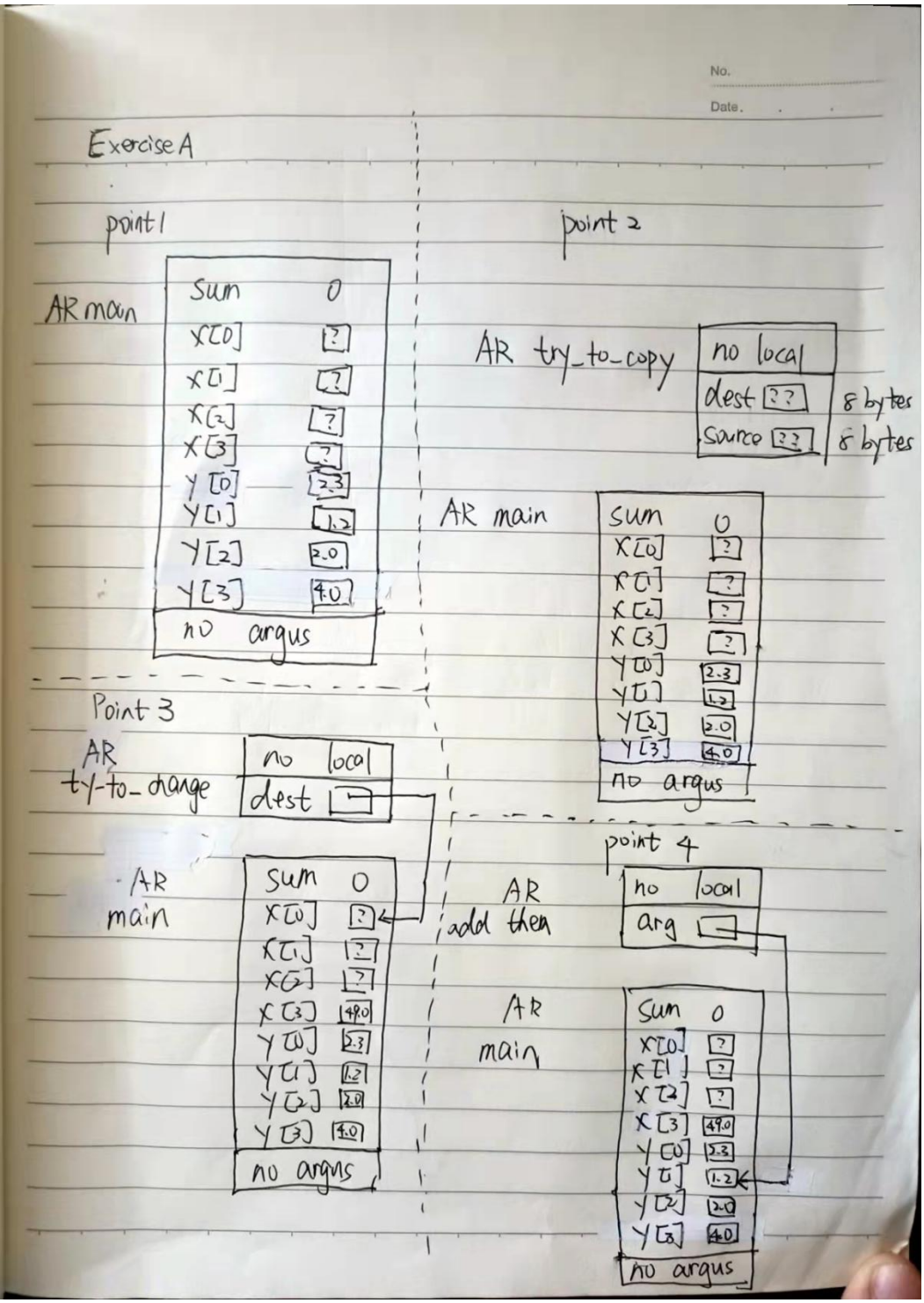
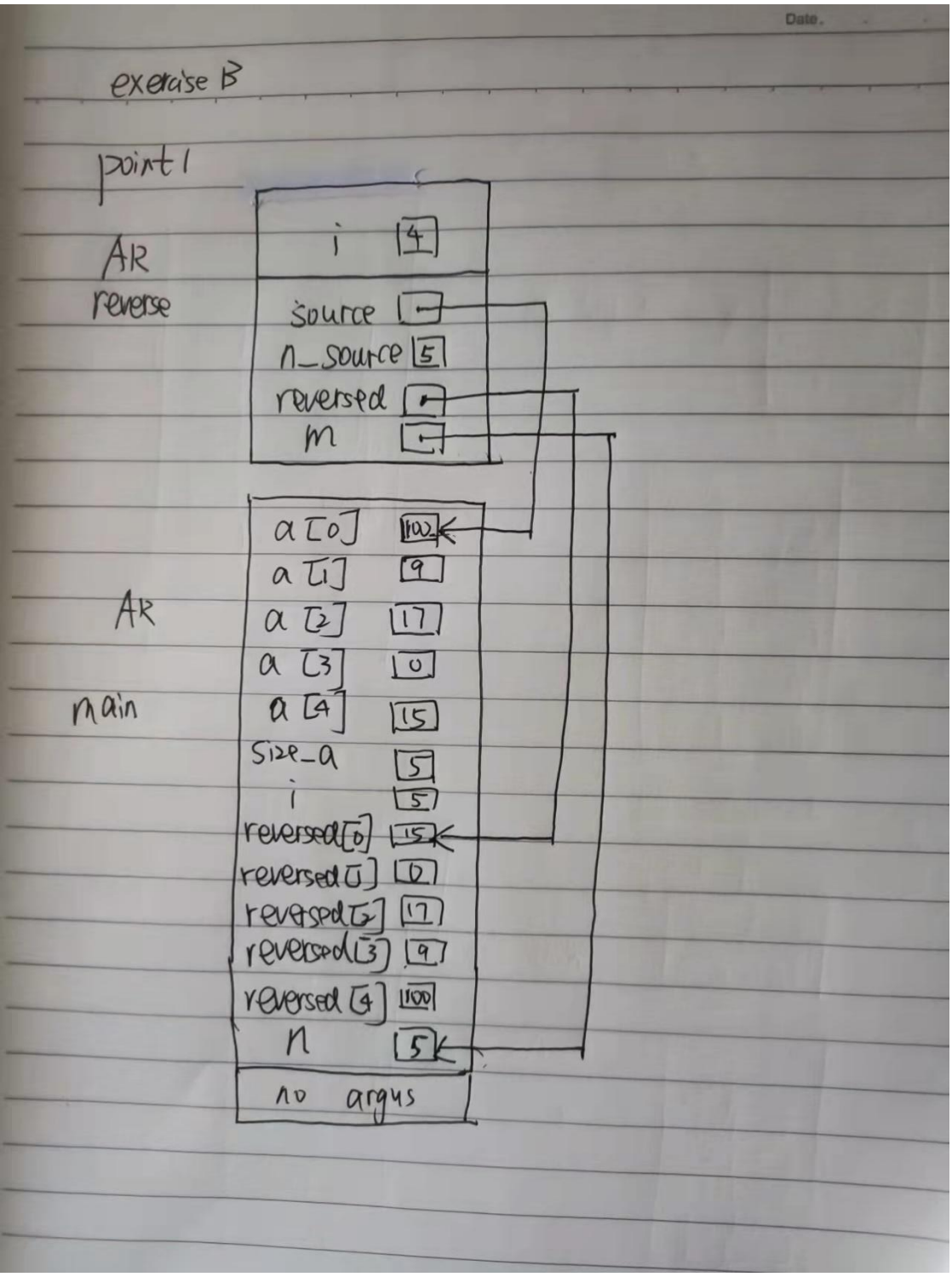
Course: Programming Fundamental – ENSF337
Lab #: Lab 2
Instructor: M. Moussavi
Student Name: Shanzi Ye
Lab Section: B01
Date submitted: May 17, 2022

