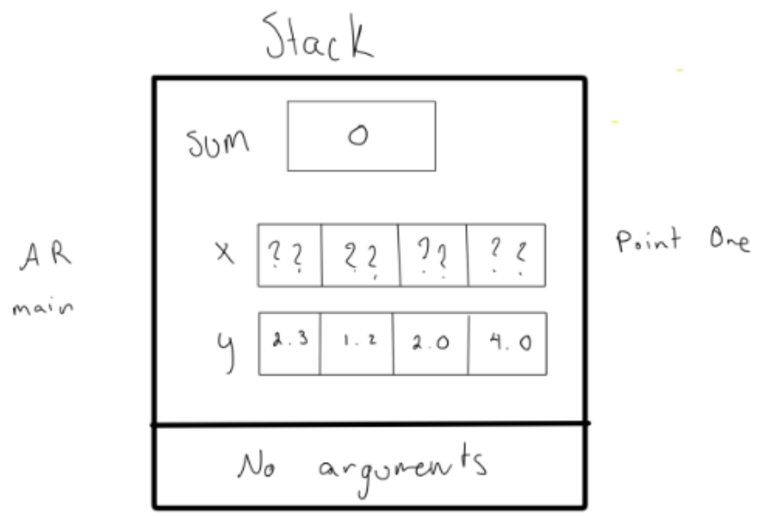
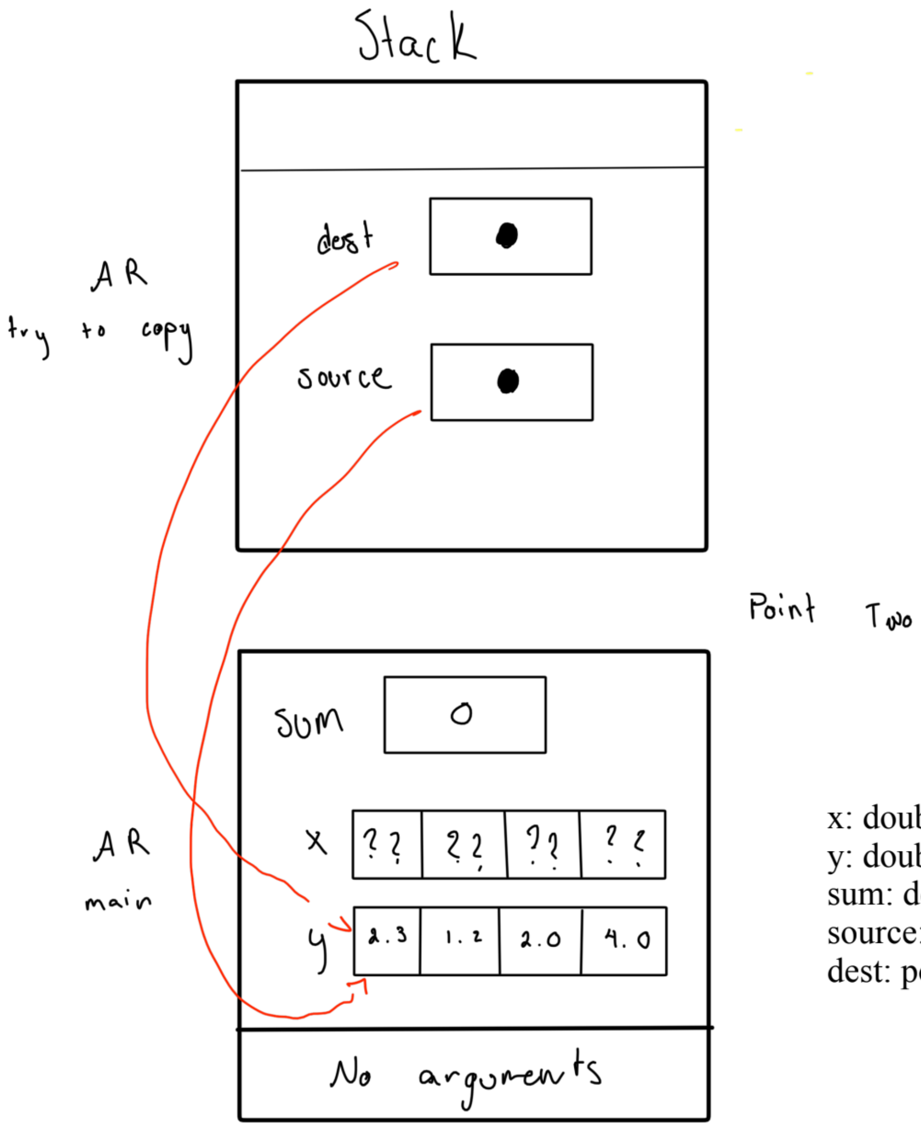
Course: Programming Fundamental – ENSF 337
Lab #: Lab 3
Instructor: M. Moussavi
Student Name: Carl Soriano
Lab Section: B01
Date submitted: Oct 5, 2022

