Course: ENSF 337 - Fall 2020

Lab #: 3

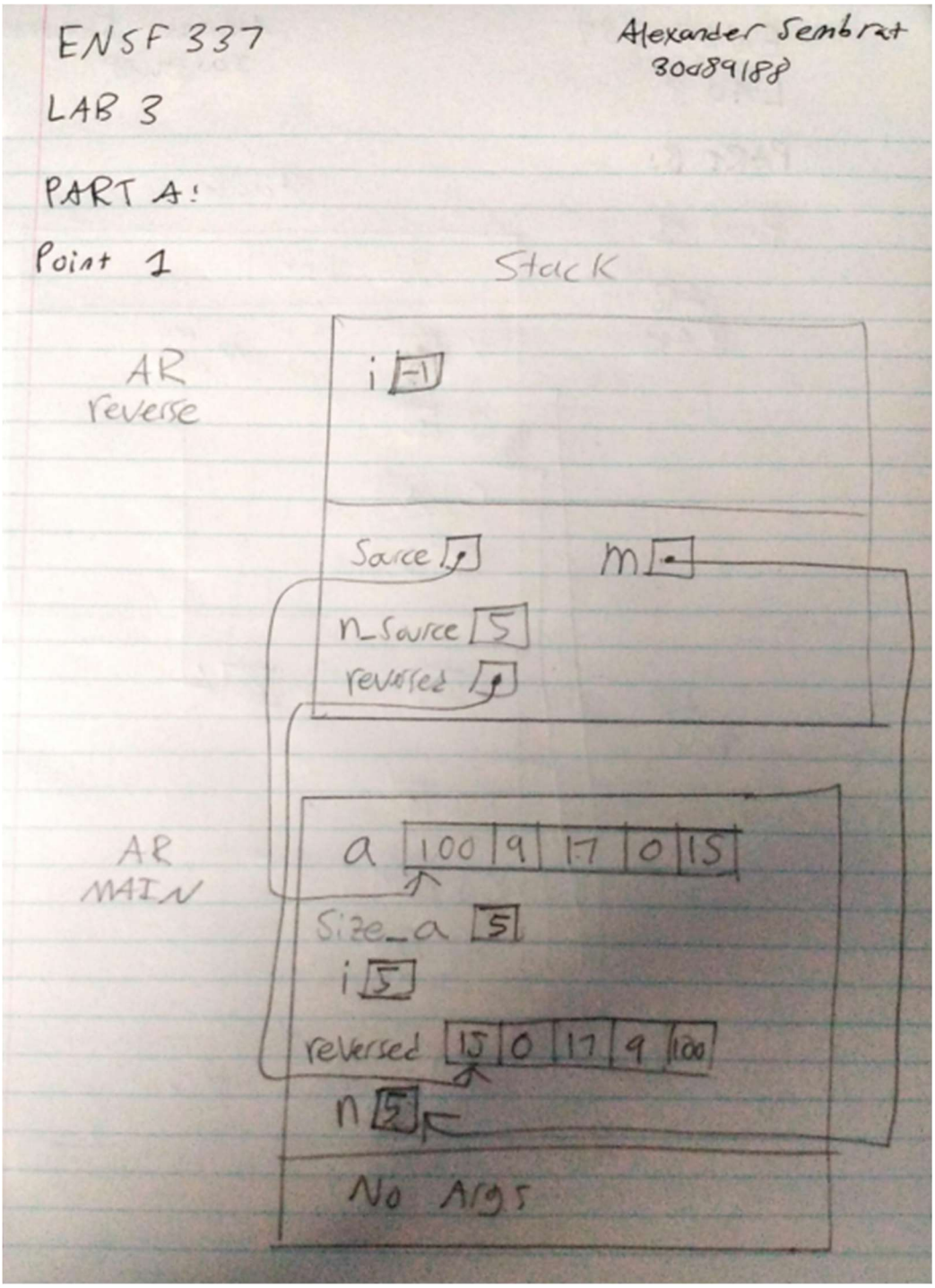
Instructor: M. Moussavi

Student Name: Alexander Sembrat (30089188)

