Course: ENSF 337 - Fall 2020

Lab #: 4

Instructor: M. Moussavi

Student Name: Alexander Sembrat (30089188)

Lab Section: B01

Submission Date: October 12th 2020

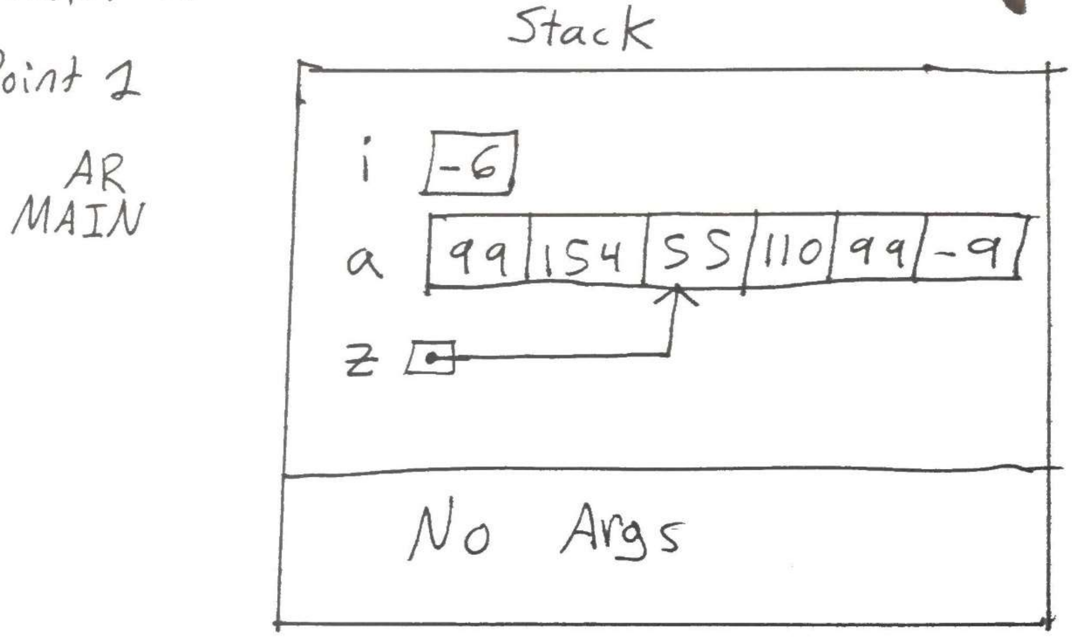
Part A

ENSF LAB 4

PART A

Alexander Sembratt
30089188

Point 1