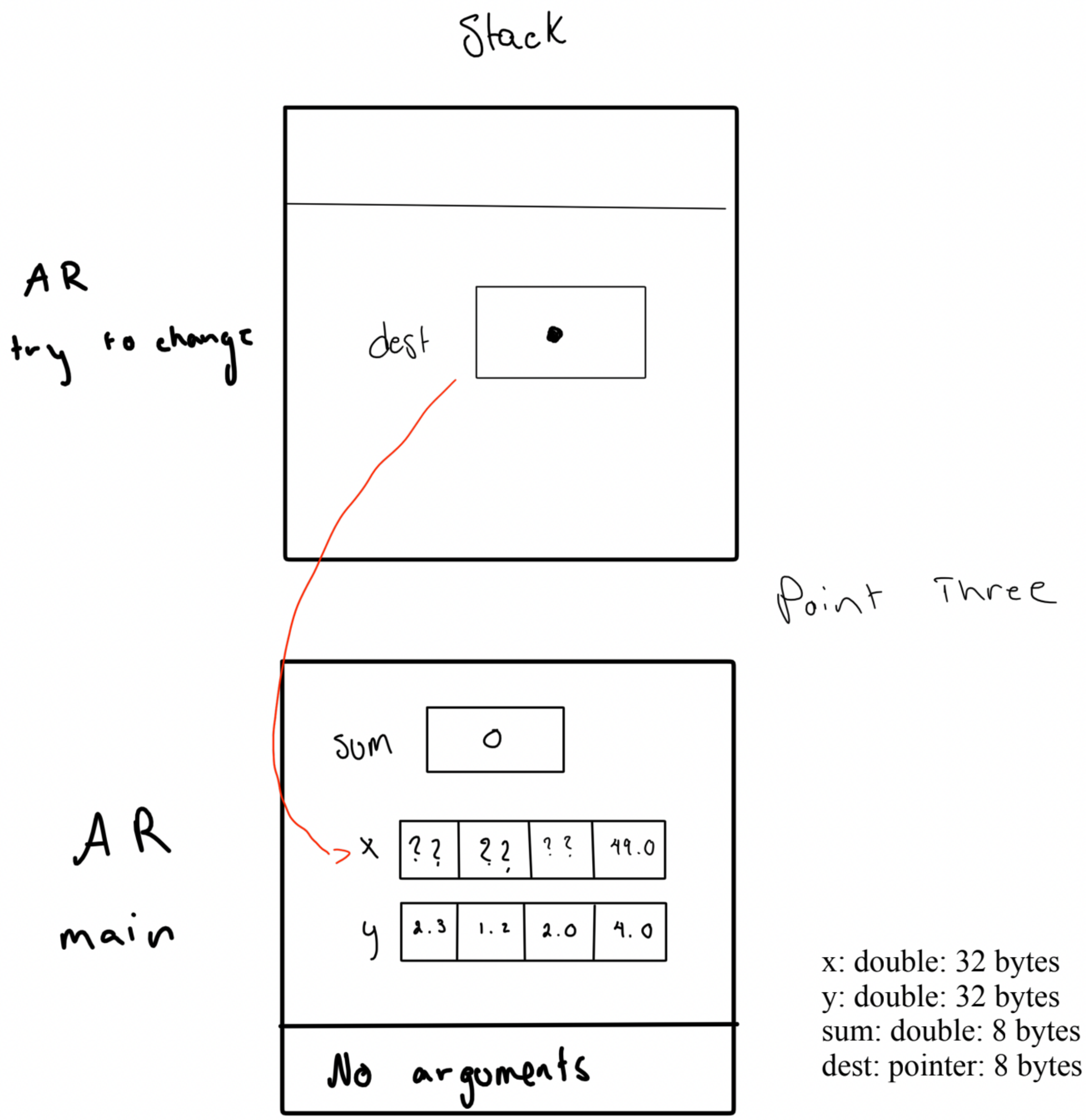
Exercise A



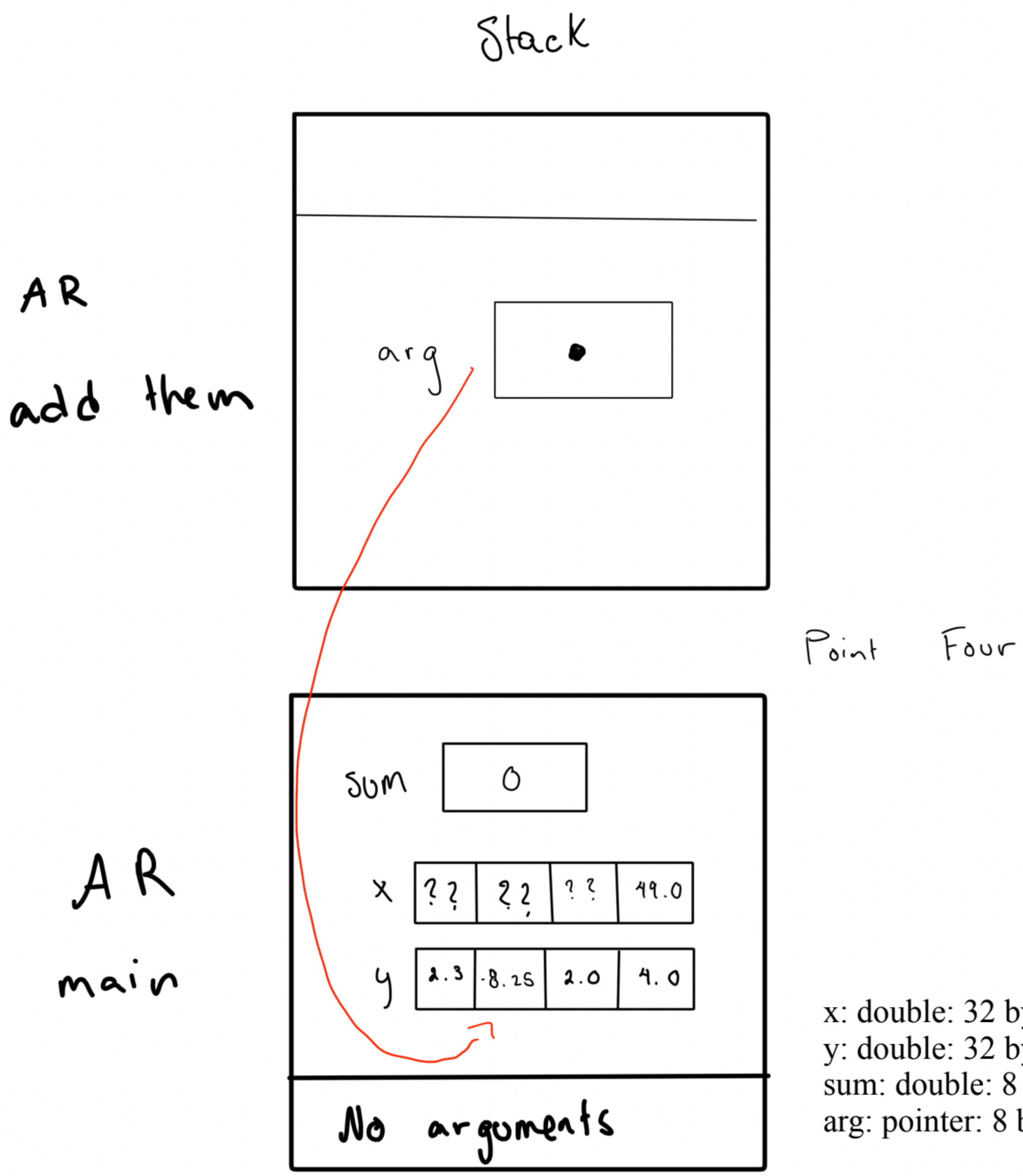
x: double: 32 bytes
y: double: 32 bytes
sum: double: 8 bytespointer: 8 bytes



x: double: 32 bytes
y: double: 32 bytes
sum: double: 8 bytes
source: pointer: 8 bytes
dest: pointer: 8 bytes