Exercise A

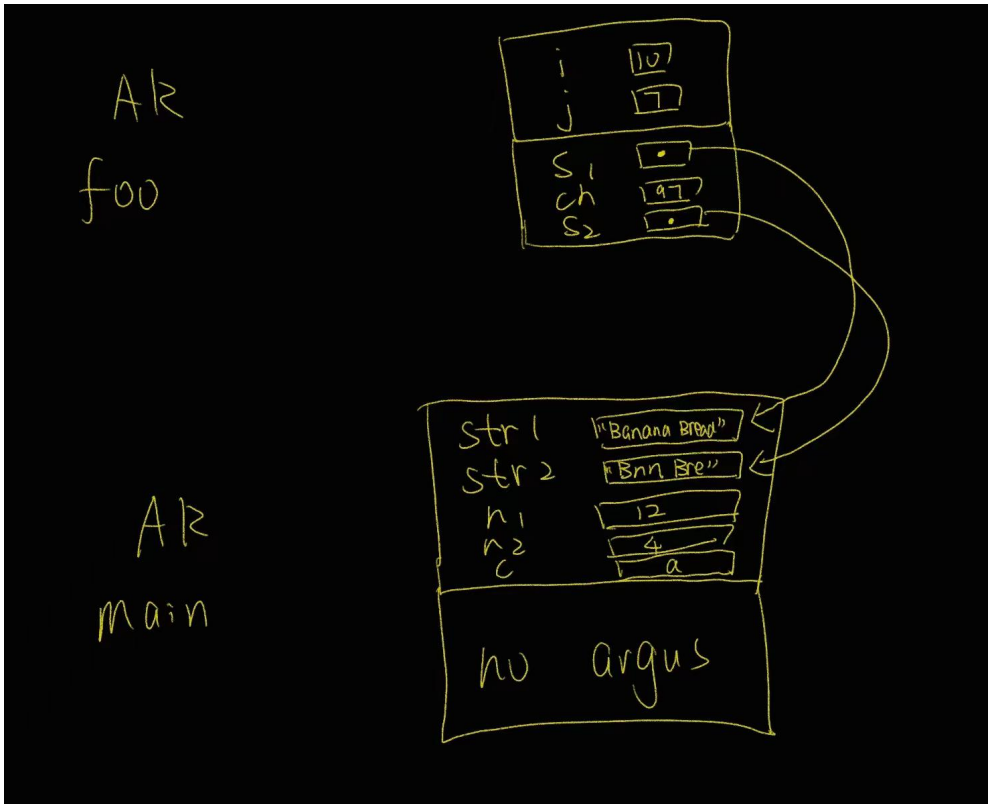

No.
Date.
Exercise A
point 1
AR main
sum 0
x[0] ?
x[1] ?
x[2] ?
x[3] ?
y[0] 2.3
y[1] 1.2
y[2] 2.0
y[3] 4.0
no argus
point 2
AR try_to_copy
no local
dest ??
source ??
8 bytes
8 bytes
AR main
sum 0
x[0] ?
x[1] ?
x[2] ?
x[3] ?
y[0] 2.3
y[1] 1.2
y[2] 2.0
y[3] 4.0
no argus
Point 3
AR try-to-change
no local
dest
AR main
sum 0
x[0] ?
x[1] ?
x[2] ?
x[3] 49.0
y[0] 2.3
y[1] 1.2
y[2] 2.0
y[3] 4.0
no argus
point 4
AR add then
no local
arg
AR main
sum 0
x[0] ?
x[1] ?
x[2] ?
x[3] 49.0
y[0] 2.3
y[1] 1.2
y[2] 2.0
y[3] 4.0
no argus