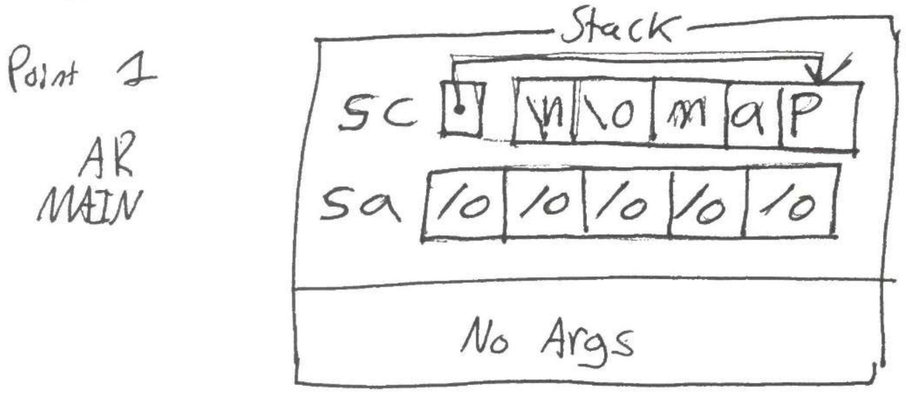


Part B

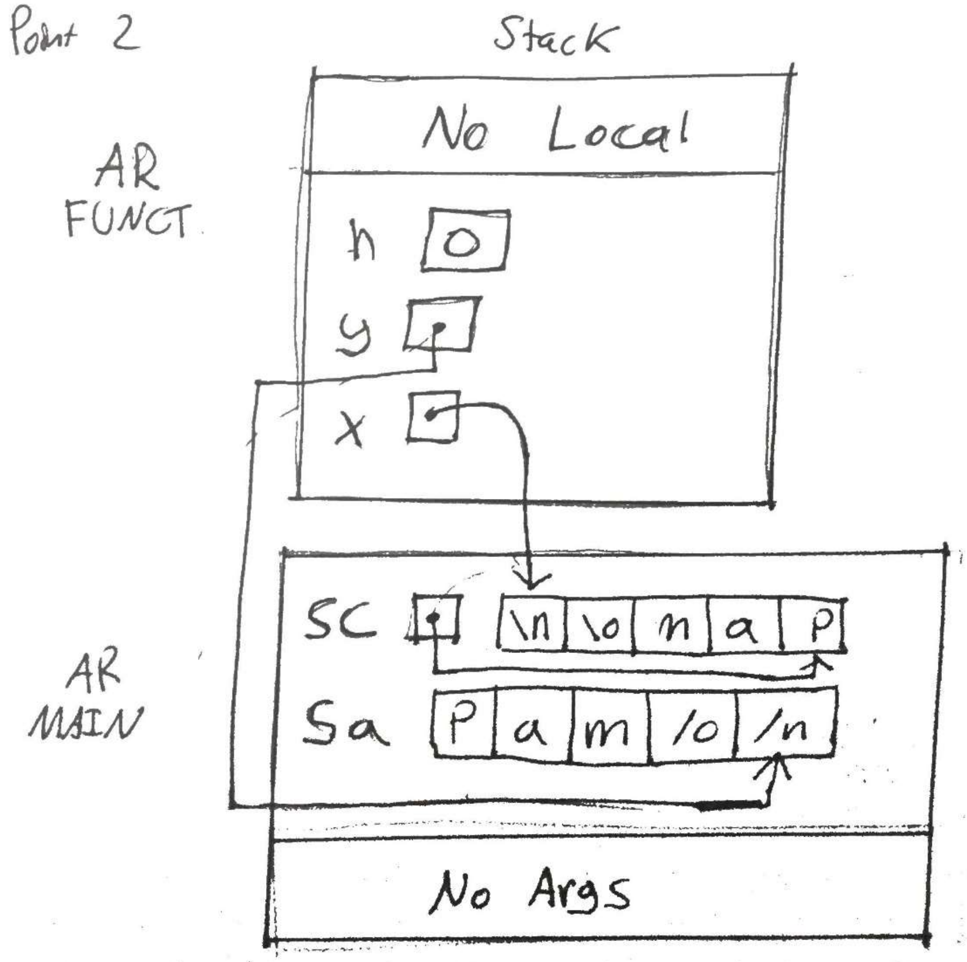
ENSF LAB 4
PART B

Alexander Sembrat
30089188

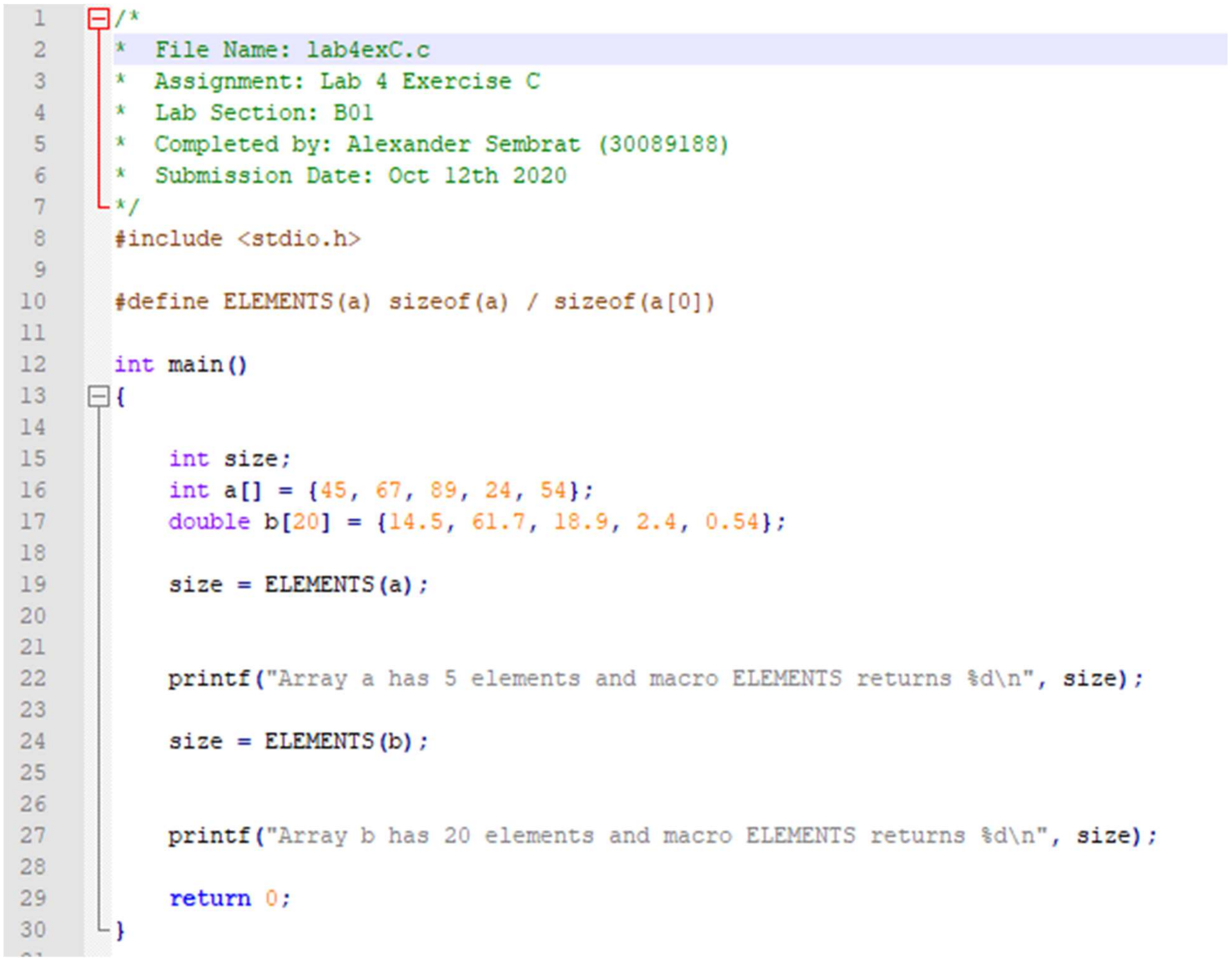
Point 1



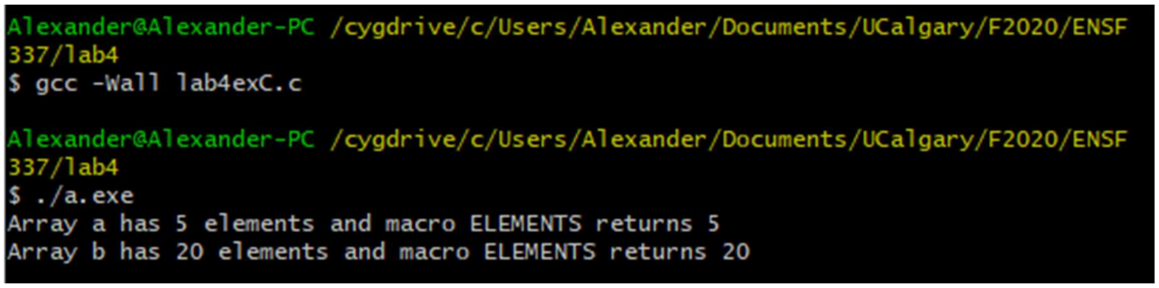
Point 2



Part C

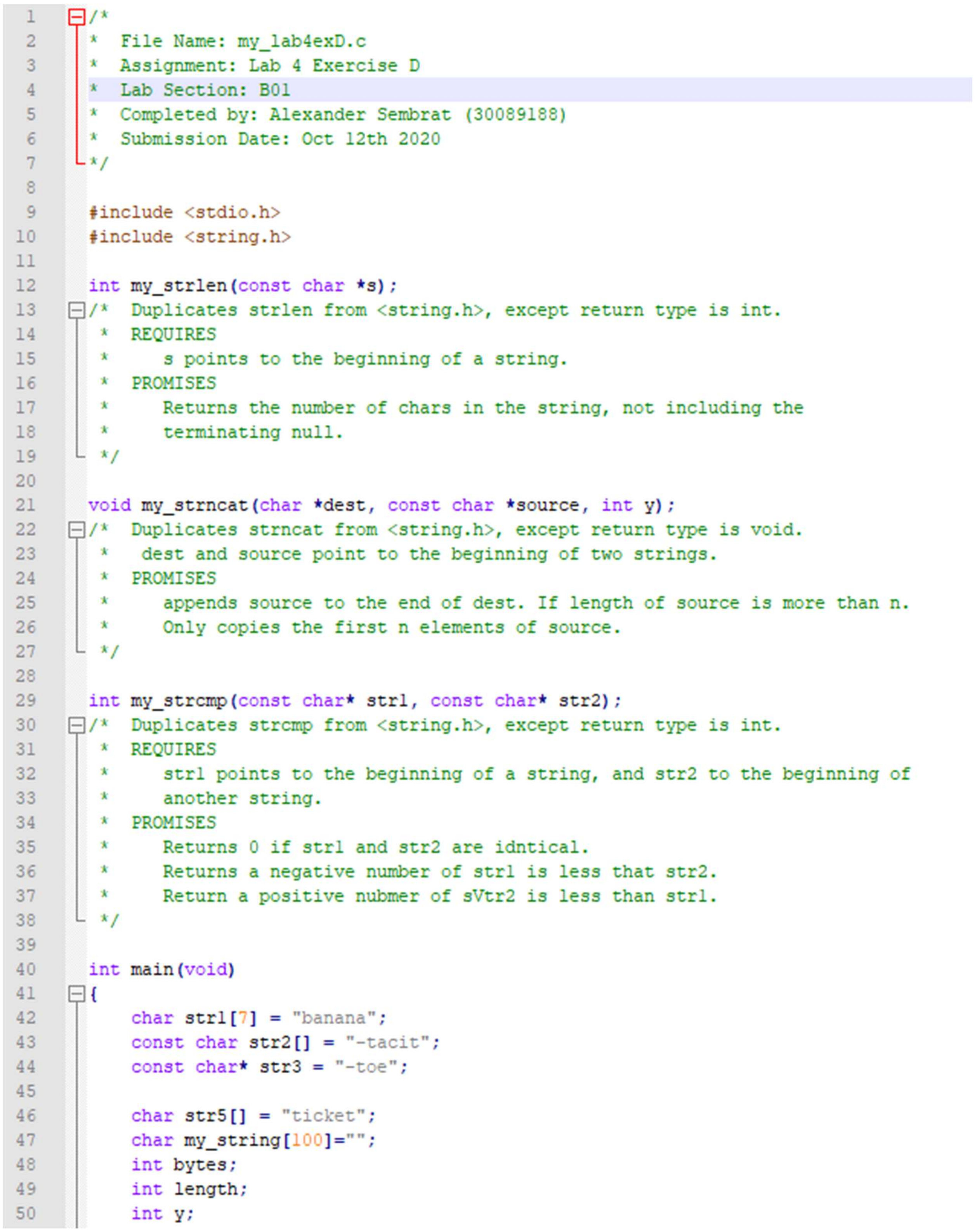


```c
/*
*  File Name: lab4exC.c
*  Assignment: Lab 4 Exercise C
*  Lab Section: B01
*  Completed by: Alexander Sembrat (30089188)
*  Submission Date: Oct 12th 2020
*/
#include <stdio.h>

#define ELEMENTS(a) sizeof(a) / sizeof(a[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;
}
```

