Course : Programming Fundamental – ENSF 337

Lab # : Lab 5

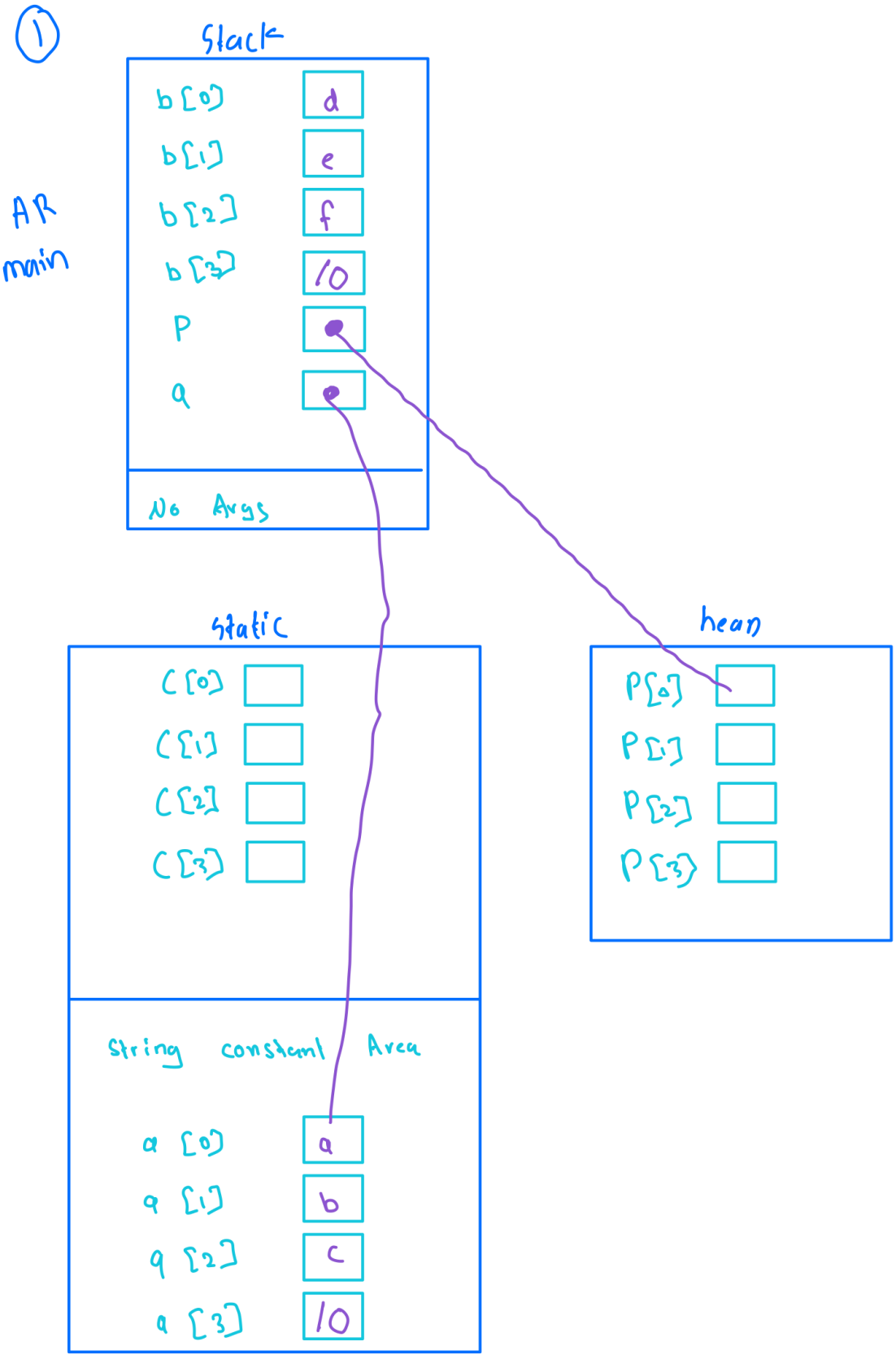
Instructor : M. Moussavi

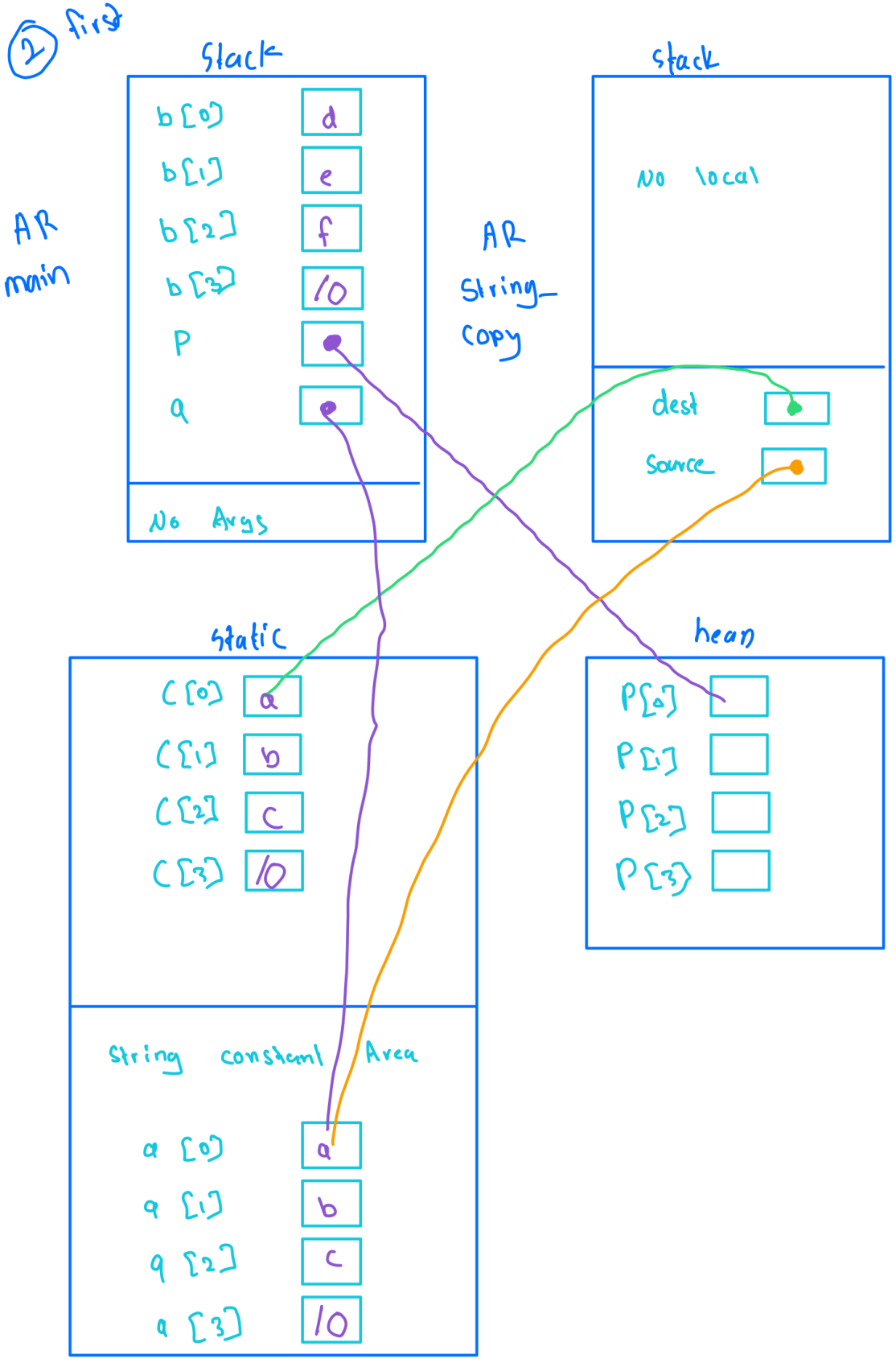
