Course : Programming Fundamental – ENSF 337

Lab # : Lab 4

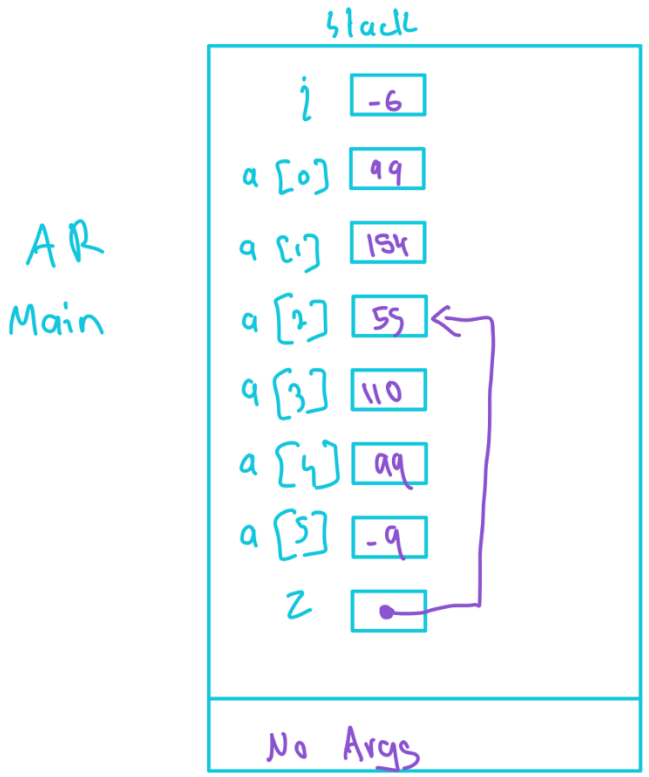
Instructor : M. Moussavi

Student Name : Nimna Wijedasa

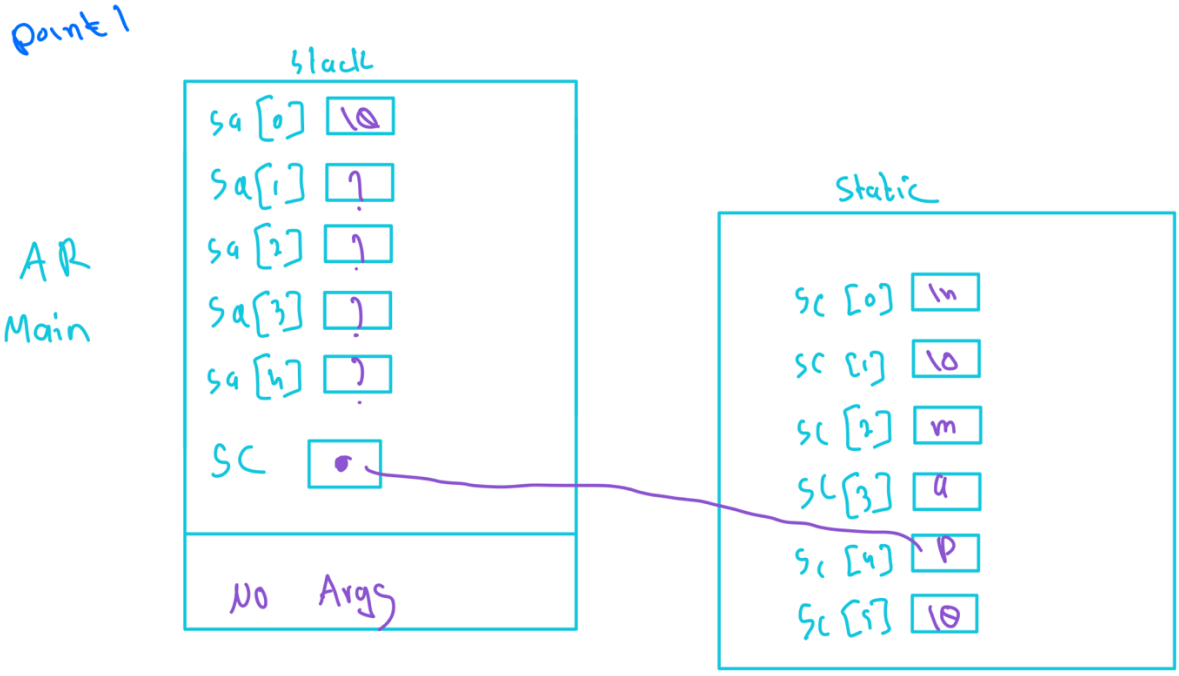
Lab Section : B02

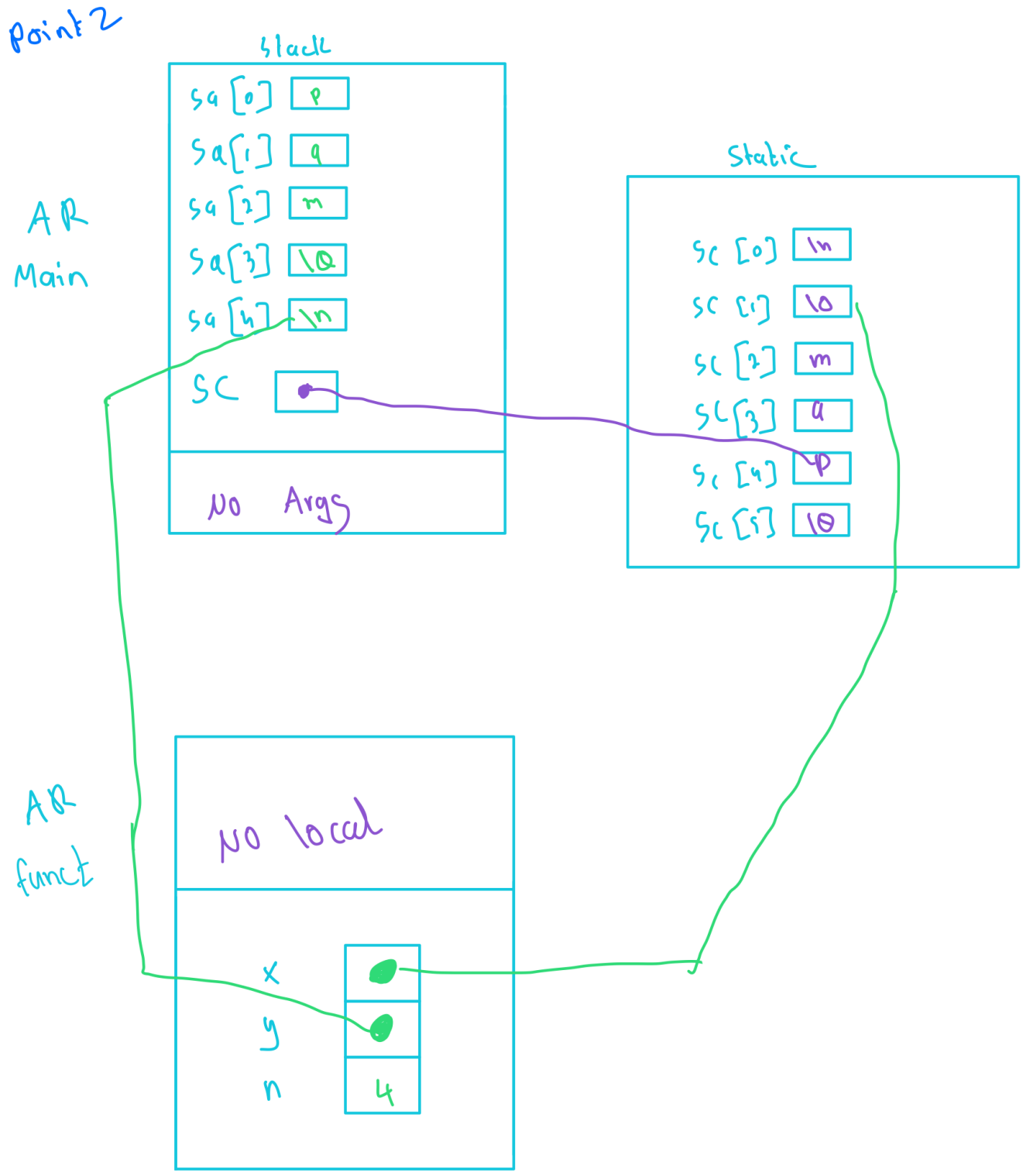
Date submitted : Oct 14, 2022

Exercise A



Exercise B




Point2
Stack
sa [0] p
sa [1] q
sa [2] m
sa [3] 10
sa [4] 1n
sc
No Args
AR
Main
Static
sc [0] 1n
sc [1] 10
sc [2] m
sc [3] q
sc [4] p
sc [5] 10
AR
funct
No local
x
y
n
4

Exercise C

```c
//  lab2exC.c
// ENSF 337 Lab 4 Exercise  C
//

#include <stdio.h>
#define ELEMENTS(x) (sizeof (x)/sizeof (x[0]))

int main()
{

    int size;
    int a[] = { [0]: 45,  [1]: 67,  [2]: 89,  [3]: 24,  [4]: 54};
    double b[20] = { [0]: 14.5,  [1]: 61.7,  [2]: 18.9,  [3]: 2.4,  [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;
}
```

Run: lab4

```
/Users/nimnawijedasa/Desktop/fall/337/lab4/cmake-build-debug/lab4
Array a has 5 elements and macro ELEMENTS returns 5
Array b has 20 elements and macro ELEMENTS returns 20

Process finished with exit code 0
```

Exercise D