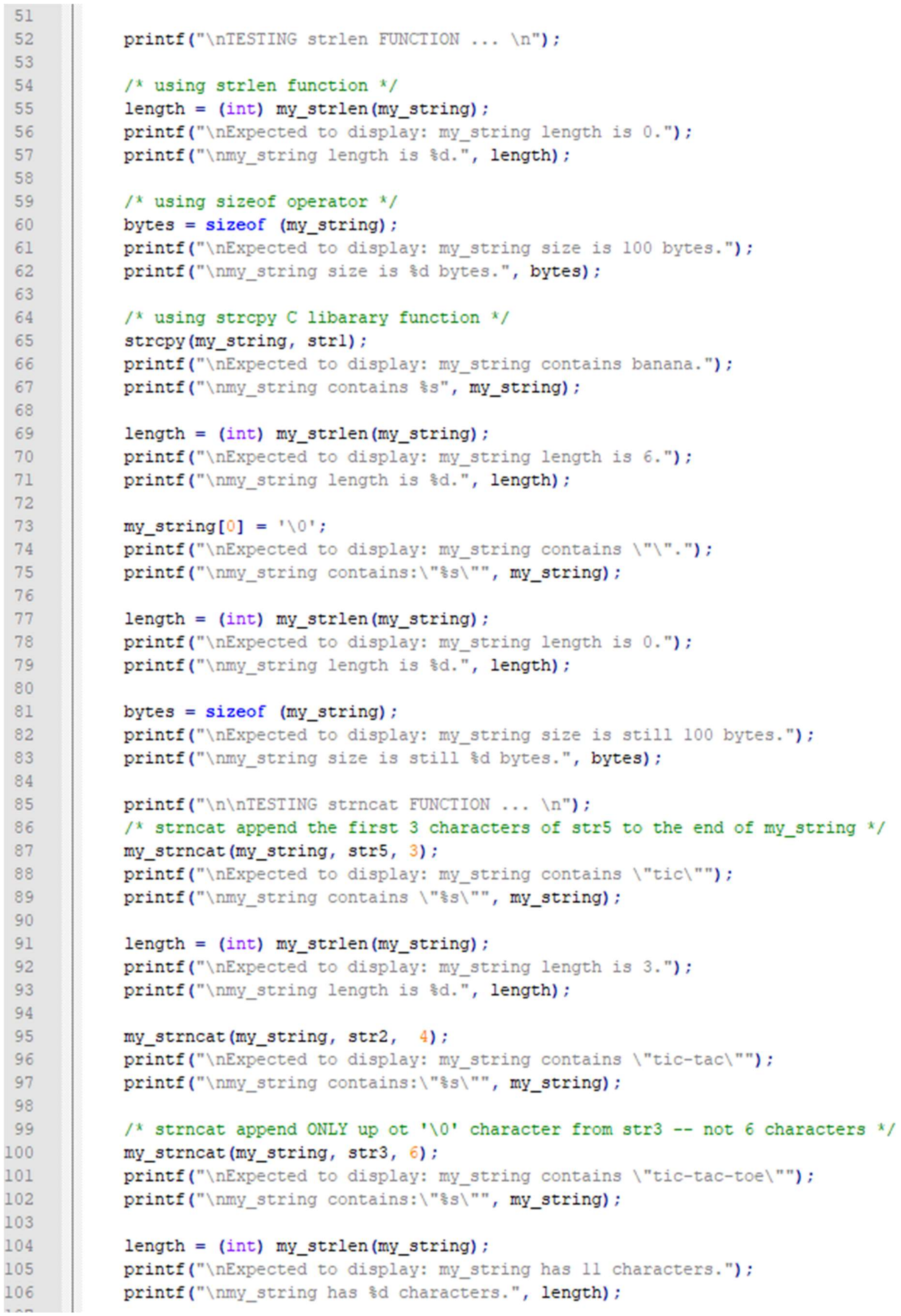


```
Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF
337/lab4
$ gcc -Wall lab4exC.c

Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF
337/lab4
$ ./a.exe
Array a has 5 elements and macro ELEMENTS returns 5
Array b has 20 elements and macro ELEMENTS returns 20
```