Exercise B

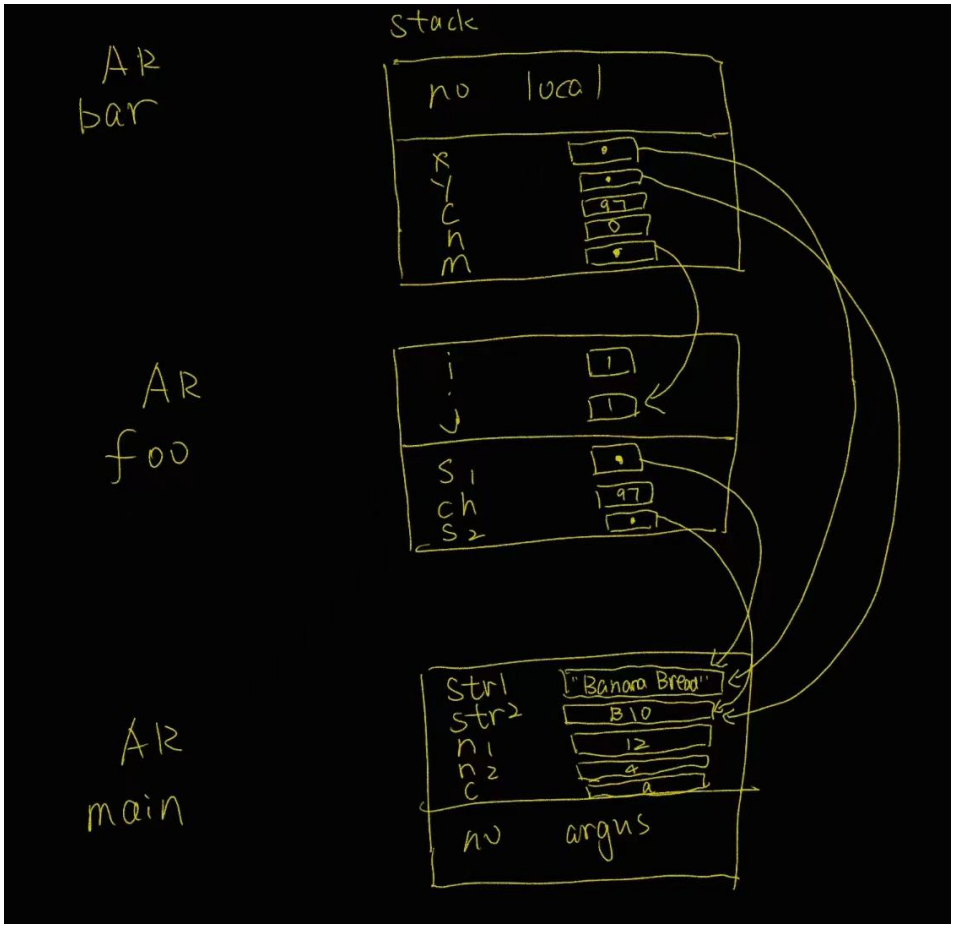

exercise B
point 1
AR
reverse
i
4
source
n_source
5
reversed
m
AR
main
a[0]
100
a[1]
9
a[2]
17
a[3]
0
a[4]
15
size_a
5
i
5
reversed[0]
15
reversed[1]
0
reversed[2]
17
reversed[3]
9
reversed[4]
100
n
5
no argus

Exercise C

Point 2:



Point 1:



Exercise D

```
/*
* File Name: lab2_exe_D.c
* Assignment: Lab 2 Exercise D
* Lab section: (B01)
* Completed by: Shanzi Ye
* Submission Date: May 17, 2022
*/

#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) {
        // STUDENTS MUST COMPLETE THE REST OF IMPLEMENATION OF THIS FUNCTION
    int current_array[20] = { 1,0 };
    int iLayer,iCol;
    int iPrevLValue, iPrevRValue;
    for (iLayer = 0; iLayer < n; iLayer++)
    {
                printf("row:%-4d", iLayer);
                iPrevLValue = 0;
                for (iCol = 0; iCol <= iLayer; iCol++)
                {
                        iPrevRValue = current_array[iCol];
                        current_array[iCol] = iPrevLValue + iPrevRValue;
```

```
            printf("%-9d", current_array[iCol]);
            iPrevLValue = iPrevRValue;
        }
        printf("\n");
    }
}
```

```
jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$ ./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

jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$
```