```c
/*
 *  lab4exD.c
 *
 * ENSF 337 Lab 4 Exercise  D
 *
 */

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

int my_strlen(const char *s);
/*  Duplicates 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.
 */

void my_strncat(char *dest, const char *source, int n);
/*  Duplicates 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.
 */

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 strlen FUNCTION ... \n");

```

```c
    /* using 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 strncat FUNCTION ... \n");
    /* 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);

    /* 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: ");
```

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


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

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

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

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

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

    return 0;
}

int my_strlen(const char *s){
    int i = 0;
    while (*(s + i) != '\0') {
        i++;
    }
    return i;
}

void my_strncat(char *dest, const char *source, int n){
    while (*dest) {
        dest++;
    }
    int i = 0;
    while ( i < n) {
        *dest = *source;
        dest++;
        source++;
        i++;
    }
    *dest ='\0';
}
```

```
150
151 int my_strncmp(const char* str1, const char* str2){
152     int i = 0 , j = 0 ;
153     while (*str1) {
154         i += (int) str1;
155         str1++;
156     }
157     while (*str2) {
158         j += (int) str2;
159         str2++;
160     }
161     if (i == j)
162         return 0;
163     else
164         return (i - j);
165 }
166
167
```

```
TESTING 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 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 -172
"ABCD" is equal "ABCD" ... strcmp returns 0
"ABCD" is less than "ABCd" ... strcmp returns -576
"Orange" is greater than "Apple" ... strcmp returns 16193
Program ended with exit code: 0
```

Exercise E

```c
/* prog_two.c
* ENSF 337 Lab 4 Exercise  E
*
*/

#include <stdio.h>
#include <limits.h>
#include "read_input.h"

