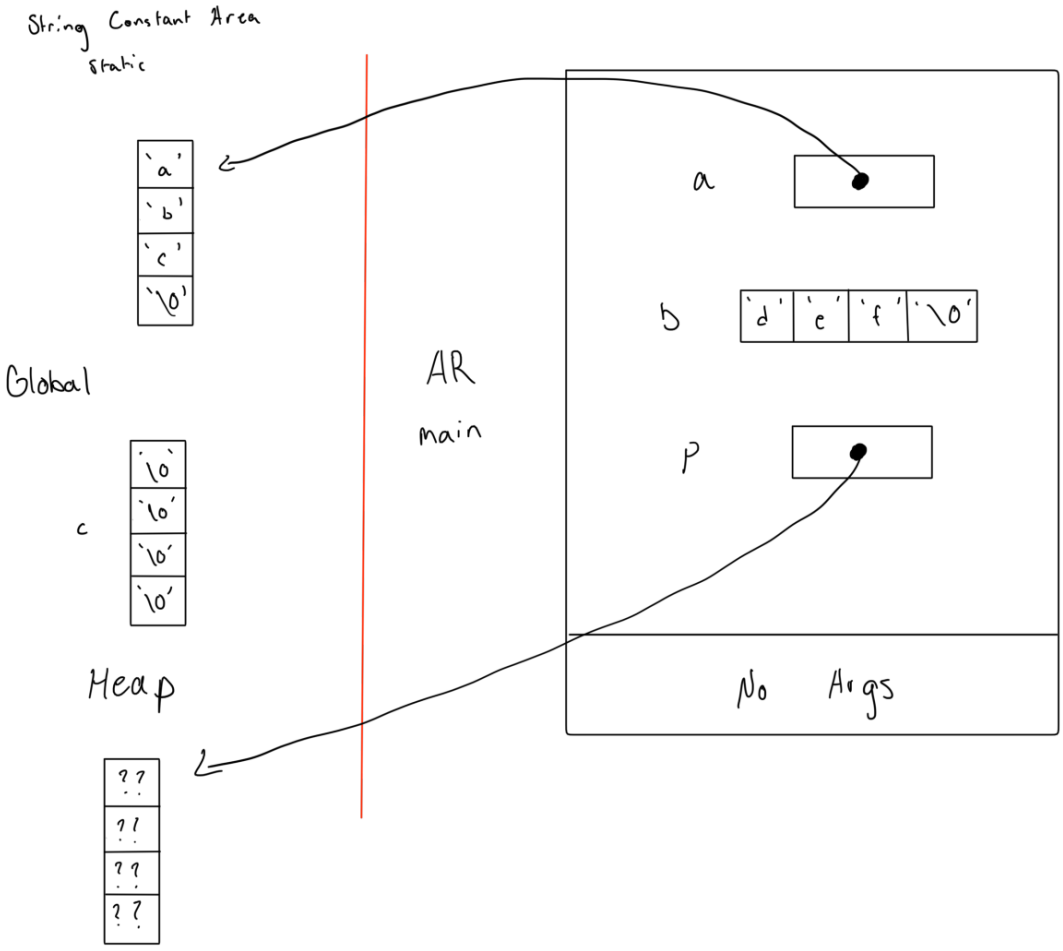
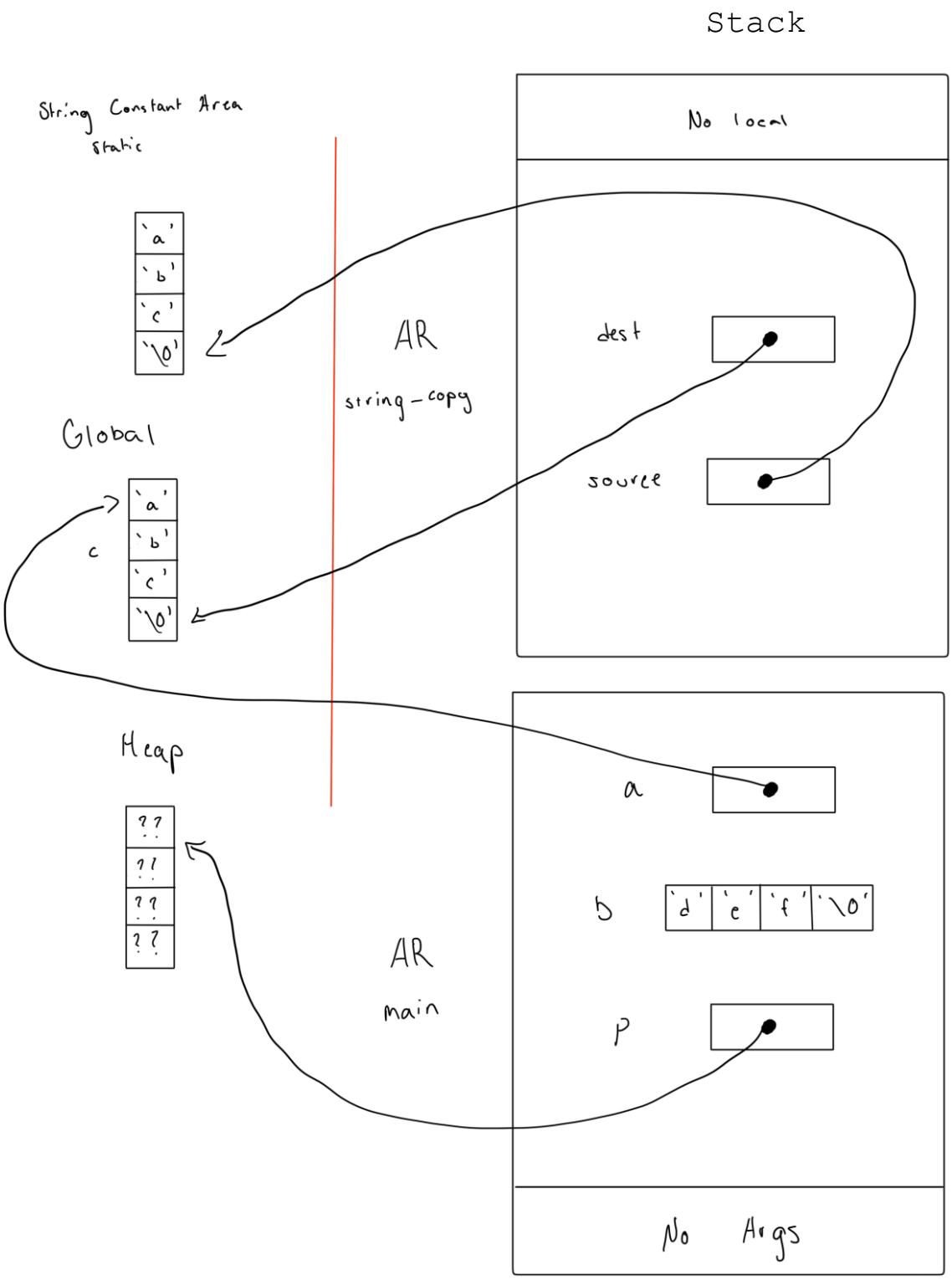
Course: Programming Fundamental – ENSF 337
Lab #: Lab 5
Instructor: M. Moussavi
Student Name: Carl Soriano
Lab Section: B01
Date submitted: Oct 19, 2022