Student Name : Nimna Wijedasa

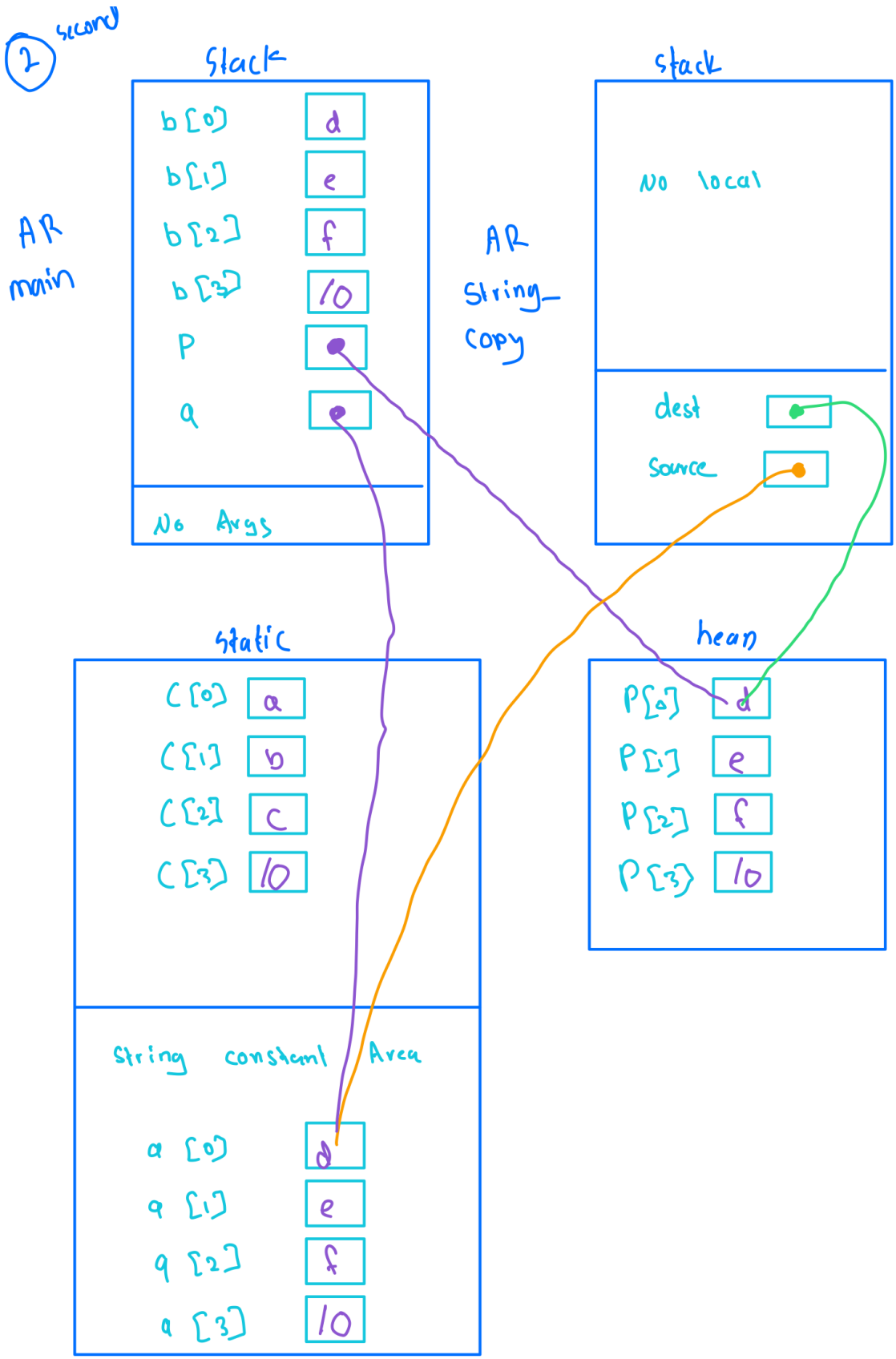
Lab Section : B02

Date submitted : Oct 21, 2022

Exercise A


1
Stack
AR
main
b[0]
d
b[1]
e
b[2]
f
b[3]
10
P
q
No Args
static
C[0]
C[1]
C[2]
C[3]
String constant Area
a[0]
a
a[1]
b
a[2]
c
a[3]
10
heap
P[0]
P[1]
P[2]
P[3]

