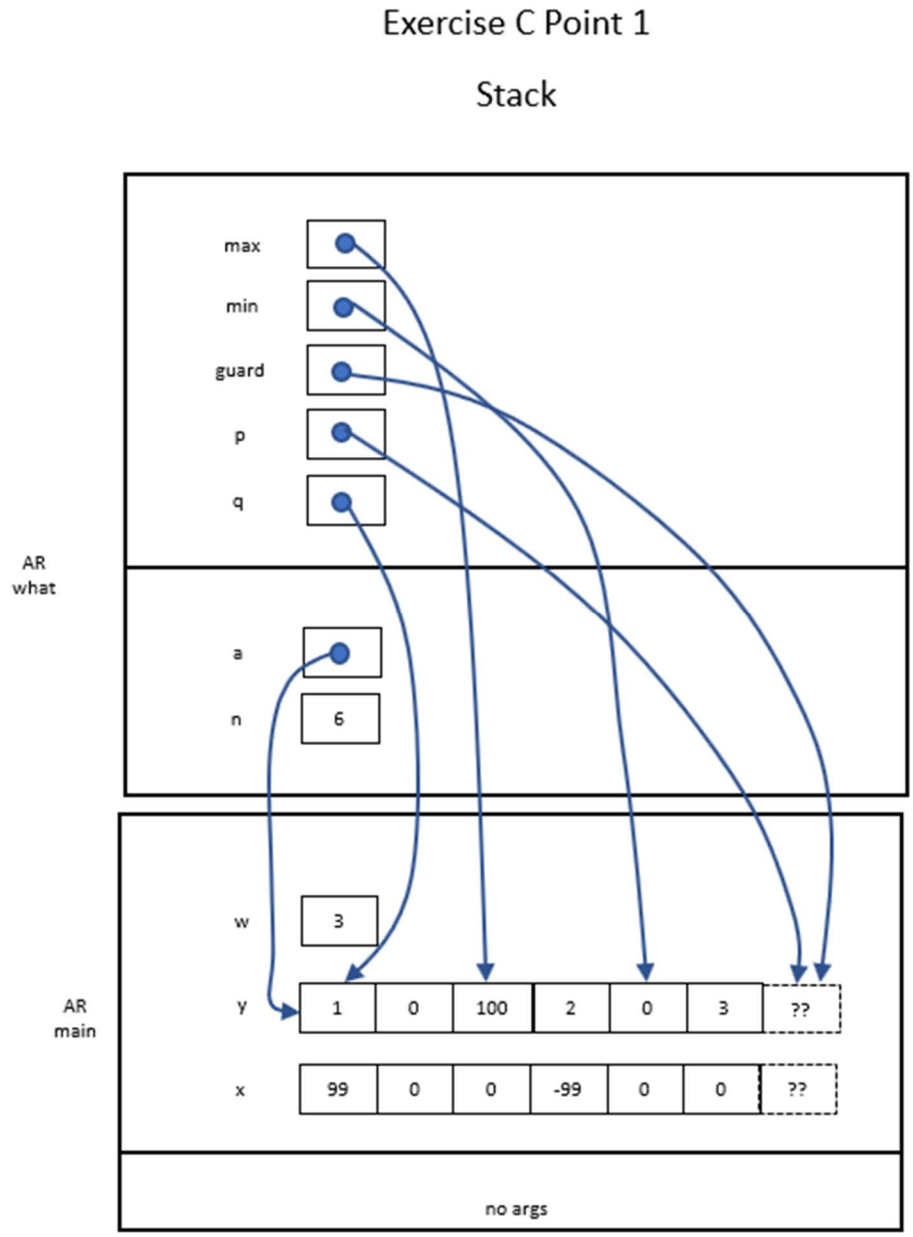
```
    int length;

    /* using my_strlen C libarary function */
    length = (int) my_strlen(my_string);
    printf("\nLine 1: my_string length is %d.", length);

    /* using sizeof operator */
    bytes = sizeof (my_string);
    printf("\nLine 2: my_string size is %d bytes.", bytes);

    /* using strcpy C libarary function */
    strcpy(my_string, str1);
    printf("\nLine 3: my_string contains: %s", my_string);

    length = (int) my_strlen(my_string);
    printf("\nLine 4: my_string length is %d.", length);

    my_string[0] = '\0';
    printf("\nLine 5: my_string contains:\"%s\"", my_string);

    length = (int) my_strlen(my_string);
    printf("\nLine 6: my_string length is %d.", length);

    bytes = sizeof (my_string);
    printf("\nLine 7: my_string size is still %d bytes.", bytes);

    /* my_strncat append the first 3 characters of str5 to the end of my_string */
    my_strncat(my_string, str5, 3);
    printf("\nLine 8: my_string contains:\"%s\"", my_string);

    length = (int) my_strlen(my_string);
    printf("\nLine 9: my_string length is %d.", length);

    my_strncat(my_string, str2,  4);
    printf("\nLine 10: my_string contains:\"%s\"", my_string);

    /* my_strncat append ONLY up ot '\0' character from str3 -- not 6 characters */
    my_strncat(my_string, str3, 6);
    printf("\nLine 11: my_string contains:\"%s\"", my_string);

    length = (int) my_strlen(my_string);
    printf("\nLine 12; my_string has %d characters.", length);

    printf("\n\nUsing my_strcmp - C library function: ");

    printf("\n\"ABCD\" is less than \"ABCDE\" ... my_strcmp returns: %d",
           my_strcmp("ABCD", "ABCDE"));
    printf("\n\"ABCD\" is less than \"ABND\" ... my_strcmp returns: %d",
           my_strcmp("ABCD", "ABND"));
    printf("\n\"ABCD\" is equal than \"ABCD\" ... my_strcmp returns: %d",
```

```
           my_strcmp("ABCD", "ABCD"));
    printf("\n\"ABCD\" is less than \"ABCd\" ... my_strcmp returns: %d",
           my_strcmp("ABCD", "ABCd"));
    printf("\n\"Orange\" is greater than \"Apple\" ... my_strcmp returns: %d\n",
           my_strcmp("Orange", "Apple"));