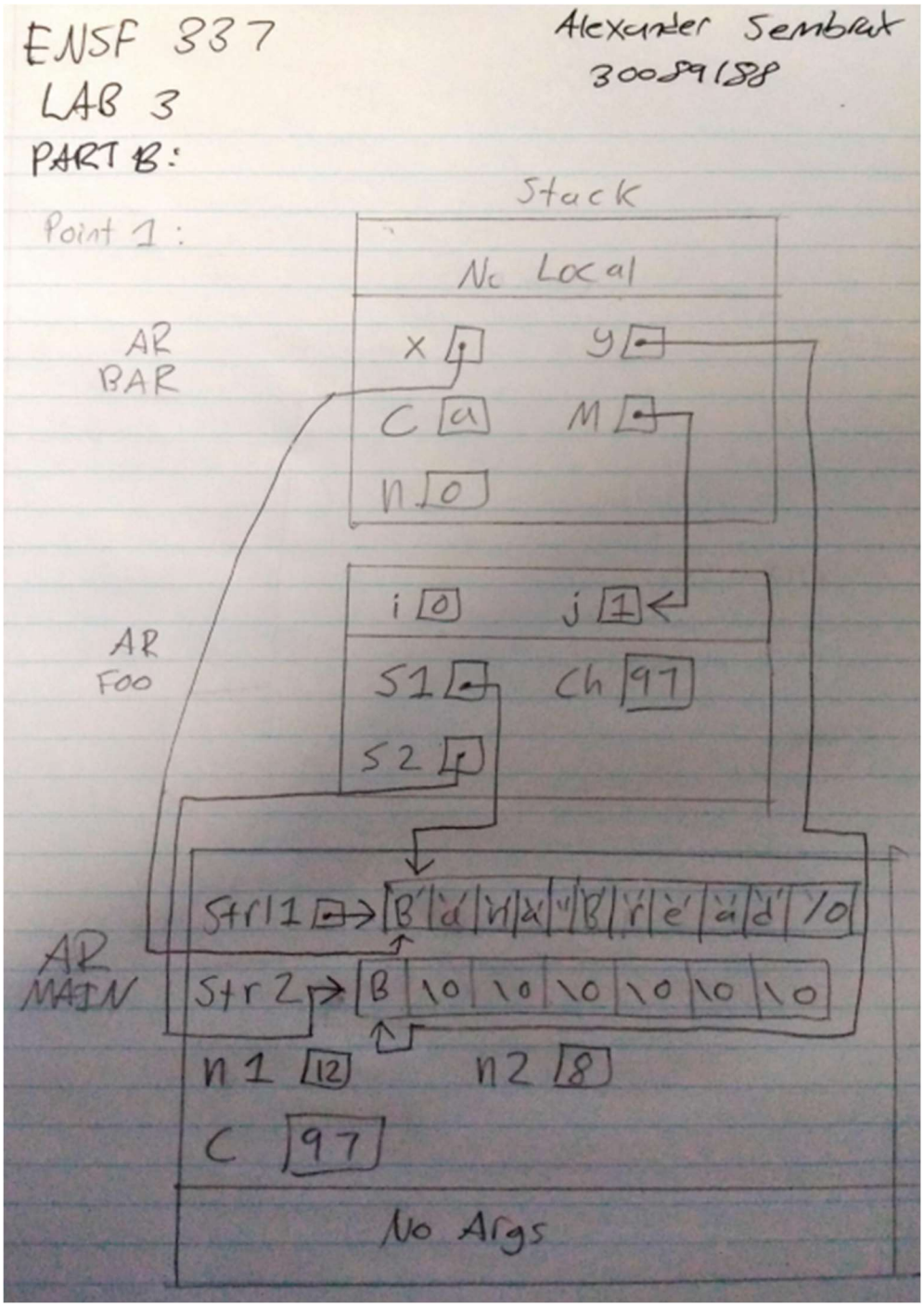
Lab Section: B01

Submission Date: October 05th 2020

Part A:


ENSF 337
Alexander Sembrat
30089188
LAB 3
PART A:
Point 1
Stack
AR
reverse
i -1
Source
m
n_source 5
reversed
AR
MAIN
a 100 9 17 0 15
size_a 5
i 5
reversed 15 0 17 9 100
n 5
No Args