Exercise E

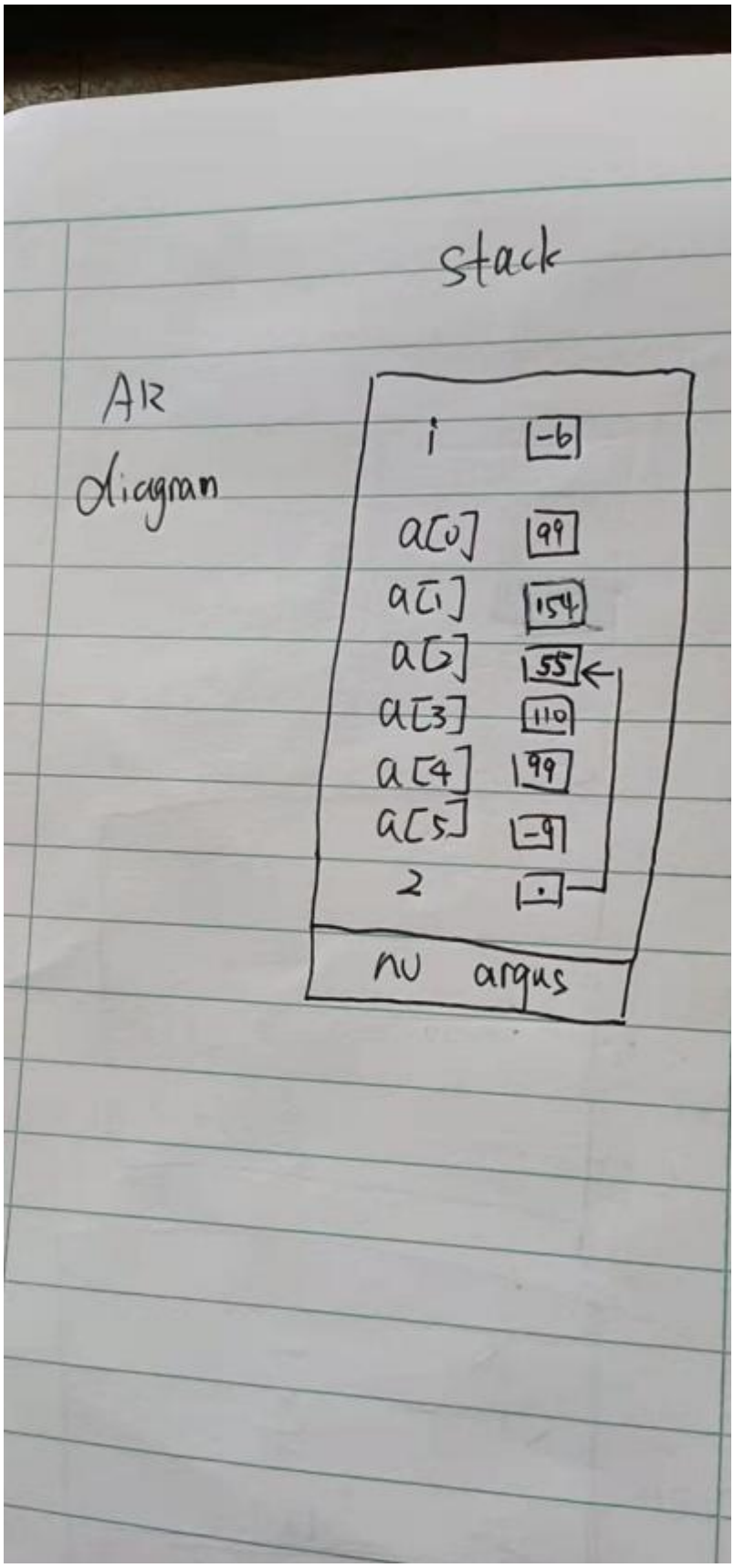
```
        /*
* File Name: lab2_exe_E.c
* Assignment: Lab 2 Exercise E
* Lab section: (B01)
* Completed by: Shanzi Ye
* Submission Date: May,17,2022
*/

#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 i = 0;
    int j;
    int m;
    int flag = 0;
    while (s1[i] != '\0')
    {
            j = 0;
          if (s2[0] == s1[i])
          {
                m = i;
                while (s2[j] != '\0')
                {
                     if (s1[m] == s2[j])
                     {
                              flag = 1;
                     }
                     else
                     {
                             flag = 0;
                             break;
                       }
                       m++;
                       j++;

                  }

                if (flag == 1)
                {
                         return 1;
                }
            }
            i++;
        }

    return 0;
}

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

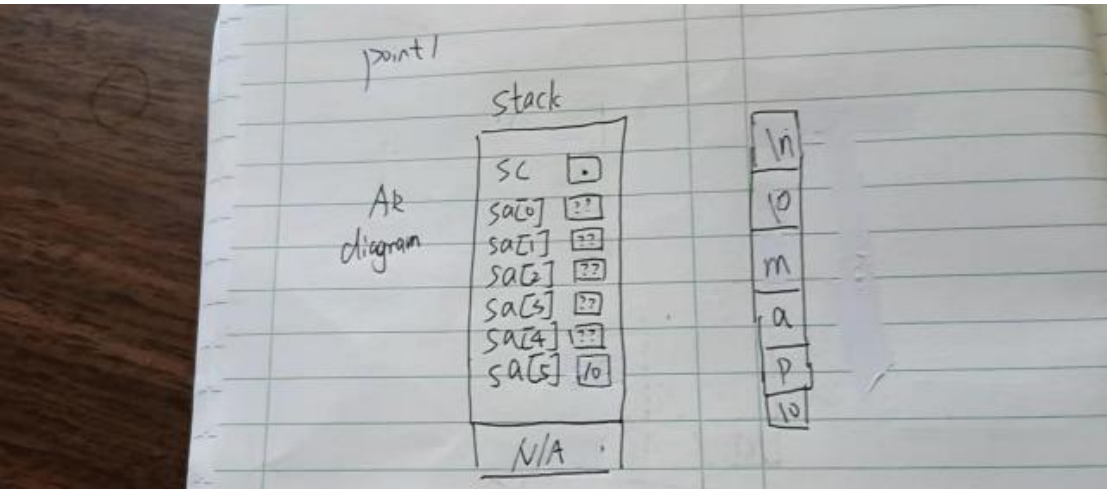
```
    int i ;
    *number_of_negatives = 0;
    int    j = 0;