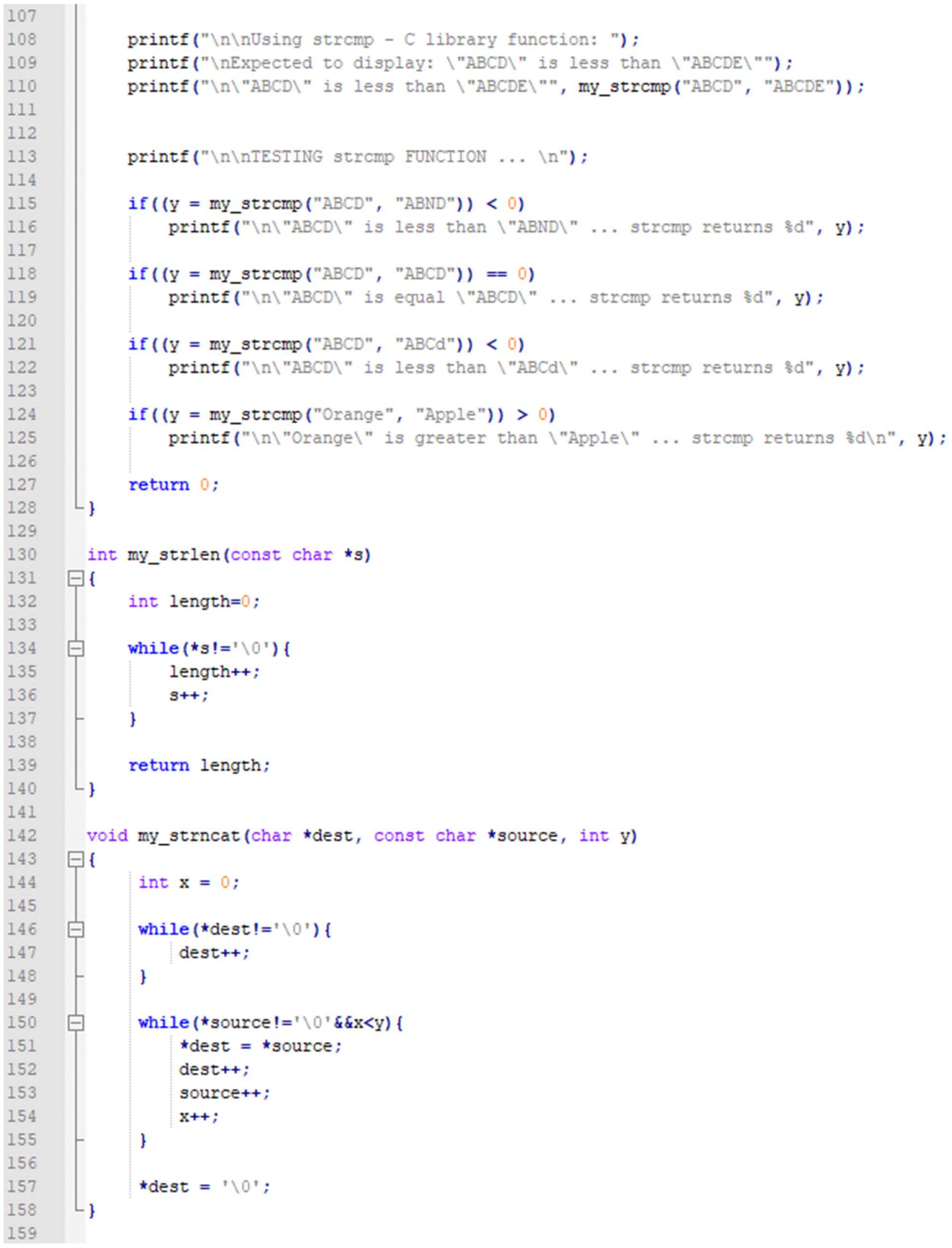
Part D



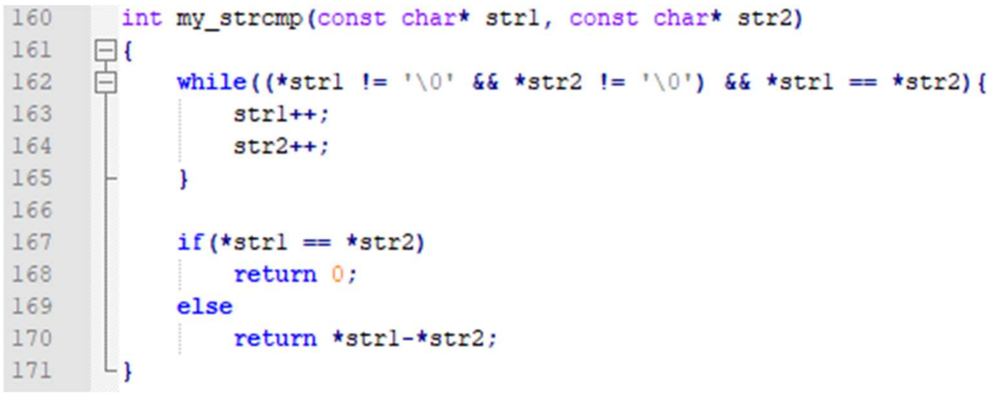
```c
/*
*  File Name: my_lab4exD.c
*  Assignment: Lab 4 Exercise D
*  Lab Section: B01
*  Completed by: Alexander Sembrat (30089188)
*  Submission Date: Oct 12th 2020
*/

#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 y);
/*  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_strcmp(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 positive nubmer of sVtr2 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;
```