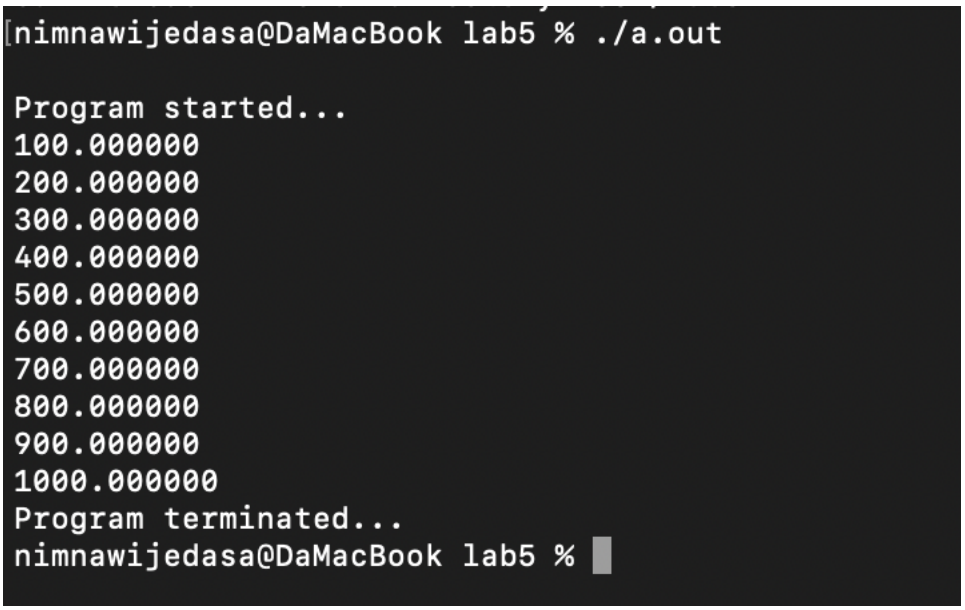

2 first
stack
AR main
b[0]
b[1]
b[2]
b[3]
P
q
d
e
f
\0
No Args
stack
AR String_ Copy
No local
dest
source
static
C[0]
C[1]
C[2]
C[3]
a
b
c
\0
String constant Area
a[0]
a[1]
a[2]
a[3]
a
b
c
\0
heap
P[0]
P[1]
P[2]
P[3]


2 second
stack
b[0] d
b[1] e
b[2] f
b[3] 10
p
q
No Args
AR main
stack
No local
dest
source
AR String_ copy
static
c[0] a
c[1] b
c[2] c
c[3] 10
String constant Area
a[0] d
a[1] e
a[2] f
a[3] 10
heap
p[0] d
p[1] e
p[2] f
p[3] 10

