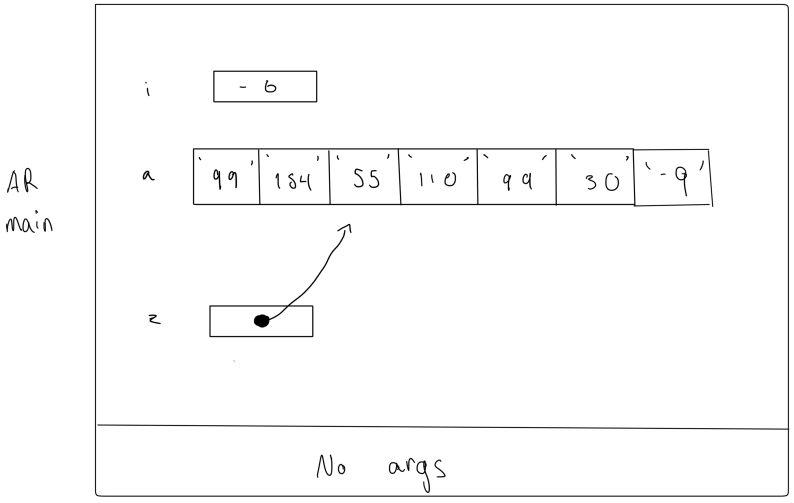
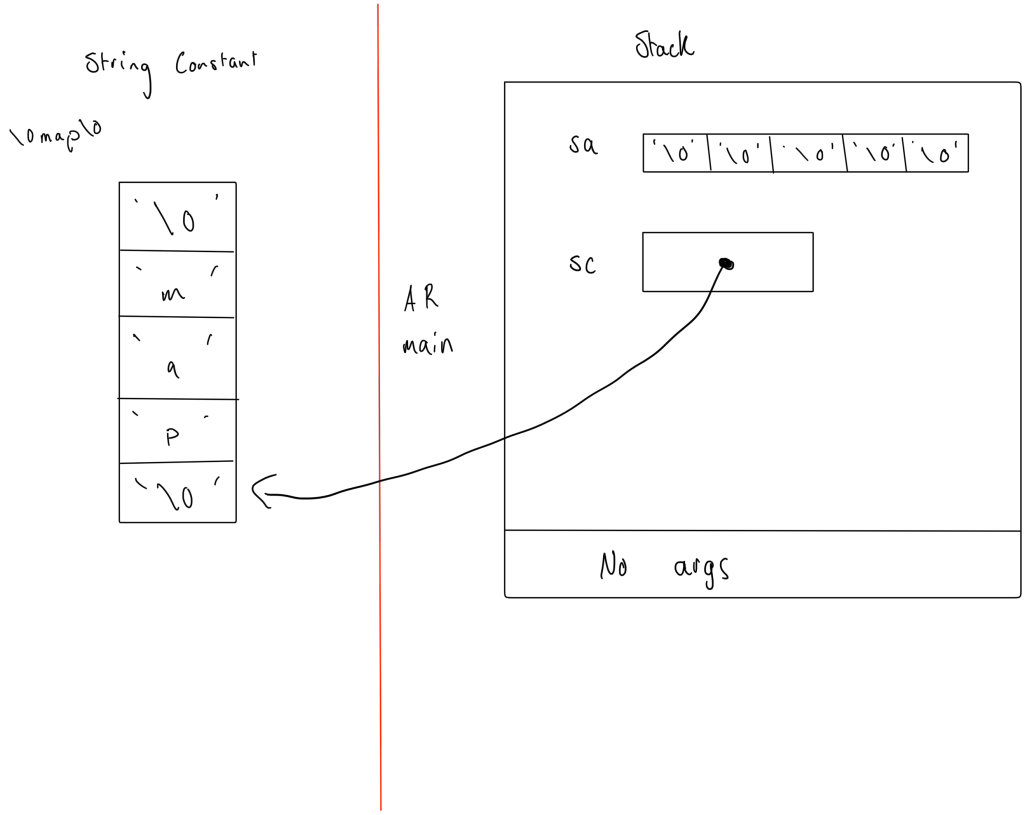
Course: Programming Fundamental – ENSF 337
Lab #: Lab 4
Instructor: M. Moussavi
Student Name: Carl Soriano
Lab Section: B01
Date submitted: Oct 5, 2022