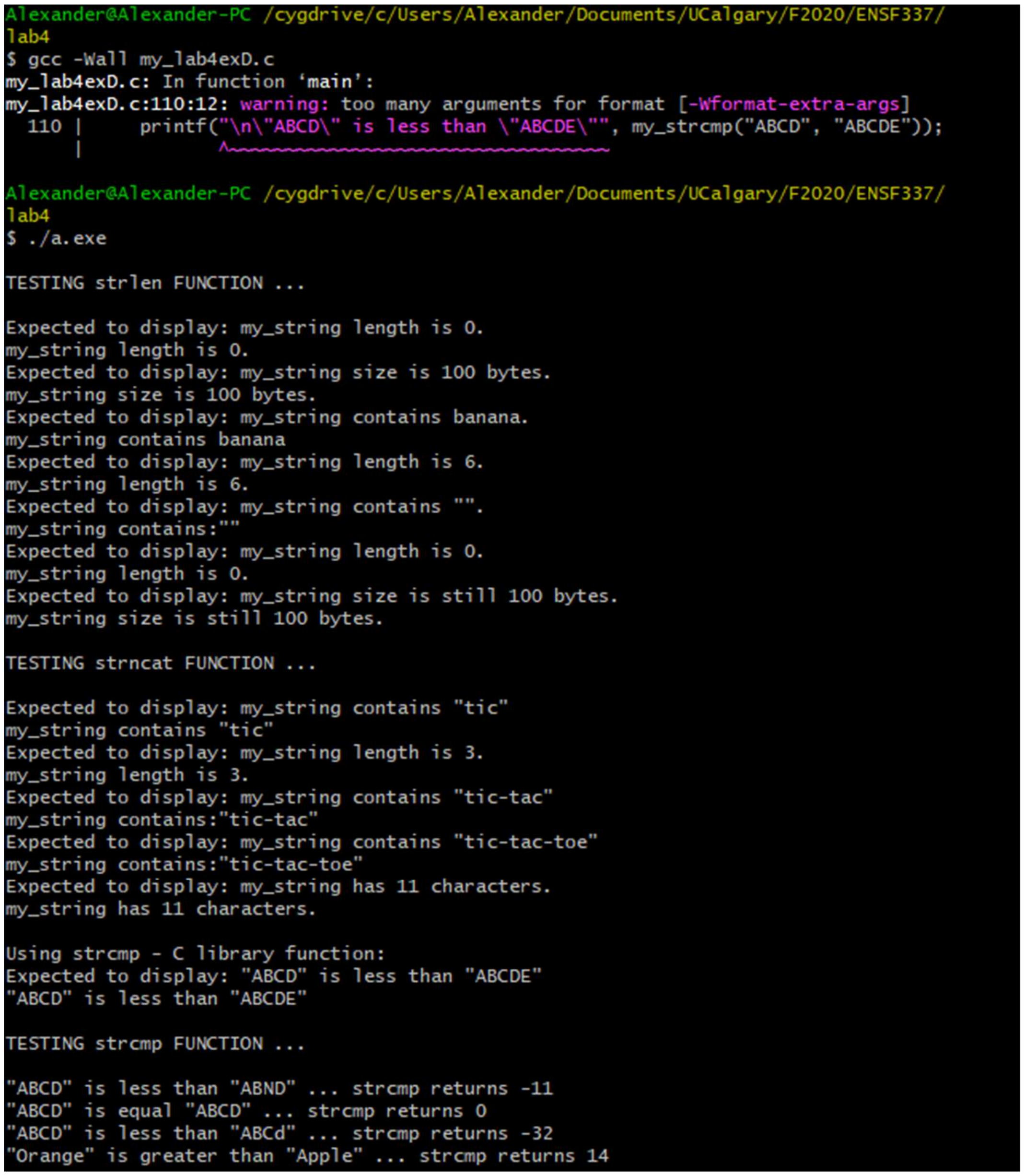
```c

    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 */
    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);
```



```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_strcmp("ABCD", "ABCDE"));


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

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

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

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

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

    return 0;
}

int my_strlen(const char *s)
{
    int length=0;

    while(*s!='\0'){
        length++;
        s++;
    }

    return length;
}

void my_strncat(char *dest, const char *source, int y)
{
     int x = 0;

     while(*dest!='\0'){
         dest++;
     }

     while(*source!='\0'&&x<y){
         *dest = *source;
         dest++;
         source++;
         x++;
     }

     *dest = '\0';
}