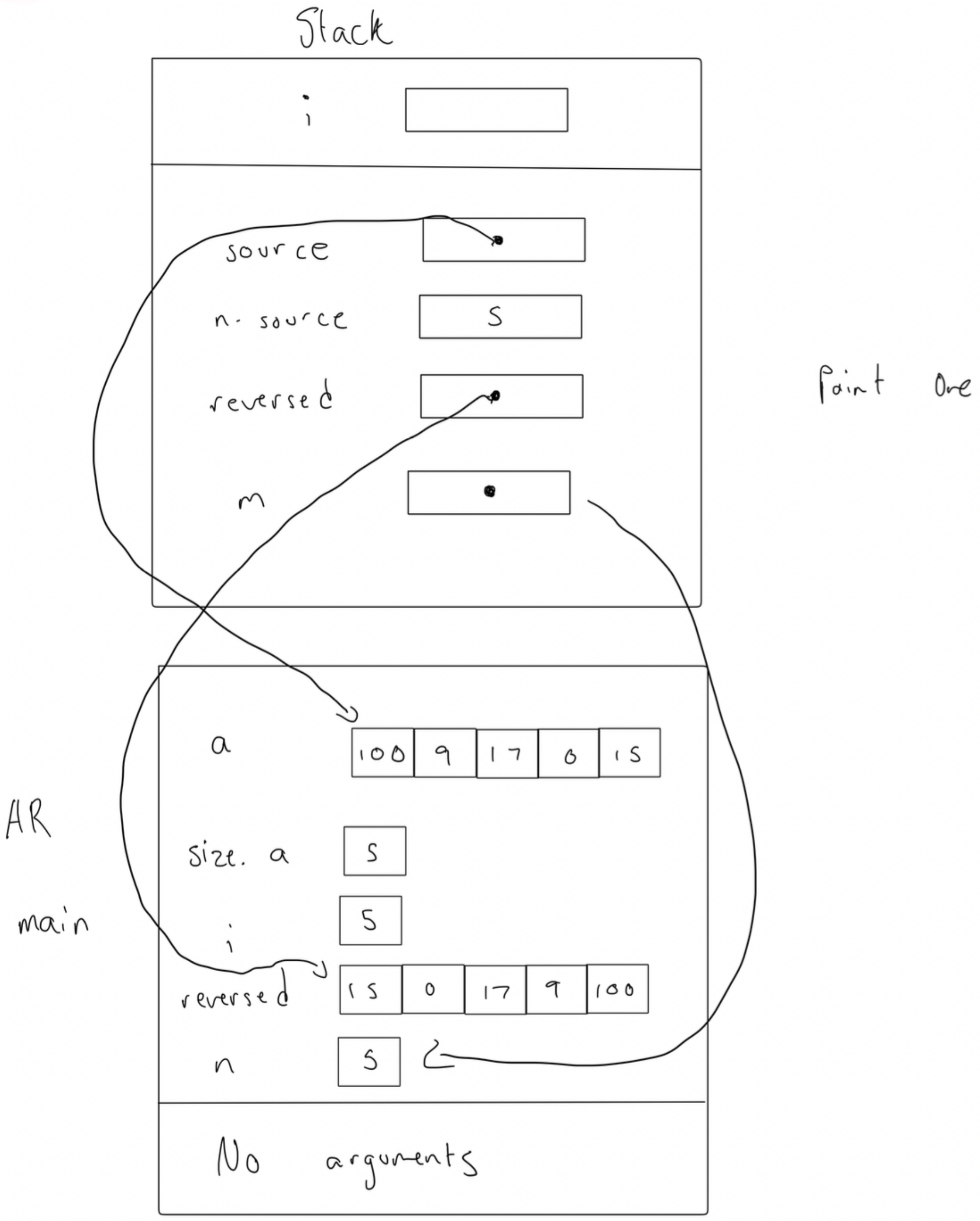


x: double: 32 bytes
y: double: 32 bytes
sum: double: 8 bytes
dest: pointer: 8 bytes



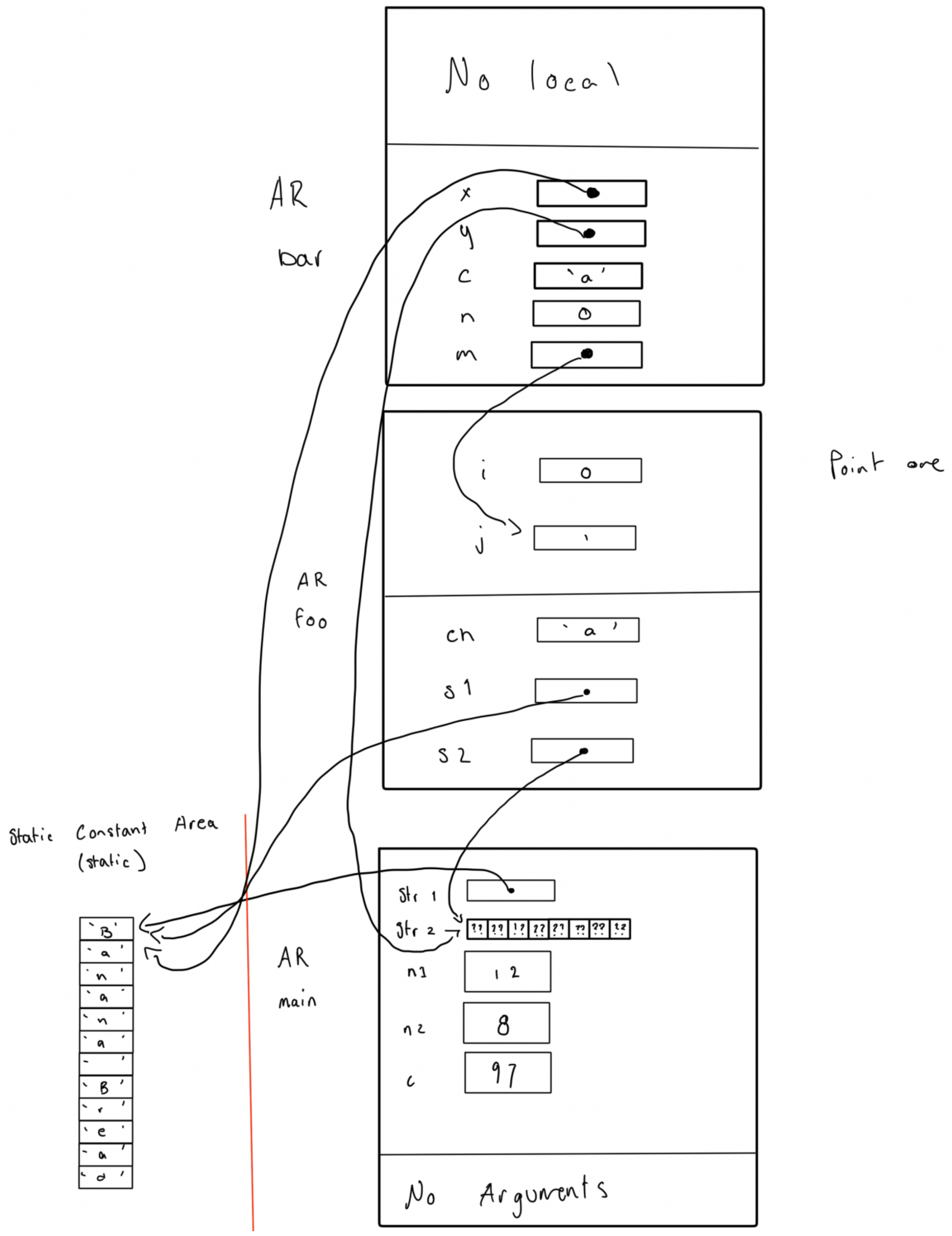
x: double: 32 bytes
y: double: 32 bytes
sum: double: 8 bytes
arg: pointer: 8 bytes