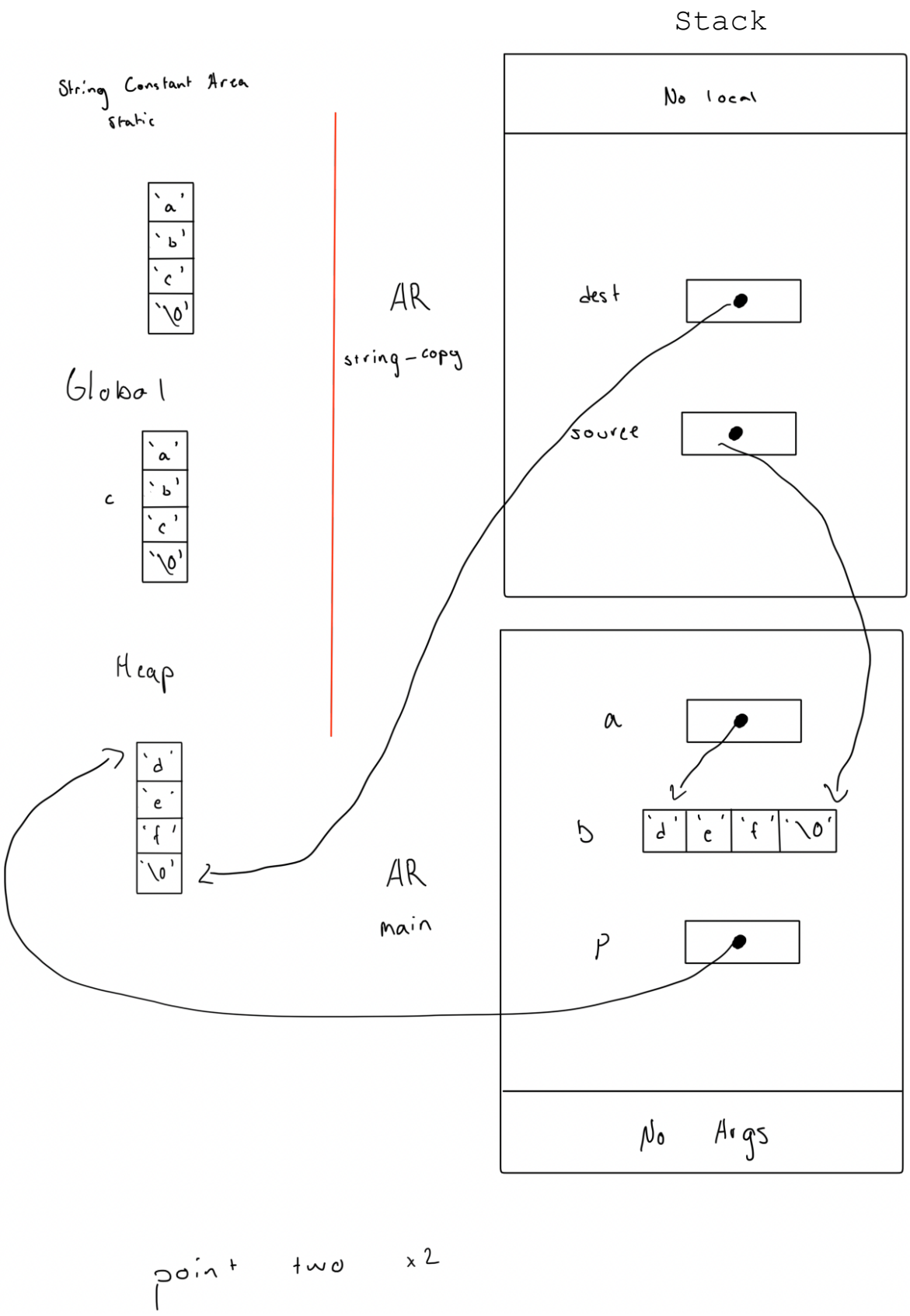
Exercise A



Point One


Stack
String Constant Area
static
'a'
'b'
'c'
'\0'
AR
string-copy
No local
dest
source
Global
c
'a'
'b'
'c'
'\0'
Heap
??
??
??
??
a
b
'd'
'e'
'f'
'\0'
AR
main
p
No Args

point two


Stack
String Constant Area
static
'a'
'b'
'c'
'\0'
Global
c
'a'
'b'
'c'
'\0'
Heap
'd'
'e'
'f'
'\0'
AR
string-copy
No local
dest
source
AR
main
a
b
'd'
'e'
'f'
'\0'
p
No Args

point two x2

Exercise B

```c
/*
 *  File: lab5exB.c
 *  ENSF 337, lab 5, Exercise B
 */
#include <stdio.h>
#include <stdlib.h>

// This is a simple C program that is supposed to create an array of type double,
// (dyanamically on the heap), filling it with some arbitrary numbers and then
// using the array as needed. But the program doesn't do any thing useful becaues
// some flaws in the main function and improper design of the function
// create_array.

double* create_array(double * x, unsigned long n);
void populate_array(double *array, int n);

int main(void) {
    printf("\nProgram started...\n");
    double *array;
    int n = 20;
    array = create_array(array,n);

    if( array != NULL) {
        populate_array(array, n);

        // displays half of the values of the array
        for(int i = 0; i < n/2; i++){

            printf("%lf\n", array[i]);
        }

        // According to C standard, the program's behaviour, after the following
        // call to the function free is considered "Undefined" and needs to be fixed.
        free(array);
    }

    printf("Program terminated...\n");
    return 0;
}

// THE FOLLOWING FUNCTION IS NOT PROPERLY DESINGED AND NEEDS TO BE FIXED
double* create_array(double *x, unsigned long n) {
    double *array;
    array = malloc(n *sizeof(double));
    if(array == NULL){
        printf("Sorry Memory Not Available. Program Terminated.\n");
        exit(1);
    }
    return array;
}

void populate_array(double *array, int n) {
    int i;
```

```
Exercise_B              Exercise_B.c.xcodeproj
[(base) MacBook-Pro:Exercise_B carlsoriano$ cd Exercise_B
[(base) MacBook-Pro:Exercise_B carlsoriano$ ls
a.out   main.c
[(base) MacBook-Pro:Exercise_B carlsoriano$ gcc main.c
[(base) MacBook-Pro:Exercise_B carlsoriano$ s
-bash: s: command not found
[(base) MacBook-Pro:Exercise_B carlsoriano$ /a.out
-bash: /a.out: No such file or directory
[(base) MacBook-Pro:Exercise_B carlsoriano$ ./a.out

Program started...
100.000000
200.000000
300.000000
400.000000
500.000000
600.000000
700.000000
800.000000
900.000000
1000.000000
Program terminated...
(base) MacBook-Pro:Exercise_B carlsoriano$
```