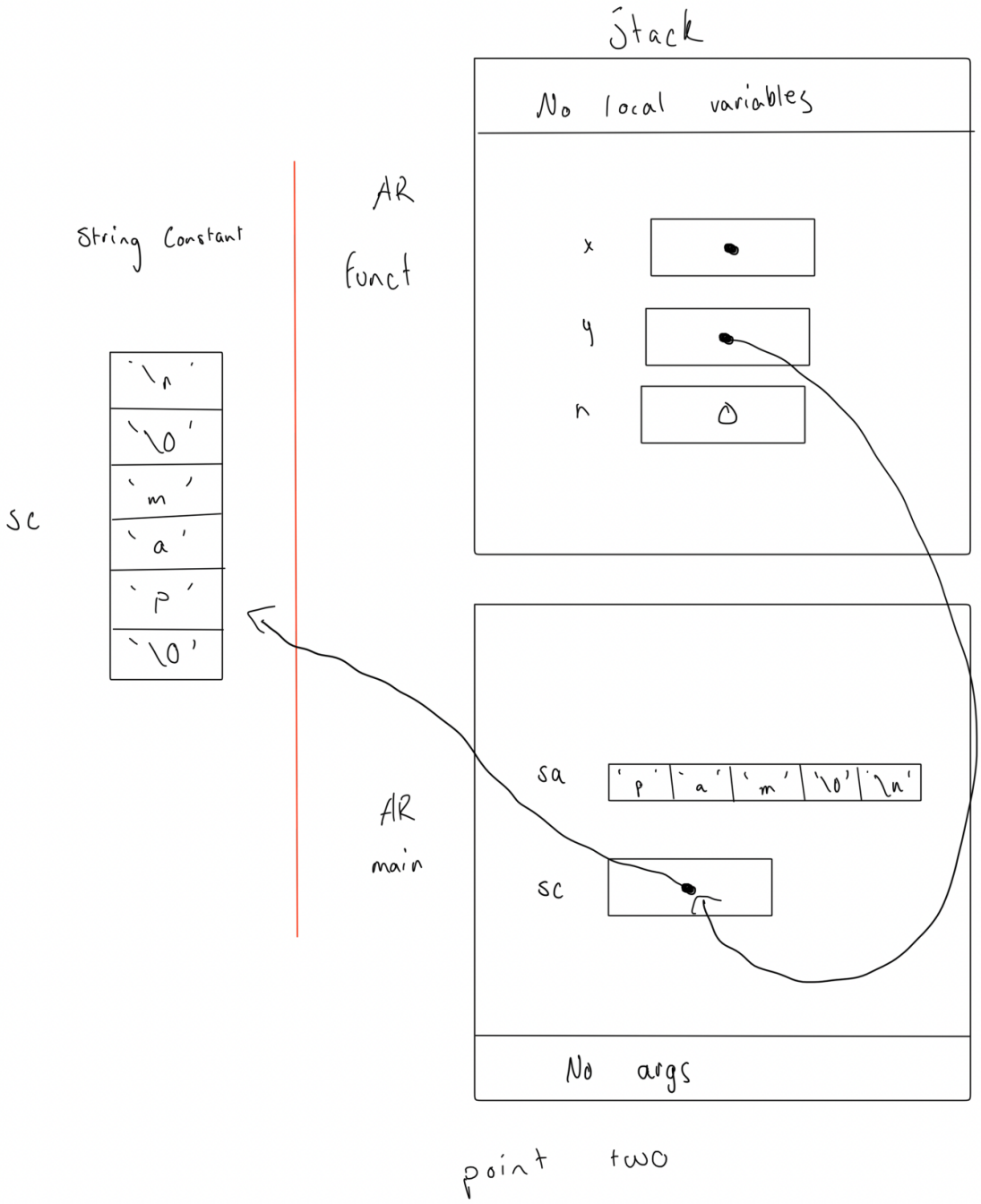
Exercise A



Exercise B



point one


Stack
No local variables
AR
Funct
String Constant
x
y
n
sc
'\n'
'\0'
'm'
'a'
'p'
'\0'
sa
'p'
'a'
'm'
'\0'
'\n'
AR
main
sc
No args

point two

Exercise C

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

#include <stdio.h>
#define ELEMENTS(x) (sizeof(x)/sizeof(x[0]))
//indexs x[0] it gives the size of the specific element IE int or double


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;
}
```

```
Desktop                         Practice_for_319
Documents                       Public
Downloads                       Sites
HelloWorld                      main
Library                         main.s
Movies                          print("Hello world").py
Music                           seaborn-data
(base) MacBook-Pro:~ carlsoriano$ cd Documents/ENSF337/
(base) MacBook-Pro:ENSF337 carlsoriano$ ls
LAB1    LAB2    LAB3    LAB4
(base) MacBook-Pro:ENSF337 carlsoriano$ cd LAB4
(base) MacBook-Pro:LAB4 carlsoriano$ ls
Exercise_C      Lab4 Extra
(base) MacBook-Pro:LAB4 carlsoriano$ cd Exercise_C/
(base) MacBook-Pro:Exercise_C carlsoriano$ ls
Exercise_C              Exercise_C.xcodeproj
(base) MacBook-Pro:Exercise_C carlsoriano$ cd Exercise_C
(base) MacBook-Pro:Exercise_C carlsoriano$ ls
Exercise_C.c
(base) MacBook-Pro:Exercise_C carlsoriano$ gcc Exercise_C.c
(base) MacBook-Pro:Exercise_C carlsoriano$ ./a.out
Array a has 5 elements and macro ELEMENTS returns 5
Array b has 20 elements and macro ELEMENTS returns 20
(base) MacBook-Pro:Exercise_C carlsoriano$
```

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");

    /* 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 */
    strncat(my_string, str5, 3);