Exercise B


Stack
i
source
n. source
5
reversed
m
Point One
AR
main
a
100
9
17
0
15
size. a
5
i
5
reversed
15
0
17
9
100
n
5
No arguments

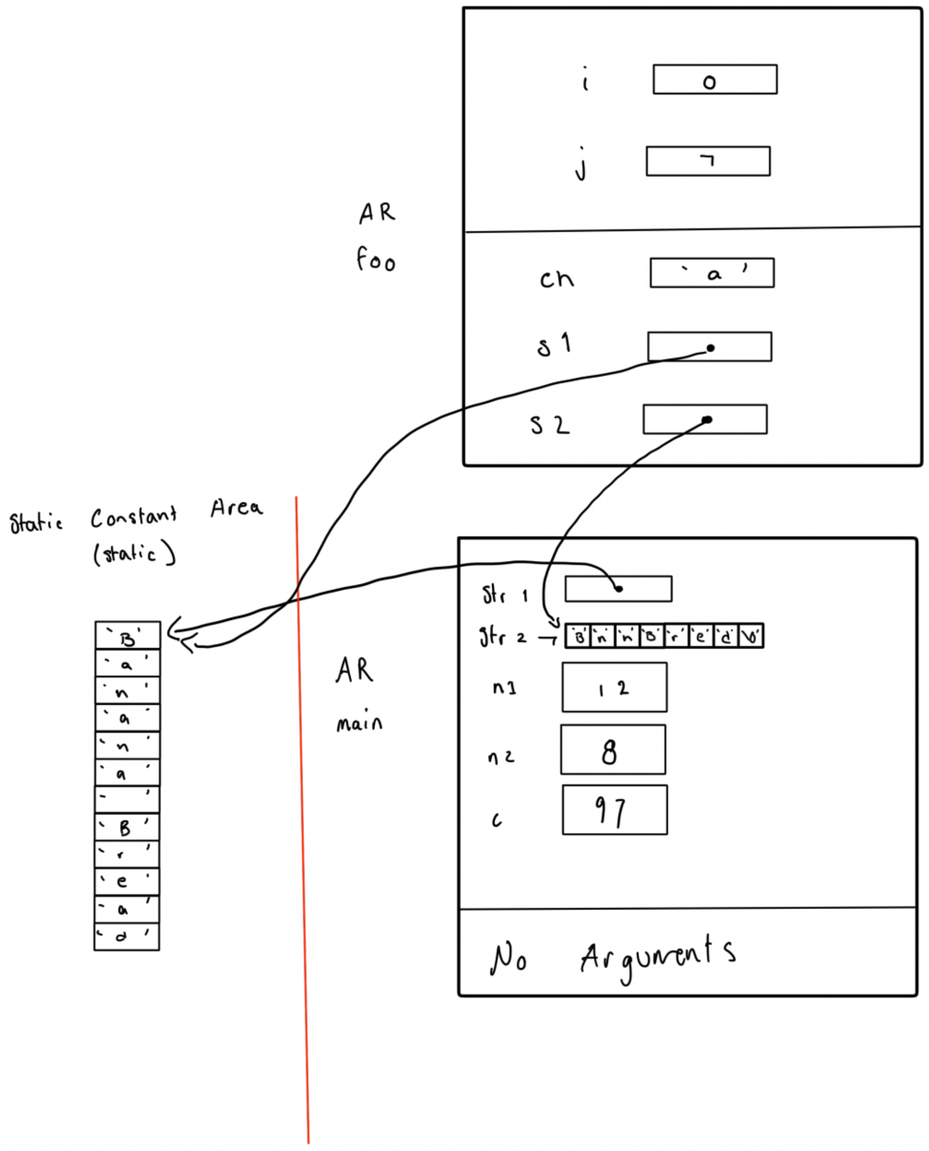
Exercise C

Stack

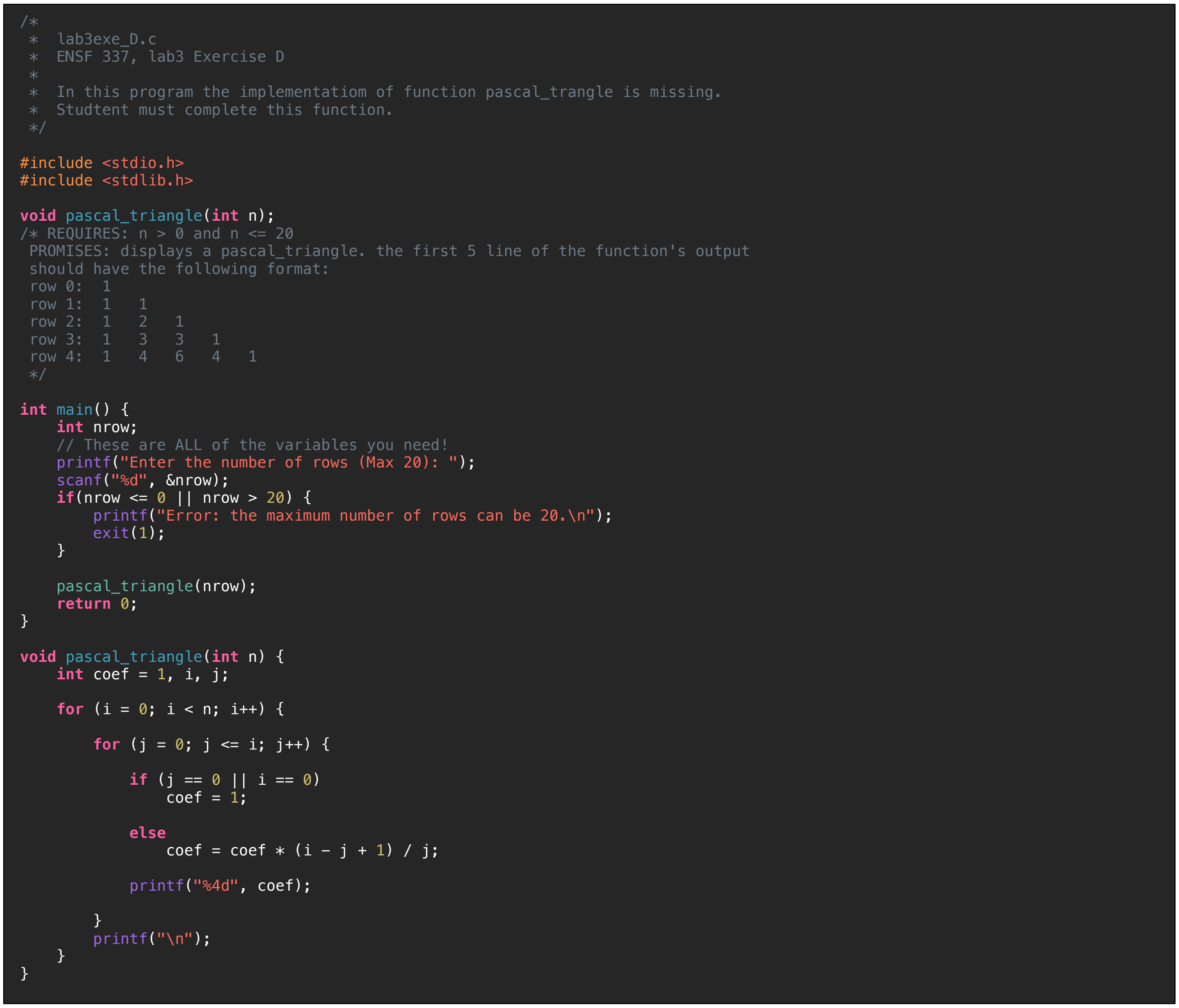


Stack

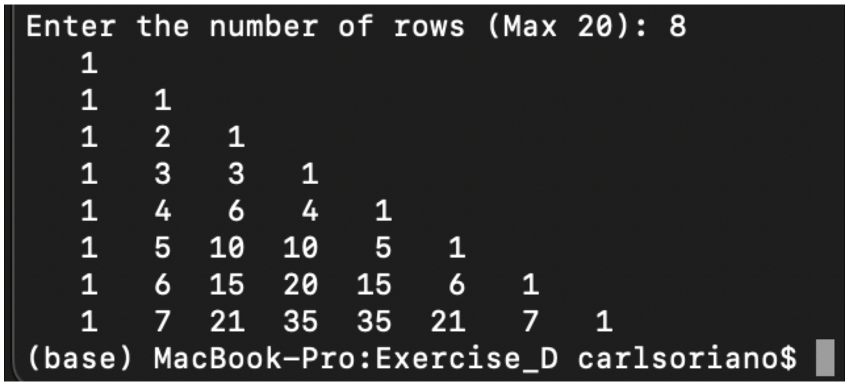
Point Two



Exercise D

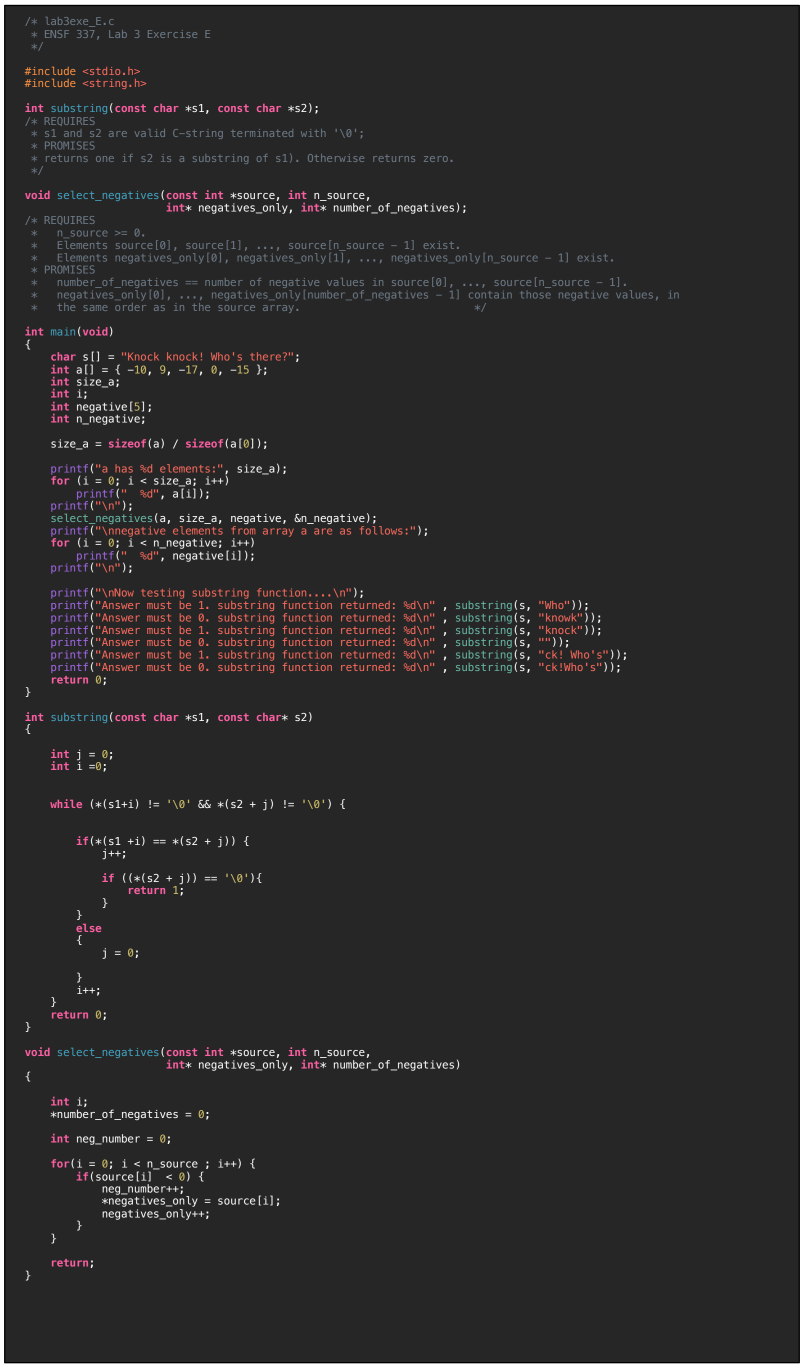


```c
/*
 *  lab3exe_D.c
 *  ENSF 337, lab3 Exercise D
 *
 *  In this program the implementatiom of function pascal_trangle is missing.
 *  Studtent must complete this function.
 */

#include <stdio.h>
#include <stdlib.h>

void pascal_triangle(int n);
/* REQUIRES: n > 0 and n <= 20
 PROMISES: displays a pascal_triangle. the first 5 line of the function's output
 should have the following format:
 row 0:  1
 row 1:  1   1
 row 2:  1   2   1
 row 3:  1   3   3   1
 row 4:  1   4   6   4   1
 */

int main() {
    int nrow;
    // These are ALL of the variables you need!
    printf("Enter the number of rows (Max 20): ");
    scanf("%d", &nrow);
    if(nrow <= 0 || nrow > 20) {
        printf("Error: the maximum number of rows can be 20.\n");
        exit(1);
    }

    pascal_triangle(nrow);
    return 0;
}

void pascal_triangle(int n) {
    int coef = 1, i, j;

    for (i = 0; i < n; i++) {

        for (j = 0; j <= i; j++) {

            if (j == 0 || i == 0)
                coef = 1;

            else
                coef = coef * (i - j + 1) / j;

            printf("%4d", coef);

        }
        printf("\n");
    }
}
```

