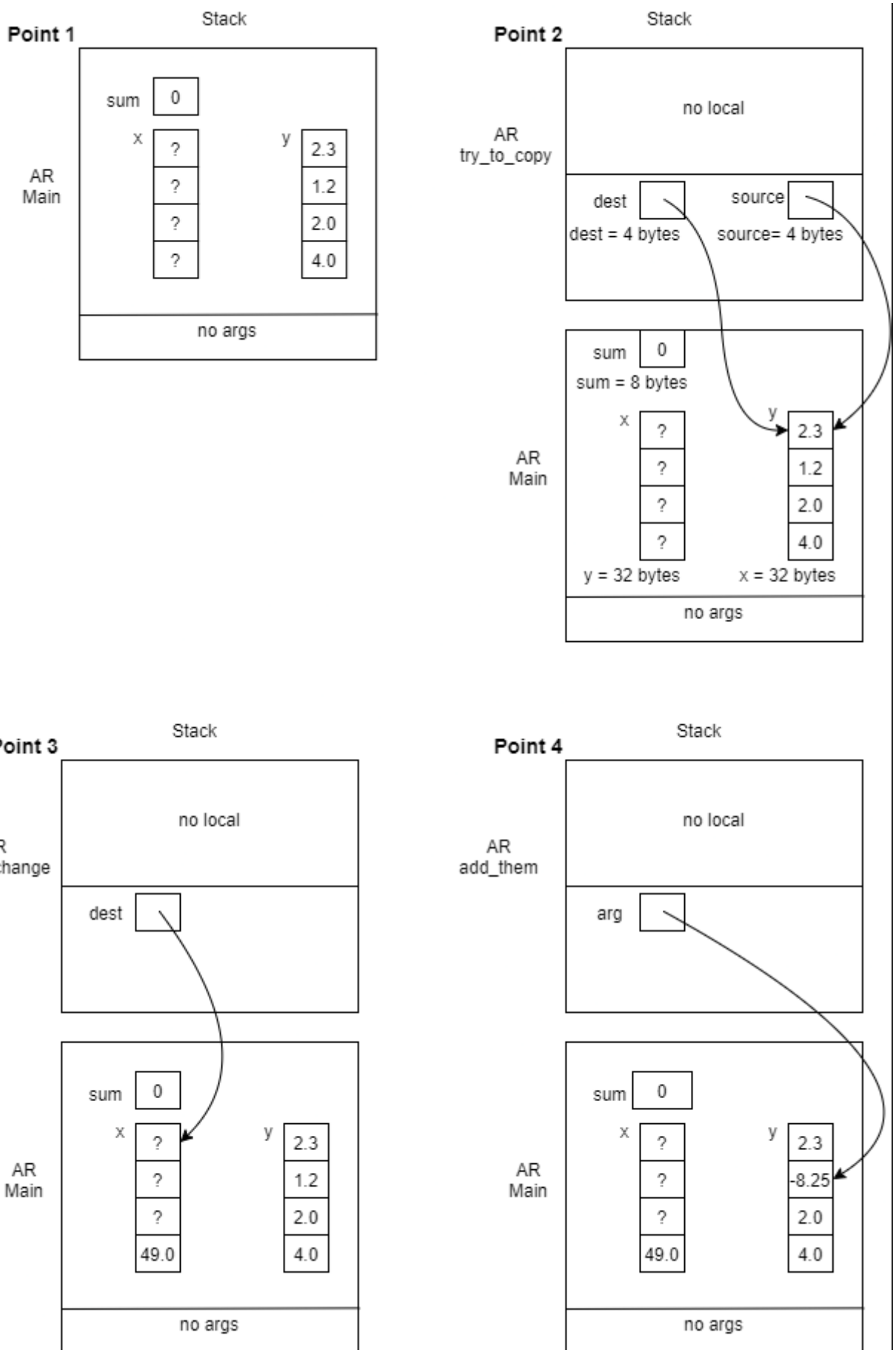
Lab 2 Report

Course: ENSF 619 - Fall 2020
Lab #: Lab 2
Student Name: Davis Allan, 10016543
Submission Date: Oct 1 2020