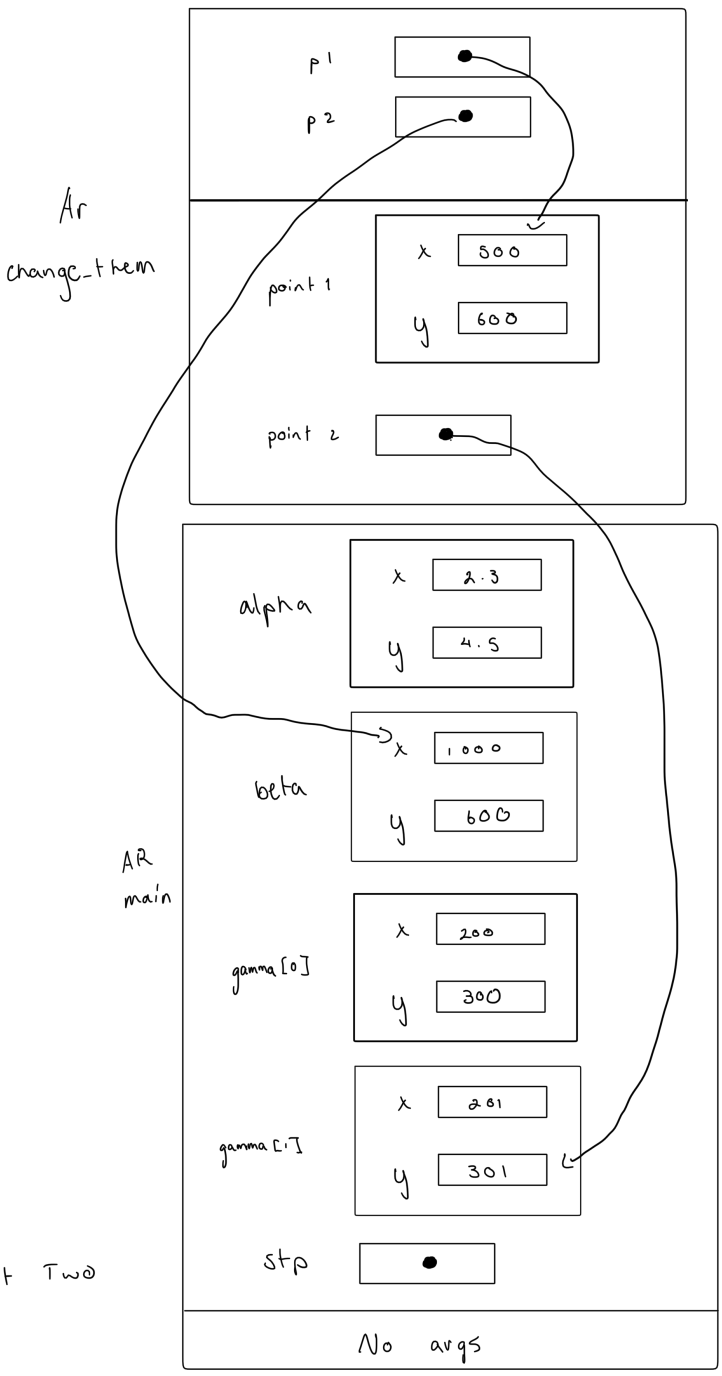
Exercise C

Static

SIZE

'2'