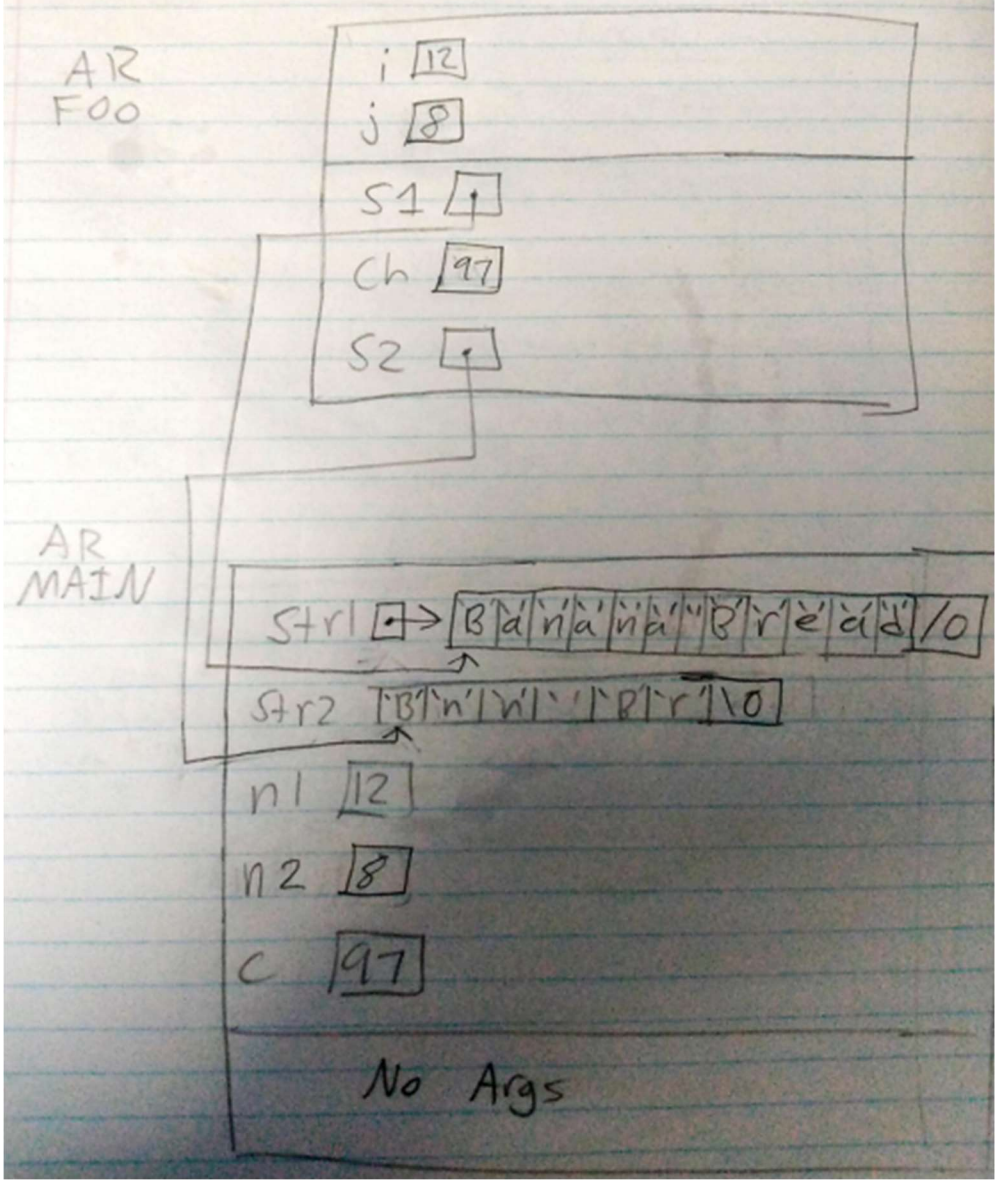
Part B:


ENSF 337
LAB 3
PART B:
Alexander Sembrat
30089188
Stack
Point 1:
No Local
AR
BAR
x
y
C a
M
n 0
i 0
j 1
AR
FOO
S1
Ch 97
S2
Str1
B à M a v B r è a d \0
AR
MAIN
Str2
B \0 \0 \0 \0 \0 \0
n1 12
n2 8
C 97
No Args

ENSF 337
LAB 3

Alexander Sembrat
30089188

Point 2



Part C:

```c
/*
*  File Name: lab3exe_C.c
*  Assignment: Lab 3 Exercise C
*  Lab Section: B01
*  Completed by: Alexander Sembrat (30089188)
*  Submission Date: Oct 05th 2020
*/
#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 i,j,x;
    printf("row 0: 1\n");
    for(int i = 1; i < n; i++)
    {
        printf("row %d:", i);
        for(int j = 0; j < i+1; j++)
        {
            if(j==0||i==0)
            {
                x = 1;
            }
            else
            {
                x = x*(i-j+1)/j;
            }
            printf(" %d", x);
        }
        printf("\n");
    }
}
```

```
Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/lab3
$ gcc -Wall lab3exe_c.c
lab3exe_c.c: In function 'pascal_triangle':
lab3exe_c.c:39:11: warning: unused variable 'j' [-Wunused-variable]
   39 |     int i,j,x;
      |           ^
lab3exe_c.c:39:9: warning: unused variable 'i' [-Wunused-variable]
   39 |     int i,j,x;
      |         ^

Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/lab3
$ ./a.exe
Enter the number of rows (Max 20): 9
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
```