Exercise A

AR Diagram



Exercise B

Source Code

```
/*
* File Name: my_lab2exe_B.c
* Lab # and Assignment #: Lab 2 Exercise B
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 1 2020
*/
int my_strcmp(const char* str1, const char* str2);
/*  Duplicates strcmp from <string.h>, except it returns the ASCII value
difference of
 *  the first two characters that are different.
 *  REQUIRES
 *      str1 points to the beginning of a string.
 *      str2 points to the beginning of a string.
 *  PROMISES
 *      Returns 0 if the strings are identical, otherwise, returns the
ASCII value
 *      difference of the first two characters that are different
 */

int my_strlen(const char *s);
/*  Duplicates 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 strncat from <string.h>, except return type is void.
 *
 *  REQUIRES
 *      dest points to the beginning of a string
 *      source points to the beginning of a string
 *      n is the upper bound of how many characters should be copied
 *
 *  PROMISES
 *      Concatenates the number of characters from source to the end of
```

```
the
 *      destination string
 */

#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);

    /* 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);

    /* 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 - replicates 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_strlen(const char *s) {

    int length = 0;
    int i = 0;
    while (s[i] != '\0')
    {
        length += 1;
        i++;
```

```
    }
    return i;
}

void my_strncat(char *dest, const char *source, int n) {
    int i = 0;
    //loop to find index of null character in destination string
    while (dest[i] != '\0')
    {
        i++;
    }
    //copying source string to destination string
    for (int j = 0; j < n; j++) {
        if (source[j] == '\0') {
            dest[i] = source[j];
            break;
        }
        dest[i] = source[j];
        i++;
    }
    //adding in final null character to the end of the new string
    dest[i] = '\0';
}

int my_strcmp(const char* str1, const char* str2) {
    int i = 0;
    while (str1[i] == str2[i]) {
        if (str1[i] == '\0' || str2[i] == '\0') {
            break;
        }
        i++;
    }
    return str1[i] - str2[i];
}
```

Program Output

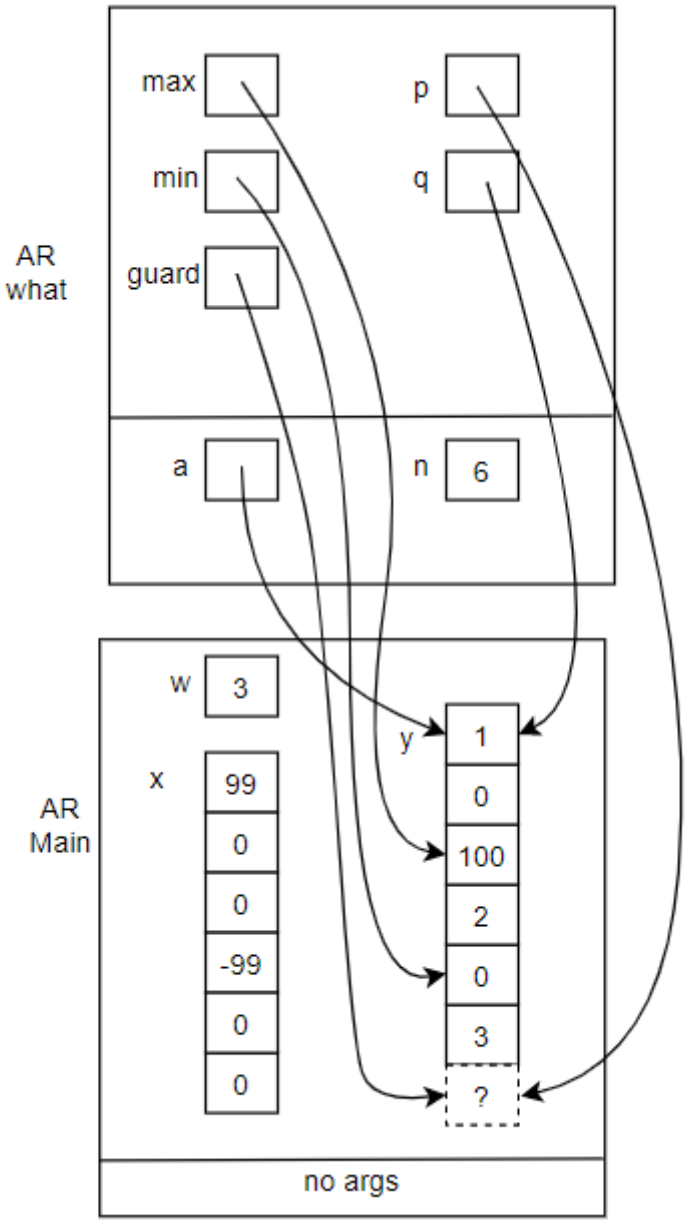
```
PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab2> cd "c:\Users\davis\Desktop\ENSF 619\Labs\Lab2\"

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 - replicates 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\davis\Desktop\ENSF 619\Labs\Lab2>
```