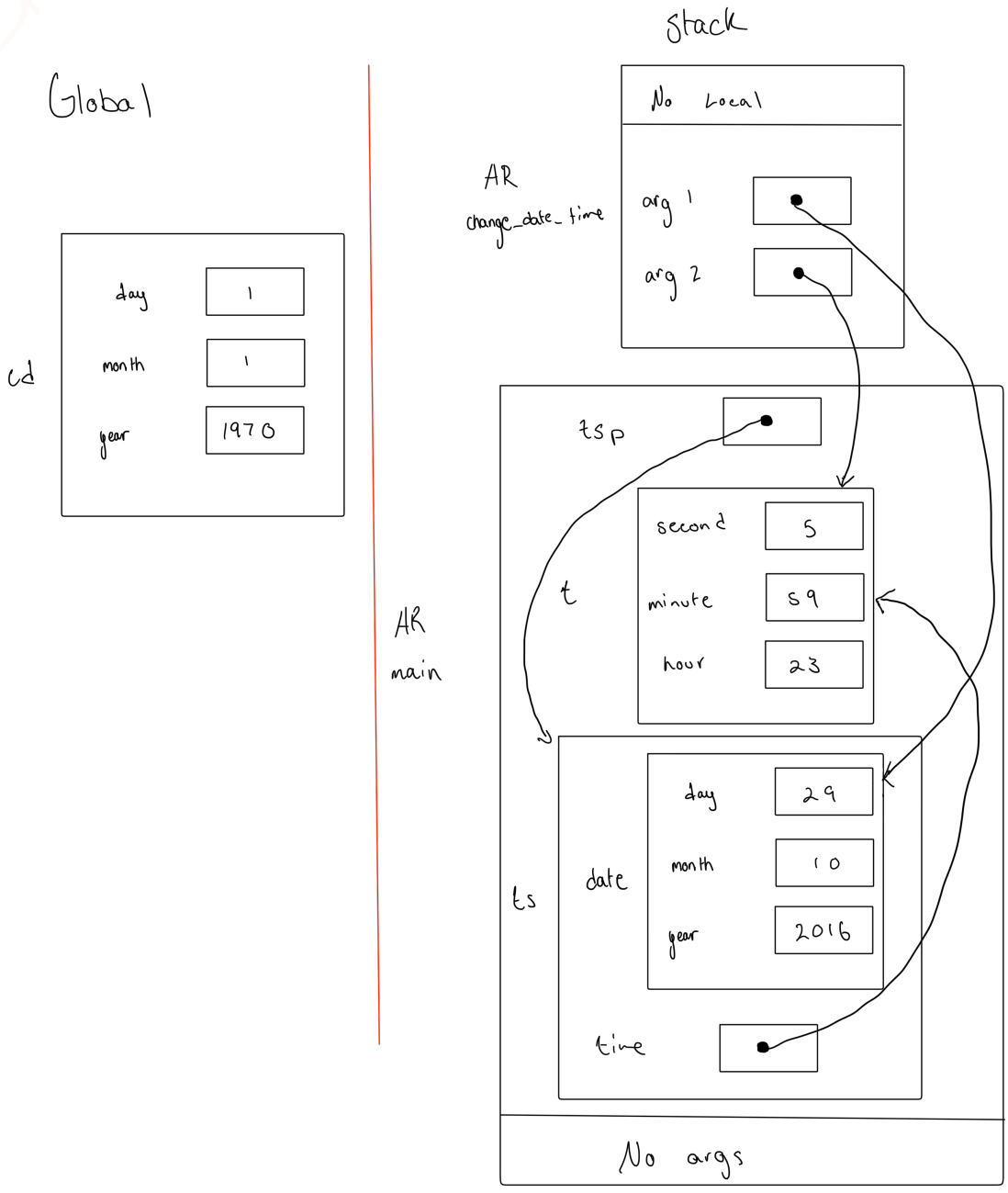
Stack



No args

Point Two

Exercise D


Stack
Global
No Local
AR
change_date_time
arg 1
arg 2
cd
day
1
month
1
year
1970
tsp
second
5
t
minute
59
AR
main
hour
23
day
29
date
month
10
ts
year
2016
time
No args