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

    return;
}
```

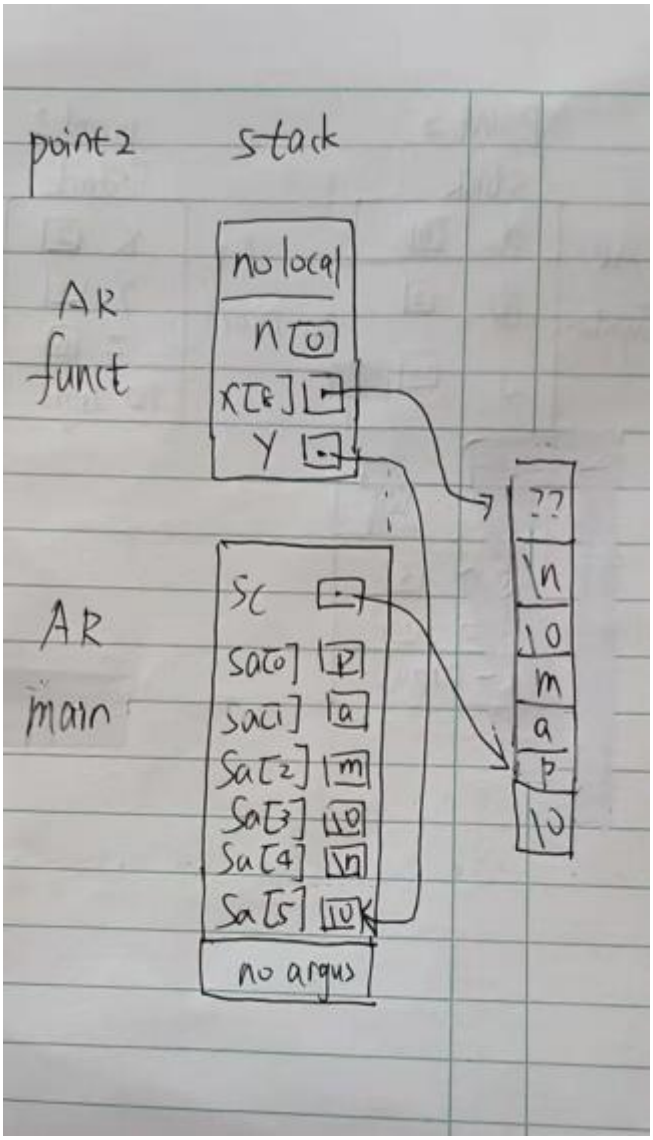
```
jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$ ./a.exe
a has 5 elements:  -10  9  -17  0  -15