#define SIZE 50

int main(void)
{
    double n = 0;
    char digits[SIZE];

    int y = EOF;

    while (1)
    {
        printf("\n\nEnter an integer or press Ctrl-D to quit: ");
        y = read_real(digits, n: SIZE, num: &n);

        if(y == 1)
            printf("\nYour integer value is: %lf", n);
        else if(y == EOF){
            printf("\nGood Bye.\n");
            break;
        }
        else
            printf("\n%s is an invalid integer.", digits);
    }

    return 0;
}
```

```c
/* read_double.c
 * ENSF 337 Lab 4 Exercise  E
 *
 */

#include "read_input.h"

int read_real(char* digits, int n, double * num)
{
    if(get_string(digits, n)== EOF)
        return EOF;

    if(is_valid_double(digits)){
        if(digits[0] == '-')
            *num = -convert_to_double( digits: digits + 1);
        else if(digits[0] == '+')
            *num = convert_to_double( digits: digits + 1);
        else
            *num = convert_to_double(digits);
        return 1;
    }

    return 0;
}

int is_valid_double(const char* digits)
{
    int valid = 1;
    int i, j=0,decimal =0;

    /* i = index where first digit should be */
    if(digits[0] == '+' || digits[0] == '-' || digits[0] == '.' )
        i = 1;
    else
        i = 0;

    /*check for decimal places*/
    while ( digits[j] != '\0')
    {
        if (digits[j] == '.')
            decimal++; j++;
    }
```

```
43
44          /* Must have at least one digit, and no non-digits. */
45          if (digits[i] == '\0')
46              valid = 0;
47          else
48              while (valid && (digits[i] != '\0')) {
49                  if(digits[i] < '0' ||  digits[i] > '9' )
50                      valid = 0;
51                  if (digits[i] == '.')
52                      valid = 1;
53                  if (decimal > 1)
54                      valid =0;
55                  i++;
56              }
57
58          return valid;
59      }
60
61      double convert_to_double(const char *digits)
62      {
63          int decimal_pos =0;
64          int decimal_pres = 0;
65          double sum  = 0;
66          double sum_decimal =0;
67          int i = 0;
68          int j = 0;
69          int power = 1;
70
71
72          int len = 0;
73          while(digits[len] != '\0')
74          {
75              len++;
76          }
77
78          while(digits[decimal_pos] != '\0')
79          {
80              if(digits[decimal_pos] == 46)
81              {
82                  decimal_pres = 1;
83                  break;
84              }
85              decimal_pos++;
```

```
        }
        decimal_pos++;

    }

    if (decimal_pres == 0)
    {
        while (digits[i] != '\0') {
            sum = 10 * sum + (digits[i] - '0');
            i++;
        }

    }
    else
    {
            while(digits[i] != 46) {
                sum= 10 * sum+ (digits[i] - '0');
                i++;
            }
            i++;
            j=i;
            while(digits[i] != '\0'){
                sum_decimal = 10 * sum_decimal + (digits[i] - '0');
                i++;
            }
            for(int k = 0; k < i-j; k++)
                power *= 10;
            sum_decimal = sum_decimal / power;
            sum = sum+ sum_decimal;

    }
    return sum;
}
```

```
Enter an integer or press Ctrl-D to quit: 23.4
Your integer value is: 23.400000
Enter an integer or press Ctrl-D to quit: .56
Your integer value is: 0.560000
Enter an integer or press Ctrl-D to quit: -.23
Your integer value is: -0.230000
Enter an integer or press Ctrl-D to quit: -.045
Your integer value is: -0.045000
Enter an integer or press Ctrl-D to quit: -0.0000067
Your integer value is: -0.000007
Enter an integer or press Ctrl-D to quit: 564469999
Your integer value is: 564469999.000000
Enter an integer or press Ctrl-D to quit: +8773469
Your integer value is: 8773469.000000
Enter an integer or press Ctrl-D to quit: +.5
Your integer value is: 0.500000
Enter an integer or press Ctrl-D to quit: 12..999
12..999 is an invalid integer.
Enter an integer or press Ctrl-D to quit: 23avb45
23avb45 is an invalid integer.
Enter an integer or press Ctrl-D to quit: 23,347
23,347 is an invalid integer.
Enter an integer or press Ctrl-D to quit: + 234 77
+ 234 77 is an invalid integer.
Enter an integer or press Ctrl-D to quit:
```