```



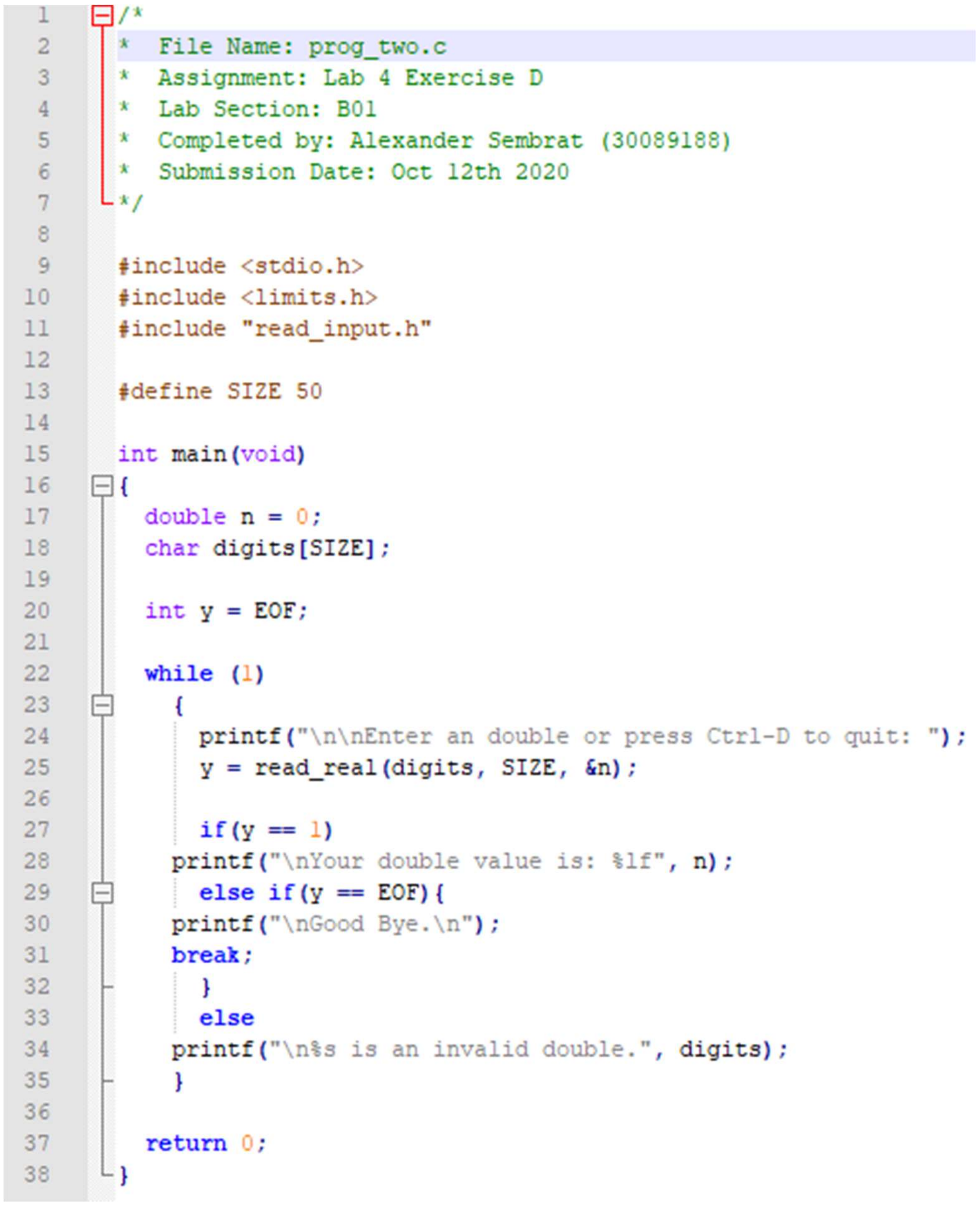
```c
int my_strcmp(const char* str1, const char* str2)
{
    while((*str1 != '\0' && *str2 != '\0') && *str1 == *str2){
        str1++;
        str2++;
    }

    if(*str1 == *str2)
        return 0;
    else
        return *str1-*str2;
}
```



```
Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/
lab4
$ gcc -Wall my_lab4exD.c
my_lab4exD.c: In function ‘main’:
my_lab4exD.c:110:12: warning: too many arguments for format [-Wformat-extra-args]
  110 |     printf("\n\"ABCD\" is less than \"ABCDE\"", my_strcmp("ABCD", "ABCDE"));
      |            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF337/
lab4
$ ./a.exe

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 -11
"ABCD" is equal "ABCD" ... strcmp returns 0
"ABCD" is less than "ABCd" ... strcmp returns -32
"Orange" is greater than "Apple" ... strcmp returns 14
```