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

jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$
```

Exercise G



Exercise H


point1
stack
AR
diagram
sc
sa[0] ??
sa[1] ??
sa[2] ??
sa[3] ??
sa[4] ??
sa[5] \0
N/A
\n
\0
m
a
p
\0


point2
stack
AR
funct
no local
n 0
x[0]
y
AR
main
sc
sa[0] p
sa[1] a
sa[2] m
sa[3] \0
sa[4] \n
sa[5] \0
no argus
??
\n
\0
m
a
p
\0

Exercise 1

```
/*
*File Name: lab2_exe_I.c
*Assignment: Lab 2 Exercise I
*Lab section: (B01)
*Completed by: Shanzi Ye
*Submission Date: May 17, 2022
*/

#include <stdio.h>
#define ELEMENTS(arrayname)   sizeof(arrayname)/sizeof(arrayname[0])
int main()
{

int size;
int a[] = {45, 67, 89, 24, 54};
double b[20] = {14.5, 61.7, 18.9, 2.4, 0.54};
size = ELEMENTS(a);

printf("Array a has 5 elements and macro ELEMENTS returns %d\n", size);
size = ELEMENTS(b);

printf("Array b has 20 elements and macro ELEMENTS returns %d\n", size);

return 0;
}
```

```
jackye@LAPTOP-BRGO8KVA /cygdrive/d/ENSF337_SPRING
$ ./a.exe
Array a has 5 elements and macro ELEMENTS returns 5
Array b has 20 elements and macro ELEMENTS returns 20

jackye@LAPTOP-BRGO8KVA /cygdrive/d/ENSF337_SPRING
$
```

Exercise J

```
/*
* File Name: lab2_exe_J.c
* Assignment: Lab 2 Exercise J
* Lab section: (B01)
* Completed by: Shanzi Ye
* Submission Date: May 17, 2021
*/


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

int    my_strlen(const char *s);
```

```
/*    Duplicates my_strlen from <string.h>, except return type is int.
  *    REQUIRES
  *         s points to the beginning of a string.
  *    PROMISES
  *         Returns the number of chars in the string, not including the
  *         terminating null.
  */

int    my_strlen(const char *s)
{
        int len =0;
        while(1)
        {
                if    (*s++ == '\0')
                {
                        return len;
                }
                len++;
        }
}

void my_strncat(char *dest, const char *source, int n);
/*    Duplicates my_strncat from <string.h>, except return type is void.
  *      dest and source point to the beginning of two strings.
  *    PROMISES
  *         appends source to the end of dest. If length of source is more than n.
  *         Only copies the first n elements of source.
  */

void my_strncat(char *dest, const char *source, int n)
{
        int count ;
        while (*dest != '\0') dest++;


        for(count =0;count <n;count++)
        {
                *dest = *source;
                dest++;
                source++;
        }

        *dest = '\0';
```

```
}

int my_strncmp(const char* str1, const char* str2);
/*    Duplicates strcmp from <string.h>, except return type is int.
  *    REQUIRES
  *         str1 points to the beginning of a string, and str2 to the beginning of
  *         another string.
  *    PROMISES
  *         Returns 0 if str1 and str2 are idntical.
  *         Returns a negative number of str1 is less that str2.
  *         Return a psitive nubmer of str2 is less than str1.
  */

int main(void)
{
        char str1[7] = "banana";
        const char str2[] = "-tacit";
        const char* str3 = "-toe";

        char str5[] = "ticket";
        char my_string[100]="";
        int bytes;
        int length;
        int y;

        printf("\nTESTING my_strlen FUNCTION ... \n");

        /* using my_strlen function */
        length = (int) my_strlen(my_string);
        printf("\nExpected to display: my_string length is 0.");
        printf("\nmy_string length is %d.", length);

        /* using sizeof operator */
        bytes = sizeof (my_string);
        printf("\nExpected to display: my_string size is 100 bytes.");
        printf("\nmy_string size is %d bytes.", bytes);