Part D:

Original Code and Output

```c
/*
*  File Name: lab3exe_d.c
*  Assignment: Lab 3 Exercise D
*  Lab Section: B01
*  Completed by: Alexander Sembrat (30089188)
*  Submission Date: Oct 05th 2020
*/

#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"));
```

```c
    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)
{
    // This function is incomplete. Student must remove the next line and
    // complete this function...
    //printf ("\nFunction substring is incmplete and doesn't work.\n");
    int i,j,x;

    for(int i = 0; i < strlen(s1); i++){
        while(s1[i]==s2[j] && j<strlen(s2)){
            x++;
            i++;
            j++;
            if(x==strlen(s2)){
                return 1;
            }
        }
        x=0;
        j=0;
    }


    return 0;
}

void select_negatives(const int* source, int n_source, int* negatives_only, int* number_of_negatives)
{
    // This function is incomplete. Student must remove the next line and
    // complete this function...
    //printf ("\nFunction select_negatives is incmplete and doesn't work.\n");

    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)++;
        }
    }

    return;
}
```

```
Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/lab3
$ gcc -Wall lab3exe_d.c
lab3exe_d.c: In function 'substring':
lab3exe_d.c:65:9: warning: unused variable 'i' [-Wunused-variable]
   65 |     int i,j,x;
      |         ^
lab3exe_d.c: In function 'select_negatives':
lab3exe_d.c:90:9: warning: unused variable 'i' [-Wunused-variable]
   90 |     int i;
      |         ^

Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/lab3
$ ./a.exe
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: 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
```

Changed Code and Output (To show the code works for other strings/numbers)

```c
/*
*  File Name: lab3exe_d.c
*  Assignment: Lab 3 Exercise D
*  Lab Section: B01
*  Completed by: Alexander Sembrat (30089188)
*  Submission Date: Oct 05th 2020
*/

#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[] = "Hello there. General Kenobi."; //Originally "Knock knock! Who's there?"
    int a[] = { 24,-55, -10, 0, 43 }; //originally {-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, "Hello")); //originally "Who"
    printf("Answer must be 0. substring function returned: %d\n" , substring(s, "Kenowbi")); //originally "knowk"
    printf("Answer must be 1. substring function returned: %d\n" , substring(s, "Kenobi")); //originally "knock"
    printf("Answer must be 0. substring function returned: %d\n" , substring(s, "")); //originally ""
    printf("Answer must be 1. substring function returned: %d\n" , substring(s, "ere. General")); //originally "ck! Who's"
```

```c
    printf("Answer must be 0. substring function returned: %d\n" , substring(s, "ere.General")); //originally "ck!Who's"
    return 0;
}

int substring(const char* s1, const char* s2)
{
    // This function is incomplete. Student must remove the next line and
    // complete this function...
    //printf ("\nFunction substring is incmplete and doesn't work.\n");
    int i,j,x;

    for(int i = 0; i < strlen(s1); i++){
        while(s1[i]==s2[j] && j<strlen(s2)){
            x++;
            i++;
            j++;
            if(x==strlen(s2)){
                return 1;
            }
        }
        x=0;
        j=0;
    }


    return 0;
}

void select_negatives(const int* source, int n_source, int* negatives_only, int* number_of_negatives)
{
    // This function is incomplete. Student must remove the next line and
    // complete this function...
    //printf ("\nFunction select_negatives is incmplete and doesn't work.\n");

    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)++;
        }
    }

    return;
}
```

```
Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/lab3
$ gcc -Wall lab3exe_d.c
lab3exe_d.c: In function 'substring':
lab3exe_d.c:65:9: warning: unused variable 'i' [-Wunused-variable]
   65 |     int i,j,x;
      |         ^
lab3exe_d.c: In function 'select_negatives':
lab3exe_d.c:90:9: warning: unused variable 'i' [-Wunused-variable]
   90 |     int i;
      |         ^

Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/lab3
$ ./a.exe
a has 5 elements:  24  -55  -10  0  43

negative elements from array a are as follows:  -55  -10

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
```