    printf("\nExpected to display: my_string contains \"tic\"");
    printf("\nmy_string contains \"%s\"", my_string);

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

    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
    strncat(my_string, str3, 6);
    printf("\nExpected to display: my_string contains \"tic-tac-toe\"");
    printf("\nmy_string contains:\"%s\"", my_string);

    length = (int) 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 returns %d",
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;
}

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

    return (int)(length - s);

}

void my_strncat(char *dest, const char *source, int n)
{
    char *temp = dest;
        while(*dest){
            dest++;
        }
        while(n && *source){
            n--;
            *dest = *source;
            dest++;
            source++;
        }
        *dest = '\0';
        dest = temp;

}


int my_strncmp(const char* str1, const char* str2)
{
    for(int i = 0; ;i++)
    {
        if(*str1 == '\0' && *str2 == '\0')
        {
            return 0;
        }
        int ascii1 = (int)*str1;
        int ascii2 = (int)*str2;

        if(ascii1>ascii2)
            return 1;
        if(ascii2>ascii1)
            return -1;

    }
}
```

Output Exercise D

```
Exercise_D.c
(base) MacBook-Pro:Exercise_D carlsoriano$ gcc Exercise_D.c
(base) MacBook-Pro:Exercise_D carlsoriano$ ls
Exercise_D.c    a.out
(base) MacBook-Pro:Exercise_D carlsoriano$ ./a.out

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"... strcmp returns -1

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
(base) MacBook-Pro:Exercise_D carlsoriano$
```

Exercise E

Compiling error

Cant compile code

Submitted my files via D2L