        /* using strcpy C libarary function */
        strcpy(my_string, str1);
        printf("\nExpected to display: my_string contains banana.");
        printf("\nmy_string contains %s", my_string);

        length = (int) my_strlen(my_string);
        printf("\nExpected to display: my_string length is 6.");
```

```
printf("\nmy_string length is %d.", length);

my_string[0] = '\0';
printf("\nExpected to display: my_string contains \"\".");
printf("\nmy_string contains:\"%s\"", my_string);

length = (int) my_strlen(my_string);
printf("\nExpected to display: my_string length is 0.");
printf("\nmy_string length is %d.", length);

bytes = sizeof (my_string);
printf("\nExpected to display: my_string size is still 100 bytes.");
printf("\nmy_string size is still %d bytes.", bytes);

printf("\n\nTESTING my_strncat FUNCTION ... \n");
/* my_strncat append the first 3 characters of str5 to the end of my_string */my_strncat(my_string,
str5, 3);
printf("\nExpected to display: my_string contains \"tic\"");
printf("\nmy_string contains \"%s\"", my_string);

length = (int) my_strlen(my_string);
printf("\nExpected to display: my_string length is 3.");
printf("\nmy_string length is %d.", length);

my_strncat(my_string, str2,    4);
printf("\nExpected to display: my_string contains \"tic-tac\"");
printf("\nmy_string contains:\"%s\"", my_string);

/* my_strncat append ONLY up ot '\0' character from str3 -- not 6 characters */
my_strncat(my_string, str3, 6);
printf("\nExpected to display: my_string contains \"tic-tac-toe\"");
printf("\nmy_string contains:\"%s\"", my_string);

length = (int) my_strlen(my_string);
printf("\nExpected to display: my_string has 11 characters.");
printf("\nmy_string has %d characters.", length);

printf("\n\nUsing strcmp - C library function: ");
printf("\nExpected to display: \"ABCD\" is less than \"ABCDE\"");
printf("\n\"ABCD\" is less than \"ABCDE\"", strcmp("ABCD", "ABCDE"));


printf("\n\nTESTING strcmp FUNCTION ... \n");
```

```
    if((y = strcmp("ABCD", "ABND")) < 0)
        printf("\n\"ABCD\" is less than \"ABND\" ... strcmp returns %d", y);

    if((y = strcmp("ABCD", "ABCD")) == 0)
        printf("\n\"ABCD\" is equal \"ABCD\" ... strcmp returns %d", y);

    if((y = strcmp("ABCD", "ABCd")) < 0)
        printf("\n\"ABCD\" is less than \"ABCd\" ... strcmp returns %d", y);

    if((y = strcmp("Orange", "Apple")) > 0)
        printf("\n\"Orange\" is greater than \"Apple\" ... strcmp returns %d\n", y);

    return 0;
}
```

```
jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$ ./a.exe

TESTING my_strlen FUNCTION ...

Expected to display: my_string length is 0.
my_string length is 0.
Expected to display: my_string size is 100 bytes.
my_string size is 100 bytes.
Expected to display: my_string contains banana.
my_string contains banana
Expected to display: my_string length is 6.
my_string length is 6.
Expected to display: my_string contains "".
my_string contains:""
Expected to display: my_string length is 0.
my_string length is 0.
Expected to display: my_string size is still 100 bytes.
my_string size is still 100 bytes.

TESTING my_strncat FUNCTION ...

Expected to display: my_string contains "tic"
my_string contains "tic"
Expected to display: my_string length is 3.
my_string length is 3.
Expected to display: my_string contains "tic-tac"
my_string contains:"tic-tac"
Expected to display: my_string contains "tic-tac-toe"
my_string contains:"tic-tac-toe"
Expected to display: my_string has 11 characters.
my_string has 11 characters.

Using strcmp - C library function:
Expected to display: "ABCD" is less than "ABCDE"
"ABCD" is less than "ABCDE"

TESTING strcmp FUNCTION ...

"ABCD" is less than "ABND" ... strcmp returns -1
"ABCD" is equal "ABCD" ... strcmp returns 0
"ABCD" is less than "ABCd" ... strcmp returns -1
"Orange" is greater than "Apple" ... strcmp returns 1

jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$
```