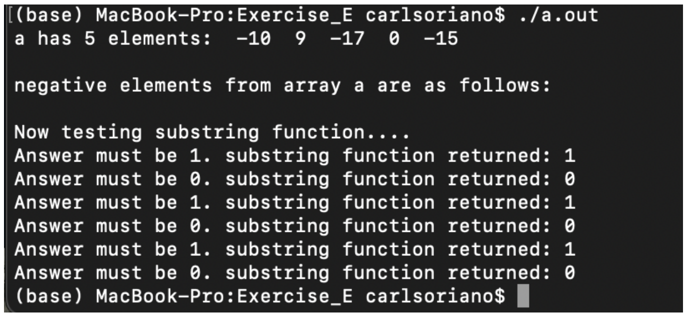


```
Enter the number of rows (Max 20): 8
   1
   1   1
   1   2   1
   1   3   3   1
   1   4   6   4   1
   1   5  10  10   5   1
   1   6  15  20  15   6   1
   1   7  21  35  35  21   7   1
(base) MacBook-Pro:Exercise_D carlsoriano$ 
```

Exercise E



```c
/* lab3exe_E.c
 * ENSF 337, Lab 3 Exercise E
 */

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

int substring(const char *s1, const char *s2);
/* REQUIRES
 * s1 and s2 are valid C-string terminated with '\0';
 * PROMISES
 * returns one if s2 is a substring of s1). Otherwise returns zero.
 */

void select_negatives(const int *source, int n_source,
                      int* negatives_only, int* number_of_negatives);
/* REQUIRES
 *   n_source >= 0.
 *   Elements source[0], source[1], ..., source[n_source - 1] exist.
 *   Elements negatives_only[0], negatives_only[1], ..., negatives_only[n_source - 1] exist.
 * PROMISES
 *   number_of_negatives == number of negative values in source[0], ..., source[n_source - 1].
 *   negatives_only[0], ..., negatives_only[number_of_negatives - 1] contain those negative values, in
 *   the same order as in the source array.                           */

int main(void)
{
    char s[] = "Knock knock! Who's there?";
    int a[] = { -10, 9, -17, 0, -15 };
    int size_a;
    int i;
    int negative[5];
    int n_negative;

    size_a = sizeof(a) / sizeof(a[0]);

    printf("a has %d elements:", size_a);
    for (i = 0; i < size_a; i++)
        printf("  %d", a[i]);
    printf("\n");
    select_negatives(a, size_a, negative, &n_negative);
    printf("\nnegative elements from array a are as follows:");
    for (i = 0; i < n_negative; i++)
        printf("  %d", negative[i]);
    printf("\n");

    printf("\nNow testing substring function....\n");
    printf("Answer must be 1. substring function returned: %d\n" , substring(s, "Who"));
    printf("Answer must be 0. substring function returned: %d\n" , substring(s, "knowk"));
    printf("Answer must be 1. substring function returned: %d\n" , substring(s, "knock"));
    printf("Answer must be 0. substring function returned: %d\n" , substring(s, ""));
    printf("Answer must be 1. substring function returned: %d\n" , substring(s, "ck! Who's"));
    printf("Answer must be 0. substring function returned: %d\n" , substring(s, "ck!Who's"));
    return 0;
}

int substring(const char *s1, const char* s2)
{

    int j = 0;
    int i =0;


    while (*(s1+i) != '\0' && *(s2 + j) != '\0') {


        if(*(s1 +i) == *(s2 + j)) {
            j++;

            if ((*(s2 + j)) == '\0'){
                return 1;
            }
        }
        else
        {
            j = 0;

        }
        i++;
    }
    return 0;
}

void select_negatives(const int *source, int n_source,
                      int* negatives_only, int* number_of_negatives)
{

    int i;
    *number_of_negatives = 0;

    int neg_number = 0;

    for(i = 0; i < n_source ; i++) {
        if(source[i]  < 0) {
            neg_number++;
            *negatives_only = source[i];
            negatives_only++;
        }
    }

    return;
}
```



```
(base) MacBook-Pro:Exercise_E carlsoriano$ ./a.out
a has 5 elements:  -10  9  -17  0  -15

negative elements from array a are as follows:

Now testing substring function....
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 0
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 0
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 0
(base) MacBook-Pro:Exercise_E carlsoriano$
```

Exercise F

Incomplete, did not finish.