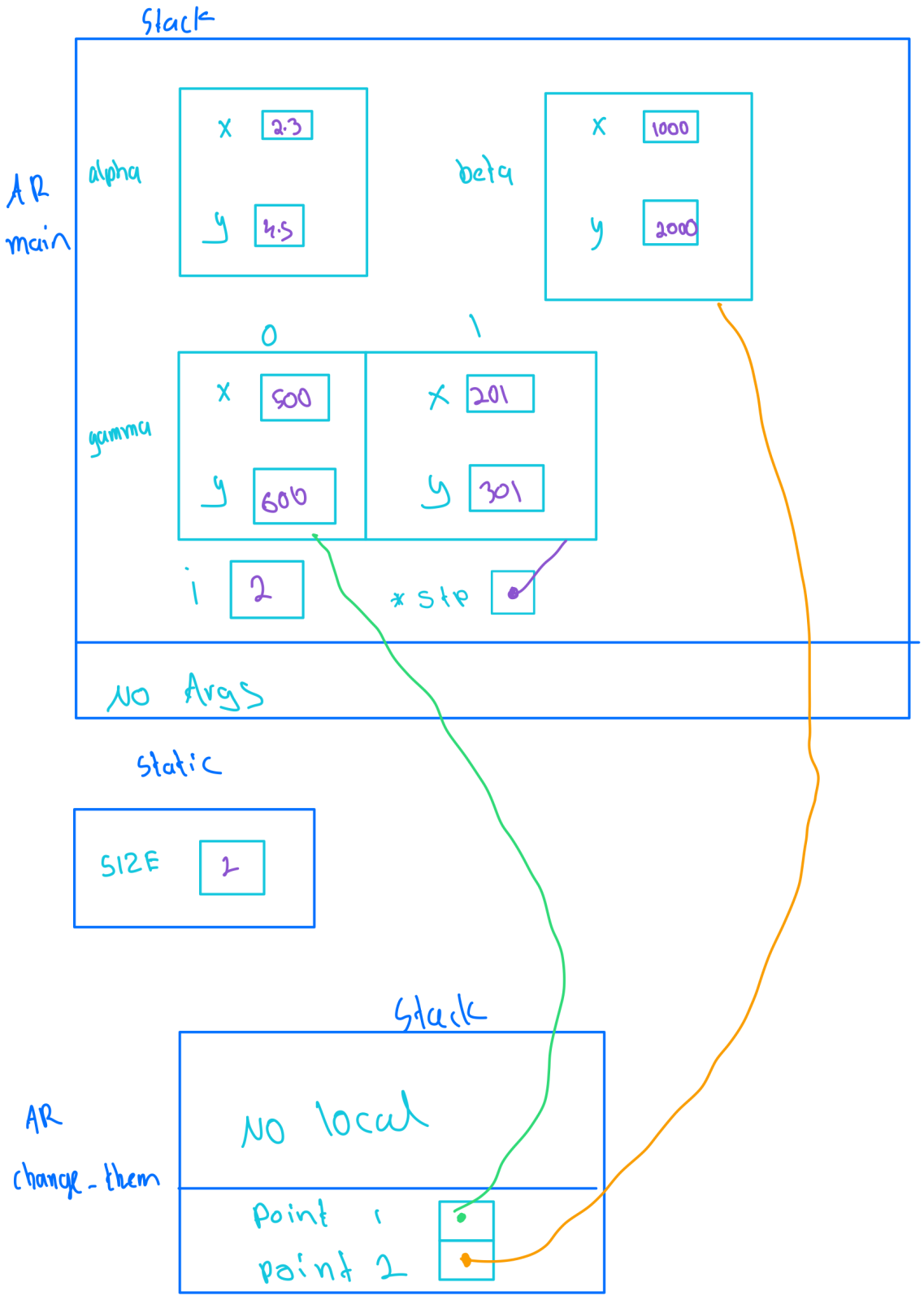
Exercise B

C Lab5exB.c 〉No Selection

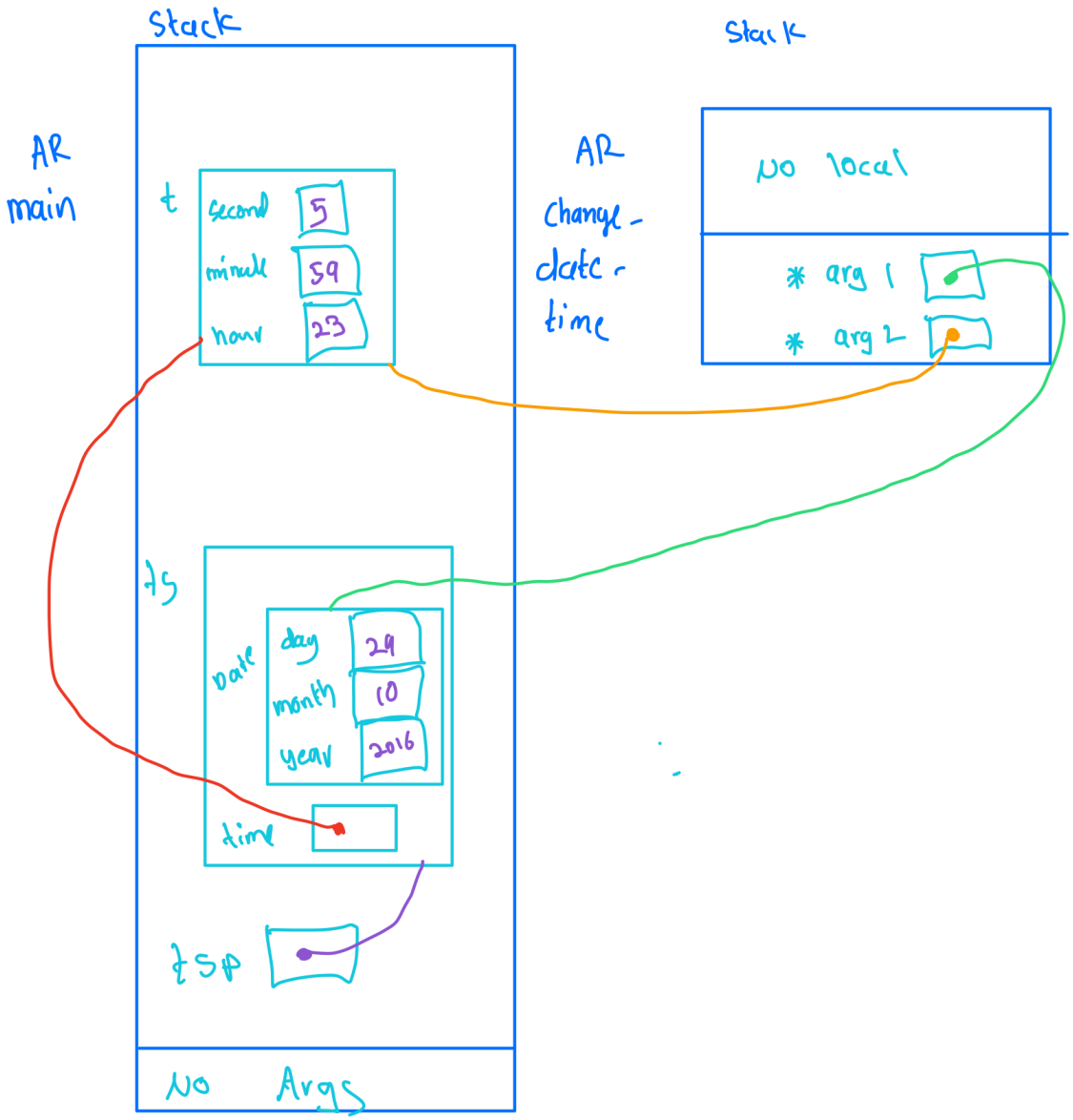
```c
/*
 *  File: lab5exB.c
 *  ENSF 337, lab 5, Exercise B
 */
#include <stdio.h>
#include <stdlib.h>

// 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 = NULL;
    int n = 20;
    int count = 0;
    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("%f\n", *array++);
            count++;
        }

        // 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 - count));
    }

    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) {
    x = malloc(n *sizeof(double));
    if(x == NULL){
        printf("Sorry Memory Not Available. Program Terminated.\n");
        exit(1);
    }
    return (x);
}

void populate_array(double *array, int n) {
    int i;
    for(i = 0; i < n; i++)
        array[i] = (i + 1) * 100;
}

```


nimnawijedasa@DaMacBook lab5 % ./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...
nimnawijedasa@DaMacBook lab5 %

Exercise C


Stack
AR
main
alpha
x
2.3
y
4.5
beta
x
1000
y
2000
0
1
gamma
x
500
y
606
x
201
y
301
i
2
*stp
No Args
Static
SIZE
2
Stack
AR
change_them
No local
point 1
point 2

Exercise D


Stack
AR
main
t
second
5
minute
59
hour
23
ts
date
day
29
month
10
year
2016
time
tsp
No Args
Stack
AR
Change-
date-
time
No local
* arg 1
* arg 2


Static
cD
day
1
month
1
year
1970

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 = { .label: "A1", .x: 2.3, .y: 4.5, .z: 56.0} ;
    Point beta = { .label: "B1", .x: 25.9, .y: 30.0, .z: 97.0 } ;
    printf ("Display the values in alpha, and beta: ");
    display_struct_point( x: alpha);
    display_struct_point( x: beta);

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

    Point gamma = mid_point( p1: stp, p2: &beta, label: "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( x: gamma);

    swap ( p1: stp, p2: &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( x: *stp);
    display_struct_point( x: beta);

```