    return 0;
}

int my_strcmp(const char* str1, const char* str2){
       int i = 0;
       while (1) {       /* Stop looping when we reach the null-character. */

              if((str1[i] == '\0') && (str2[i] == '\0')){  // got to end of both strings an
d they were always equal
                     return 0;
              }
              if(str1[i] != str2[i]){
                     return (int) (str1[i] - str2[i]);
              }
              i++;

       }
}

int my_strlen(const char *s){
       int i = 0;
       while (s[i] != '\0') {       /* Stop looping when we reach the null-character. */
              i++;
       }
       return i;
}

void my_strncat(char *dest, const char *source, int n){
       int i = 0;
       int j = 0;
       while (dest[i] != '\0') {       /* Stop looping when we reach the null-character. */
              i++;
       }
       while ((source[j] != '\0' && (j<n))) {       /* Stop looping when we reach the null-
character. */
              dest[i] = source[j];
              i++;
              j++;
       }
       dest[i] = '\0';

}
```

Program 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 my_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
PS C:\Users\grayd\OneDrive\Documents\School\MEng\Semester 3\ENSF 614\Labs\Lab 2>
```

Exercise C


Exercise C Point 1
Stack
max
min
guard
p
q
AR
what
a
n
6
w
3
AR
main
y
1
0
100
2
0
3
??
x
99
0
0
-99
0
0
??
no args

Exercise E:

```c
/* File Name: my_lab2exe_E.c
* Lab # and Assignment #: Lab #2 Exercise E
* Lab section: 1
* Completed by: Graydon Hall and Jared Kraus
* Submission Date: 2021-09-27
*/

#include "lab2exe_E.h"

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

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

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

  (*difference).real = z1.real - z2.real;
  (*difference).imag = z1.imag - z2.imag;

}

void
cplx_multiply(const struct cplx *pz1, const struct cplx *pz2, struct cplx *product){
  (*product).real = (*pz1).real*(*pz2).real - (*pz1).imag*(*pz2).imag;
  (*product).imag = (*pz1).real*(*pz2).imag + (*pz1).imag*(*pz2).real;
}
```