Programming Fundamentals – ENSF 337

Lab 3

M. Moussavi

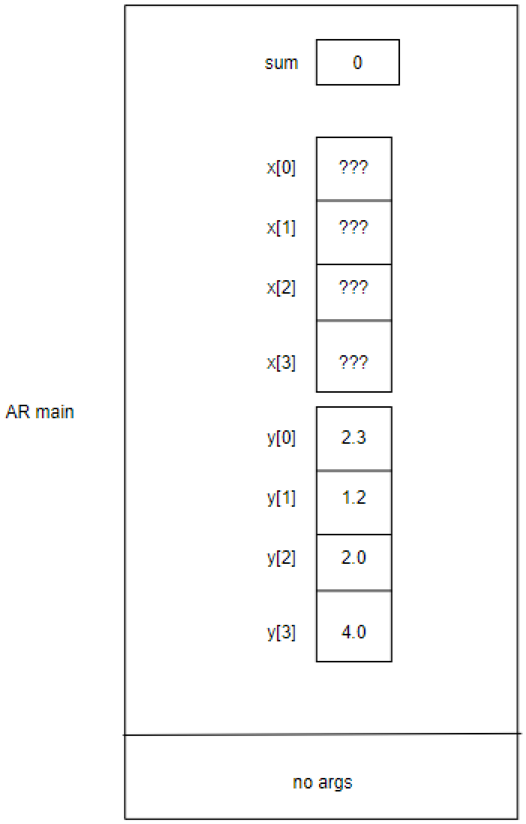
Jaxon Braun

B01

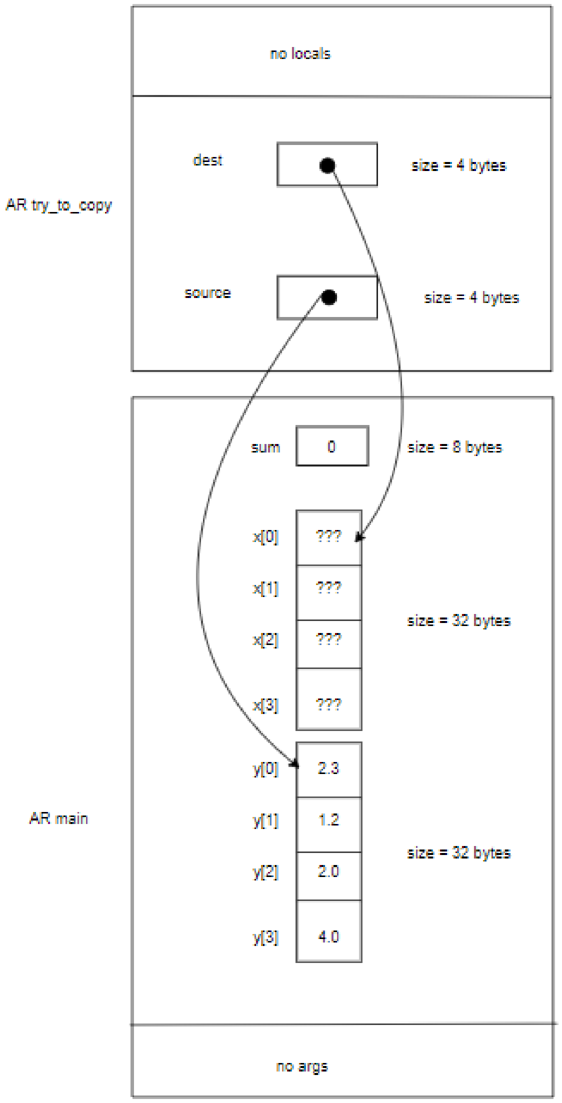
Submitted on October 13, 2021

Exercise A: Built in Arrays in C

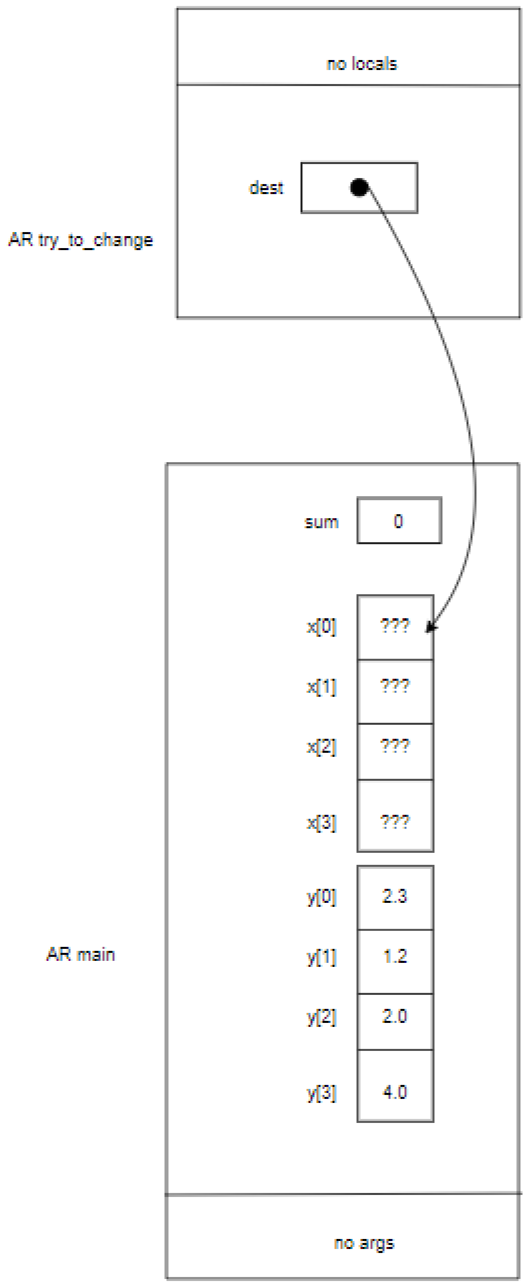
Point 1



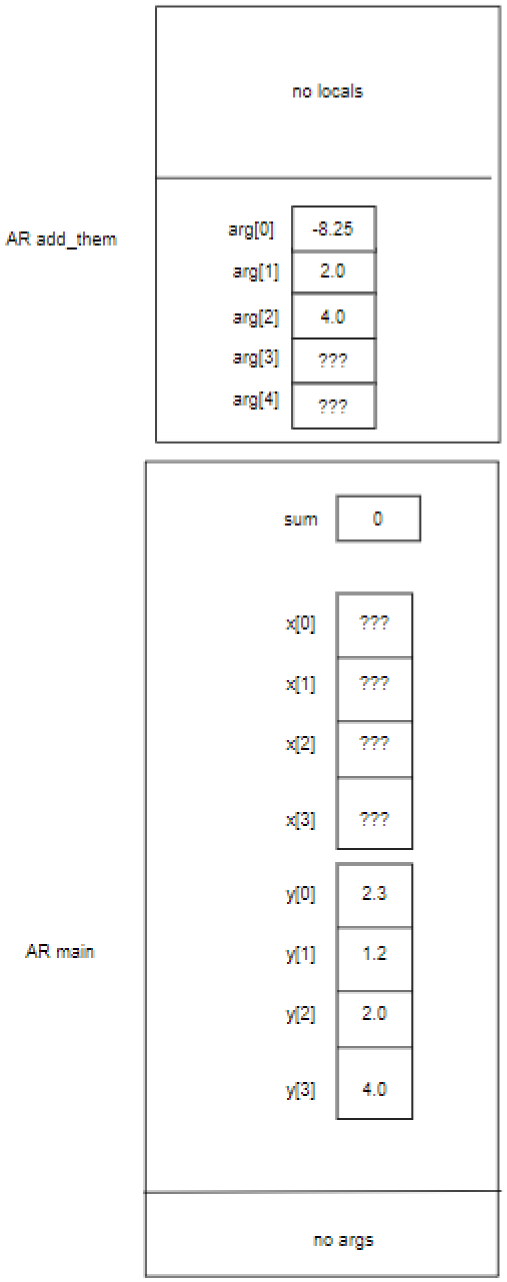
Point 2



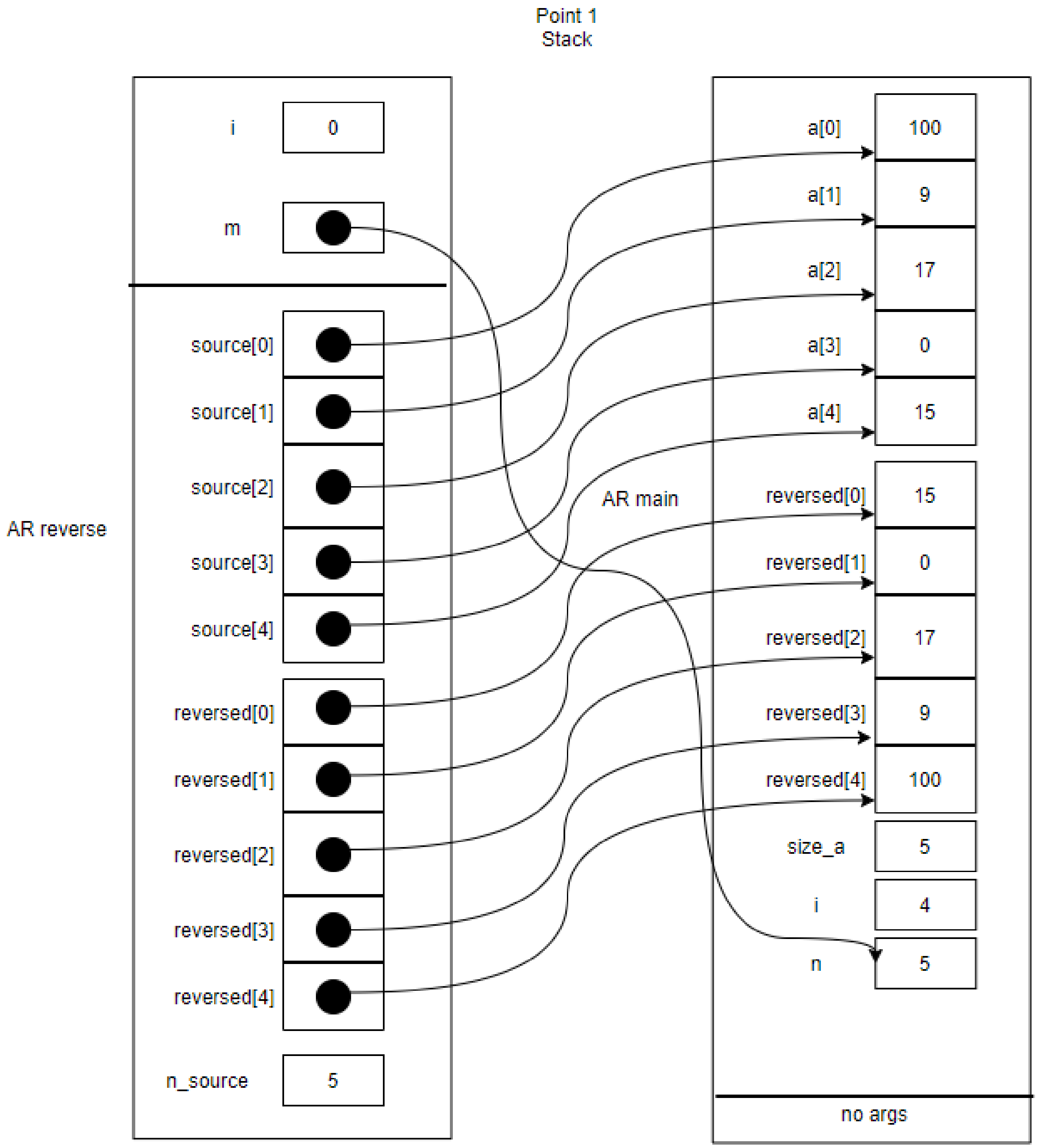
Point 3



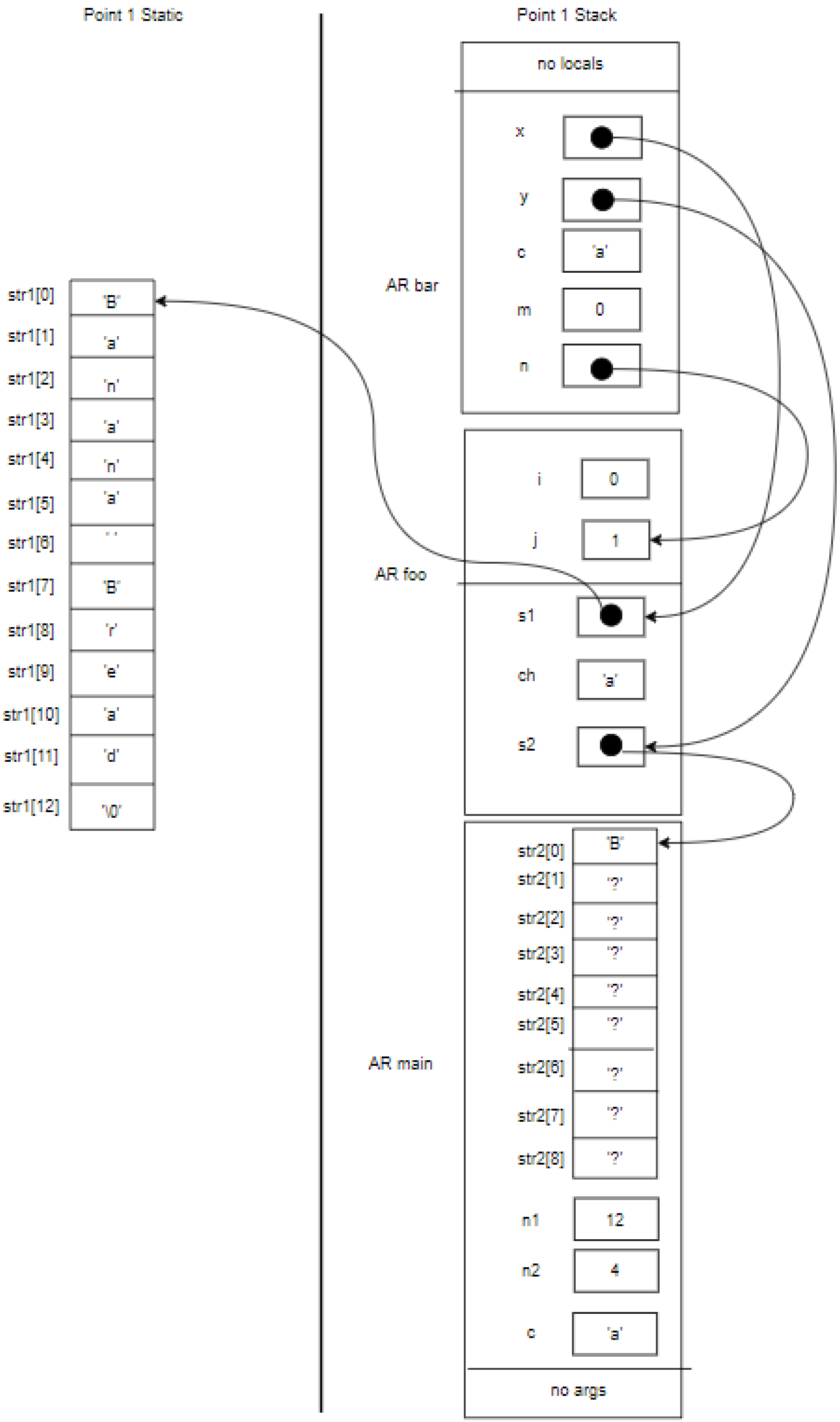
Point 4

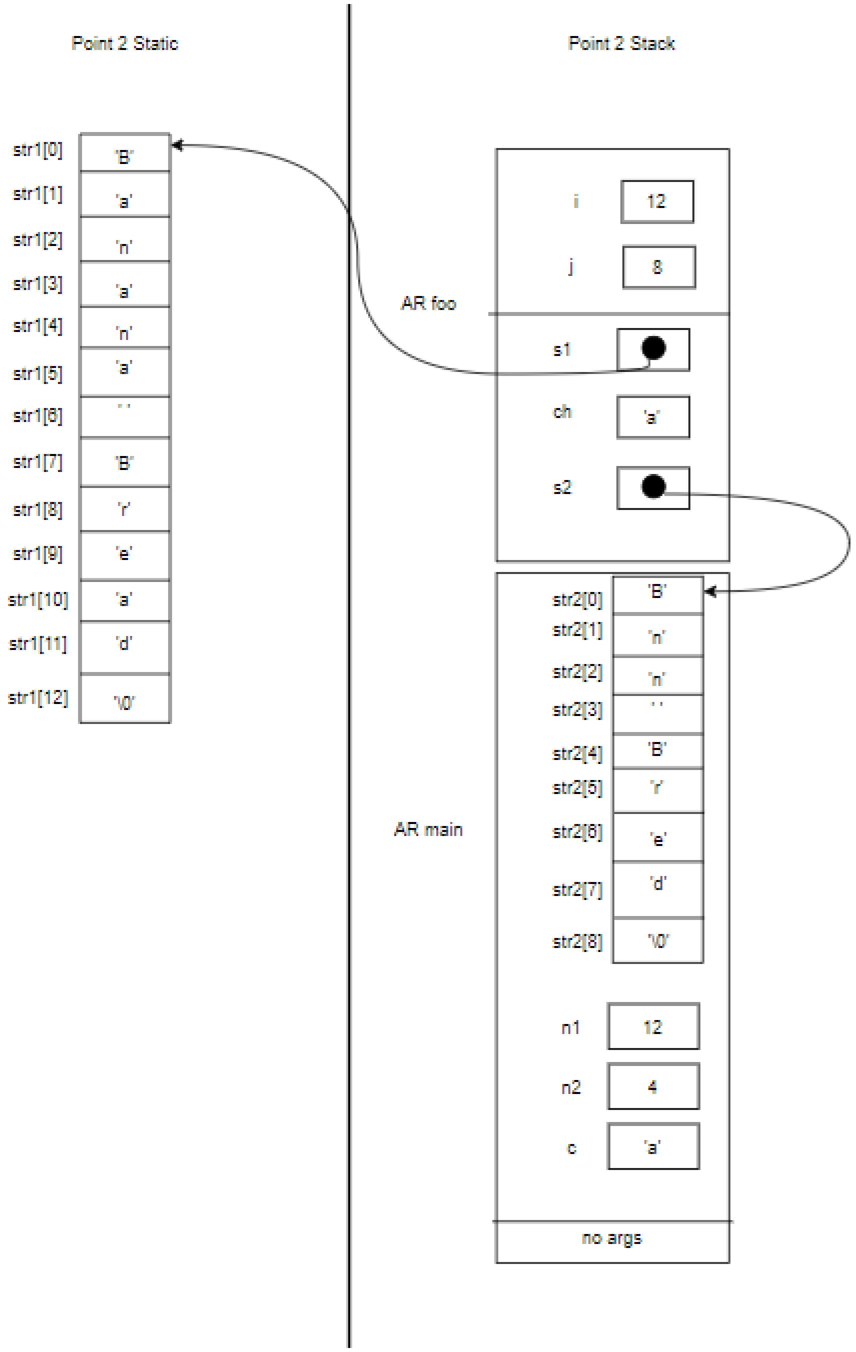


Exercise B: AR Diagrams with Arrays



Exercise C: AR Diagrams with C-String



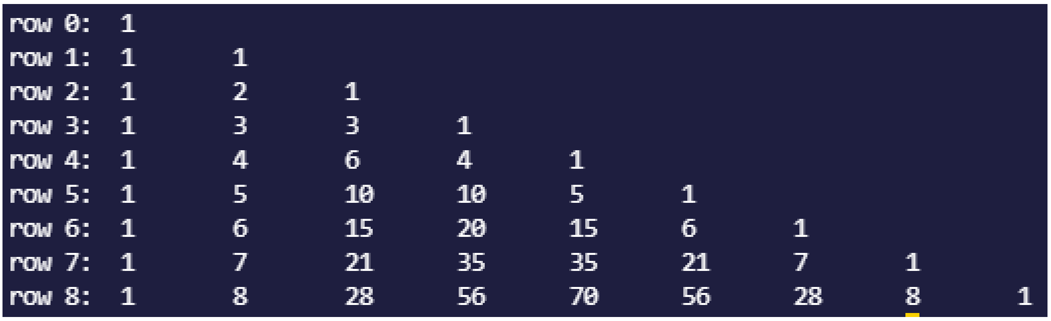


Point 2 Static

Point 2 Stack