Exercise E

```c
/* File: lab5exE.c
 * ENSF 337 - lab 5 - Exercise E
 */

#include "lab5exE.h"
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(void)
{
    Point alpha = { "A1", 2.3, 4.5, 56.0} ;
    Point beta = { "B1", 25.9, 30.0, 97.0 } ;
    printf ("Display the values in alpha, and beta: ");
    display_struct_point(alpha);
    display_struct_point(beta);

    Point *stp = &alpha;
    printf ("\n\nDisplay the values in *stp: ");
    display_struct_point(*stp);

    Point gamma = mid_point(stp, &beta, "M1");
    printf ("\n\nDisplay the values in gamma after calling mid_point function.");
    printf ("Expected result is: M1 <14.10, 17.25, 76.50>");

    printf("\n\nThe actual result of calling mid_point function is: ");
    display_struct_point(gamma);

    swap (stp, &beta);
    printf ("\n\nDisplay the values in *stp, and beta after calling swap function.");
    printf ("Expected to be:\nB1 <25.90, 30.00, 97.00>\nA1 <2.30, 4.50, 56.00>");


    printf("\n\nThe actual result of calling swap function is: ");
    display_struct_point(*stp);
    display_struct_point(beta);


    printf("\n\nThe distance between alpha and beta is: %.2f. ", distance(&alpha, &beta));
    printf ("(Expected to be: 53.74)");
    printf("\nThe distance between gamma and beta is: %.2f. ", distance(&gamma, &beta));
    printf ("(Expected to be: 26.87) \n");

    return 0;
}

void display_struct_point(const Point x)
{
    printf("\n%s <%.2lf, %.2lf, %.2f>", x.label, x.x, x.y, x.z);
}


Point mid_point(const Point* p1, const Point* p2, const char* label)
{
    // This function is incomplete and must be completed by the students
    // YOU ARE NOT ALLOWED TO USE ANY STRING LIBRARY FUNCTIONS IN THIS FUNCTION

    Point middle = {"?", 0, 0, 0};
    middle.x = ((*p1).x + (*p2).x)/2;
    middle.y = ((*p1).y + (*p2).y)/2;
    middle.z = ((*p1).z + (*p2).z)/2;
    int i = 0;
    while (*label != '\0')
    {
        middle.label[i] = *label;
        label++;
        i++;
    }
    middle.label[i] = '\0';
    return middle;
}

void swap(Point* p1, Point *p2)
{

    Point mem;
    mem = *p1;
    *p1 = *p2;
    *p2 = mem;

    // This function is incomplete and must be completed by the students
}

double distance(const Point* p1, const Point* p2)
{

    double d,x,y,z;
    x = pow((*p1).x - ((*p2).x),2);
    y = pow((*p1).y - ((*p2).y),2);
    z = pow((*p1).z - ((*p2).z),2);




    // This function is incomplete and must be completed by the students
    // NOTE: IN THIS FUNCTION YOU ARE NOT ALLOWED TO USE THE ARROW OPERATOR ->

    return sqrt(x + y + z);
}
```

```
Display the values in alpha, and beta:
A1 <2.30, 4.50, 56.00>
B1 <25.90, 30.00, 97.00>

Display the values in *stp:
A1 <2.30, 4.50, 56.00>

Display the values in gamma after calling mid_point function.Expected result is:
 M1 <14.10, 17.25, 76.50>

The actual result of calling mid_point function is:
M1 <14.10, 17.25, 76.50>

Display the values in *stp, and beta after calling swap function.Expected to be:
B1 <25.90, 30.00, 97.00>
A1 <2.30, 4.50, 56.00>

The actual result of calling swap function is:
B1 <25.90, 30.00, 97.00>
A1 <2.30, 4.50, 56.00>

The distance between alpha and beta is: 53.74. (Expected to be: 53.74)
The distance between gamma and beta is: 26.87. (Expected to be: 26.87)
(base) MacBook-Pro:LAB5_Extra carlsoriano$
```