Exercise C

AR Diagram



Exercise E

Source Code

```c
/*
* File Name: lab2exe_E.c
* Lab # and Assignment #: Lab 2 Exercise E
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 1 2020
*/

#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);
}
```

Exercise F

Source Code

```c
/*
* File Name: lab2exe_F.c
* Lab # and Assignment #: Lab 2 Exercise F
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: Oct 1 2020
*/

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

const int ARRAY_SIZE = 10;

/* a structure that represents a point on a Cartesian coordinates system.
*/
struct point
{
    char label[3];  // a label for a point
    double x ;      // x coordinate for point in a Cartesian coordinate
system
    double y;       // y coordinate for point in a Cartesian coordinate
system
    double z;       // z coordinate for point in a Cartesian coordinate
system
};

void display_struct_point(struct point x);
double distance (const struct point* a, const struct point* b);
int search(const struct point* struct_array, const char* label, int n);

int main(void)
{

    struct point alpha = { "A1", 2.3, 4.5} ;
    struct point *stp = &alpha;
    printf("Size of struct-point in our Linux lab is: %d bytes.\n",
           (int) sizeof(struct point));
    printf("Size of strcut-point pointer in our Linux lab is: %d
```

```
bytes.\n",
            (int) sizeof(stp));
    printf("Size of strcut that stp points to is:  %d bytes.\n",
            (int) sizeof(*stp));

    display_struct_point(*stp);

    struct point sigma = { "C1", 12.3, 14.5, 56.00 } ;
    struct point omega = { "D1", 125.9, 130.0, 97.00 } ;
    struct point theta = { "E1", 5.9, 303.0, 7.00 } ;
    display_struct_point(sigma);
    display_struct_point(omega);
    printf("\nThe distance between sigma and omega is: %10.2f",
distance(&sigma, &omega));
    printf("\nThe distance between sigma and theta is: %10.2f",
distance(&sigma, &theta));

    return 0;
}

void display_struct_point(struct point x)
{
    printf("\nPoint: %s <%.2lf, %.2lf, %.2lf>", x.label, x.x, x.y, x.z);
}


double distance(const struct point* p1, const struct point* p2)
{
    // This funciton is incomplete and needs to be changed and complted by
    // the students to calculate and return the distance betwn the two
three-D points
    double x_diff = p2->x - p1->x;
    double y_diff = p2->y - p1->y;
    double z_diff = p2->z - p1->z;

    return sqrt((pow(x_diff, 2) + pow(y_diff, 2) + pow(z_diff, 2)));
}
```

Program Output

```
PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab2> cd "c:\Users\davis\Desktop\ENSF 619\Labs\Lab2\" ;
Size of struct-point in our Linux lab is: 32 bytes.
Size of strcut-point pointer in our Linux lab is: 4 bytes.
Size of strcut that stp points to is:  32 bytes.

Point: A1 <2.30, 4.50, 0.00>
Point: C1 <12.30, 14.50, 56.00>
Point: D1 <125.90, 130.00, 97.00>
The distance between sigma and omega is:     167.11
The distance between sigma and theta is:     292.70
PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab2>
```