Exercise D: Problem Solving

```
/*
 *  lab3exe_D.c
 *  ENSF 337, lab3 Exercise D
 *  Completed by: Jaxon Braun
 *  Submission Date: October 13, 2021
 *  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 previous_row[21];
    int current_row[21];
    for (int i = 0; i < n; i++){
        printf("row %d:  ", i);
        for (int j = 0; j <= i; j++){
            if (i == 1){
                current_row[0] = 1;
                current_row[1] = 1;
            }
            else if ((j == 0) || (j == i)){
                current_row[j] = 1;
            }
            else{
                current_row[j] = previous_row[j-1] + previous_row[j];
            }
        }
        for (int x = 0; x <= i; x++){
            printf("%d\t", current_row[x]);

            previous_row[x] = current_row[x];

        }

        printf("\n");

    }

    return;
```



```
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
row 5:  1       5       10      10      5       1
row 6:  1       6       15      20      15      6       1
row 7:  1       7       21      35      35      21      7       1
row 8:  1       8       28      56      70      56      28      8       1
```

```
}
```

Exercise E: Writing Functions that Work with Arrays

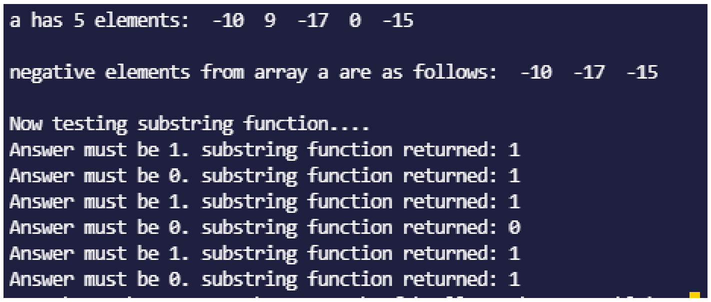