```c
    printf("\n\nThe distance between alpha and beta is: %.2f. ", distance( a: &alpha,  b: &beta));
    printf ("(Expected to be: 53.74)");
    printf("\nThe distance between gamma and beta is: %.2f. ", distance( a: &gamma,  b: &beta));
    printf ("(Expected to be: 26.87)");
    return 0;
}

void display_struct_point(const Point x)
{
    printf("\n%s <%.2lf, %.2lf, %.2lf>", 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
    double mid_x = ((p1->x) + (p2->x))/2;
    double mid_y = ((p1->y) + (p2->y))/2;
    double mid_z = ((p1->z) + (p2->z))/2;
    Point middle = { .label: "M1",  .x: mid_x,  .y: mid_y,  .z: mid_z};

    return middle;
}

void swap(Point* p1, Point *p2)
{
    // This function is incomplete and must be completed by the students
        struct point temp = *p1;
        *p1 = *p2;
        *p2 = temp;


}

double distance(const Point* p1, const Point* p2)
{
    // This function is incomplete and must be completed by the students
    // NOTE: IN THIS FUNCTION YOU ARE NOT ALLOWED TO USE THE ARROW OPERATOR ->
    double x = ((*p1).x) - ((*p2).x);
    double y = ((*p1).y) - ((*p2).y);
    double z = ((*p1).z) - ((*p2).z);
    double distance = sqrt((x*x) + (y*y) + (z*z) )  ;
    return distance;
}
```

Exercise F

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

#include "lab5exF.h"
#include <stdio.h>
#include <math.h>
#include<string.h>

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

    populate_struct_array( array: struct_array,  n: 10);

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

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


    printf("\nTest the search function");

    position = search(struct_array,  target: "v0",  n: 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,  target: "E1",  n: 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");

```

```
38
39      position = search(struct_array, target: "C5", n: 10);
40
41      if(position != -1)
42          printf("\nFound: struct_array[%d] contains %s", position,
43                 struct_array[position].label);
44      else
45          printf("\nstruct_array doesn't have label: %s.", "C5");
46
47      position = search(struct_array, target: "B7", n: 10);
48      if(position != -1)
49          printf("\nFound: struct_array[%d] contains %s", position,
50                 struct_array[position].label);
51      else
52          printf("\nstruct_array doesn't have label: %s.", "B7");
53      position = search(struct_array, target: "A9", n: 10);
54      if(position != -1)
55          printf("\nFound: struct_array[%d] contains %s", position,
56                 struct_array[position].label);
57      else
58          printf("\nstruct_array doesn't have label: %s.", "A9");
59      position = search(struct_array, target: "E11", n: 10);
60      if(position != -1)
61          printf("\nFound: struct_array[%d] contains %s", position,
62                 struct_array[position].label);
63      else
64          printf("\nstruct_array doesn't have label: %s.", "E11");
65
66      position = search(struct_array, target: "M1", n: 10);
67      if(position != -1)
68          printf("\nFound: struct_array[%d] contains %s", position,
69                 struct_array[position].label);
70      else
71          printf("\nstruct_array doesn't have label: %s.", "M1");
72
73      printf("\n\nTesting the reverse function:");
74
75      reverse( a: struct_array, n: 10);
```

```c
    reverse( a: struct_array,  n: 10);

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

    for(i=0; i < 10; i++)
        display_struct_point( x: 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);
}

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';
    }
```

```
        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)
{
    // Students should complete the definiton of this function
    // NOTE: YOU ARE NOT ALLOWED TO USE ANY C LIBRARY FUNCTION IN YOUR SOLUTION
    for( int i = 0; i < n; i++)
    {
        if(struct_array[i].label[0] == label[0] &&
            struct_array[i].label[1] == label[1] &&
            struct_array[i].label[2] == label[2]){
            return 0;
        }
    }

    return -1;
}

void reverse (Point *a, int n)
{
    // Students should complete the definiton of this function
    Point temp_array[10];
    int j = 1;
    for (int i = 0; i < n ; i++){
        temp_array[i] = a[n - j];
        j++;
    }
    for (int i = 0; i < n ; i++){
        a[i] = temp_array[i] ;
    }

}
```

```
/Users/nimnawijedasa/Desktop/fall/337/cmake-build-debug/337

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[0] contains A9
Found: struct_array[0] contains A9
Found: struct_array[0] contains A9
Found: struct_array[0] contains A9
Found: struct_array[0] contains A9
struct_array doesn't have label: E11.
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>
```

```
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>

Process finished with exit code 0
```