Exercise F

```c
// lab5exF.c
// ENSF 337, Exercise F

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

int main(void)
{
    Point struct_array[10];
    int i;
    int position;

    populate_struct_array(struct_array, 10);

    printf("\nArray of Points contains: \n");

    for(i=0; i < 10; i++)
        display_struct_point(struct_array[i], i);


    printf("\nTest the search function");

    position = search(struct_array, "v0", 10);
    if(position != -1)
        printf("\nFound: struct_array[%d] contains %s", position,
                struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "v0");

    position = search(struct_array, "E1", 10);
    if(position != -1)
        printf("\nFound: struct_array[%d] contains %s", position,
                struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "E1");

    position = search(struct_array, "C5", 10);

    if(position != -1)
        printf("\nFound: struct_array[%d] contains %s", position,
                struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "C5");

    position = search(struct_array, "B7", 10);
    if(position != -1)
        printf("\nFound: struct_array[%d] contains %s", position,
                struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "B7");
    position = search(struct_array, "A9", 10);
    if(position != -1)
        printf("\nFound: struct_array[%d] contains %s", position,
                struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "A9");
    position = search(struct_array, "E11", 10);
    if(position != -1)
        printf("\nFound: struct_array[%d] contains %s", position,
                struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "E11");

    position = search(struct_array, "M1", 10);
    if(position != -1)
        printf("\nFound: struct_array[%d] contains %s", position,
                struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "M1");

    printf("\n\nTesting the reverse function:");

    reverse(struct_array, 10);

    printf("\nThe reversed array is:");

    for(i=0; i < 10; i++)
        display_struct_point(struct_array[i], i);

    return 0;
}


void display_struct_point(const Point x , int i)
{
    printf("\nstruct_array[%d]: %s <%.2lf, %.2lf, %.2lf>\n",
           i, x.label, x.x, x.y, x.z);
}
```

```c
void populate_struct_array(Point* array, int n)
{
    int i;
    char ch1 = 'A';
    char ch2 = '9';
    char ch3 = 'z';

    for( i = 0; i < 10; i++)
    {
        /* generating some random values to fill them elements of the array: */
        array[i].x = (7 * (i + 1) % 11) * 100 - i /2;
        array[i].y = (7 * (i + 1) % 11) * 120 - i / 3;
        array[i].z = (7 * (i + 1) % 11) * 150 - i /4;

        if(i % 2 == 0)
            array[i].label[0] = ch1++;
        else
            array[i].label[0] = ch3--;
        array[i].label[1] = ch2--;
        array[i].label[2] = '\0';
    }
}

int search(const Point* struct_array, const char* label, int n)
{
    int index = 0;

    while(index < n){
        if (*struct_array[index].label == *label){
            return index;
        }
        index++;
    }

    return -1;
}

void reverse (Point *a, int n)
{
    Point copy[n];

        for (int m = 0; m < n; m++) {
            copy[n - 1 - m] = a[m];
        }
        for (int m = 0; m < n; m++) {
            a[m] = copy[m];
        }

}
```

```
a.out   main.c  main.h
(base) MacBook-Pro:Exercise_F carlsoriano$ ./a.out

Array of Points contains:

struct_array[0]: A9 <700.00, 840.00, 1050.00>

struct_array[1]: z8 <300.00, 360.00, 450.00>

struct_array[2]: B7 <999.00, 1200.00, 1500.00>

struct_array[3]: y6 <599.00, 719.00, 900.00>

struct_array[4]: C5 <198.00, 239.00, 299.00>

struct_array[5]: x4 <898.00, 1079.00, 1349.00>

struct_array[6]: D3 <497.00, 598.00, 749.00>

struct_array[7]: w2 <97.00, 118.00, 149.00>

struct_array[8]: E1 <796.00, 958.00, 1198.00>

struct_array[9]: v0 <396.00, 477.00, 598.00>

Test the search function
Found: struct_array[9] contains v0
Found: struct_array[8] contains E1
Found: struct_array[4] contains C5
Found: struct_array[2] contains B7
Found: struct_array[0] contains A9
Found: struct_array[8] contains E1
struct_array doesn't have label: M1.

Testing the reverse function:
The reversed array is:
struct_array[0]: v0 <396.00, 477.00, 598.00>

struct_array[1]: E1 <796.00, 958.00, 1198.00>

struct_array[2]: w2 <97.00, 118.00, 149.00>

struct_array[3]: D3 <497.00, 598.00, 749.00>

struct_array[4]: x4 <898.00, 1079.00, 1349.00>

struct_array[5]: C5 <198.00, 239.00, 299.00>

struct_array[6]: y6 <599.00, 719.00, 900.00>

struct_array[7]: B7 <999.00, 1200.00, 1500.00>

struct_array[8]: z8 <300.00, 360.00, 450.00>

struct_array[9]: A9 <700.00, 840.00, 1050.00>
(base) MacBook-Pro:Exercise_F carlsoriano$
```