```
/* lab3exe_E.c
 * ENSF 337, Lab 3 Exercise E
 * Completed by: Jaxon Braun
 * Submission Date: October 13, 2021
 */

#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 s1_length = 0;
    int s2_length = 0;
    while (*(s1+s1_length) != '\0'){
        s1_length += 1;
    }
    while (*(s2+s2_length) != '\0'){
        s2_length += 1;
    }
    for (int i = 0; i < s1_length; i++){
        int j;
        int counter = 0;
        for (int j; j < s2_length; j++){
            if (j == counter){
                return 1;
            }
            if (*(s1+i) == *(s2+i+j)){
                counter++;
            }
        }
    }
    return 0;
}
```

```
void select_negatives(const int *source, int n_source, int* negatives_only, int*
number_of_negatives)
{
    int i;
    *number_of_negatives = 0;
    for (int i = 0; i < n_source; i++){
        if (source[i] < 0){
            negatives_only[*number_of_negatives] = source[i];
            *number_of_negatives += 1;
        }
    }
    return;
}
```

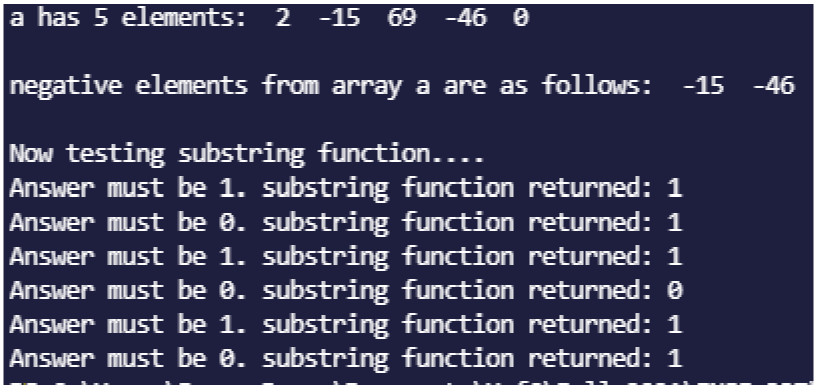
The select_negative function is fully operational, while the substring function is only partially functional

Output when a = { -10, 9, -17, 0, -15} and s = "Knock knock! Who's there?"



```
a has 5 elements:  -10  9  -17  0  -15

negative elements from array a are as follows:  -10  -17  -15

Now testing substring function....
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 1
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: 1
```

Output when a = {2, -15, 69, -46, 0} and s = “Knock knowk!Who’s there?”



```
a has 5 elements:  2  -15  69  -46  0

negative elements from array a are as follows:  -15  -46

Now testing substring function....
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 1
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: 1
```

Exercise F: More Practice with Strings