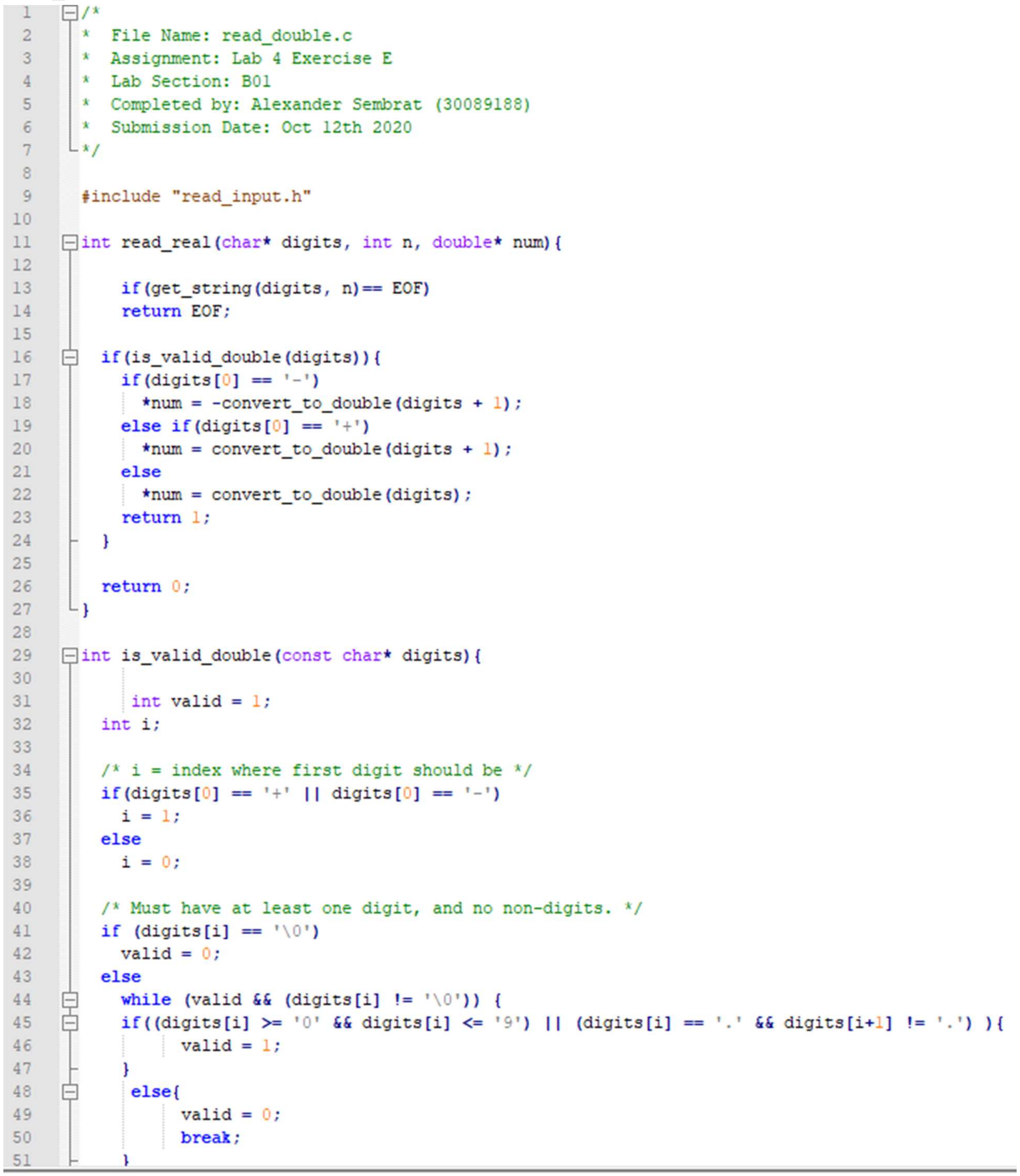
Part E

prog_two:

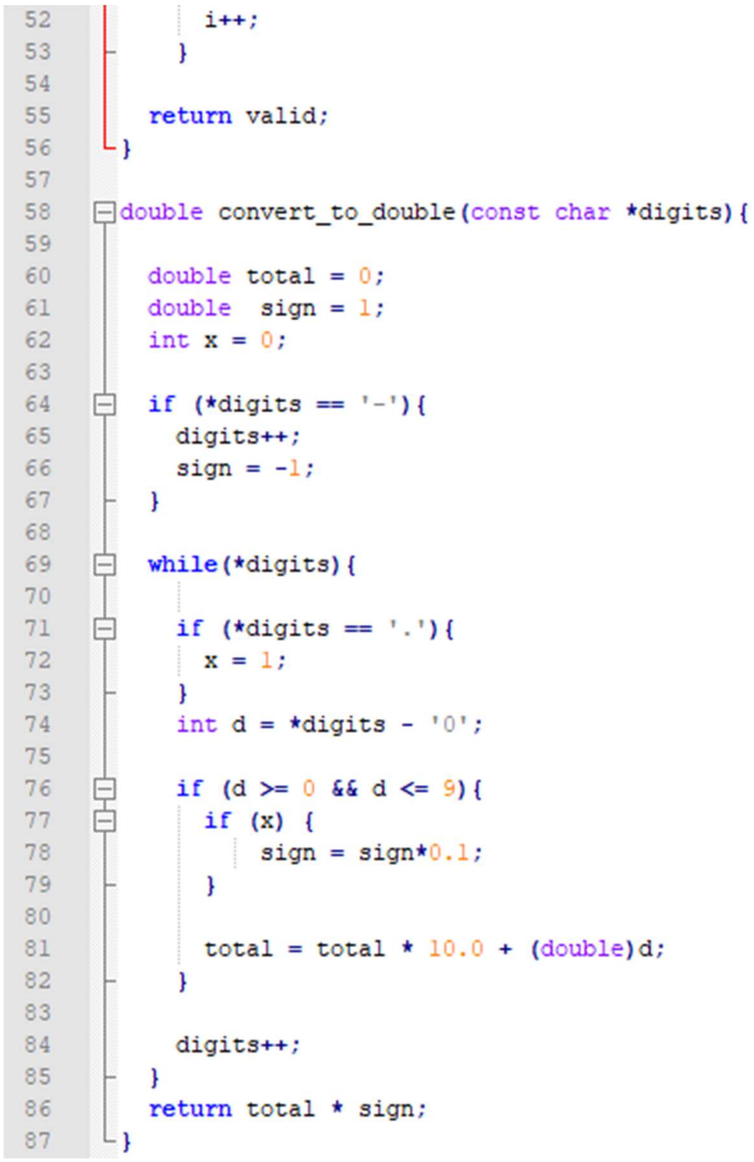


```
/*
*  File Name: prog_two.c
*  Assignment: Lab 4 Exercise D
*  Lab Section: B01
*  Completed by: Alexander Sembrat (30089188)
*  Submission Date: Oct 12th 2020
*/

#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 double or press Ctrl-D to quit: ");
      y = read_real(digits, SIZE, &n);

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

  return 0;
}
```

read_double :

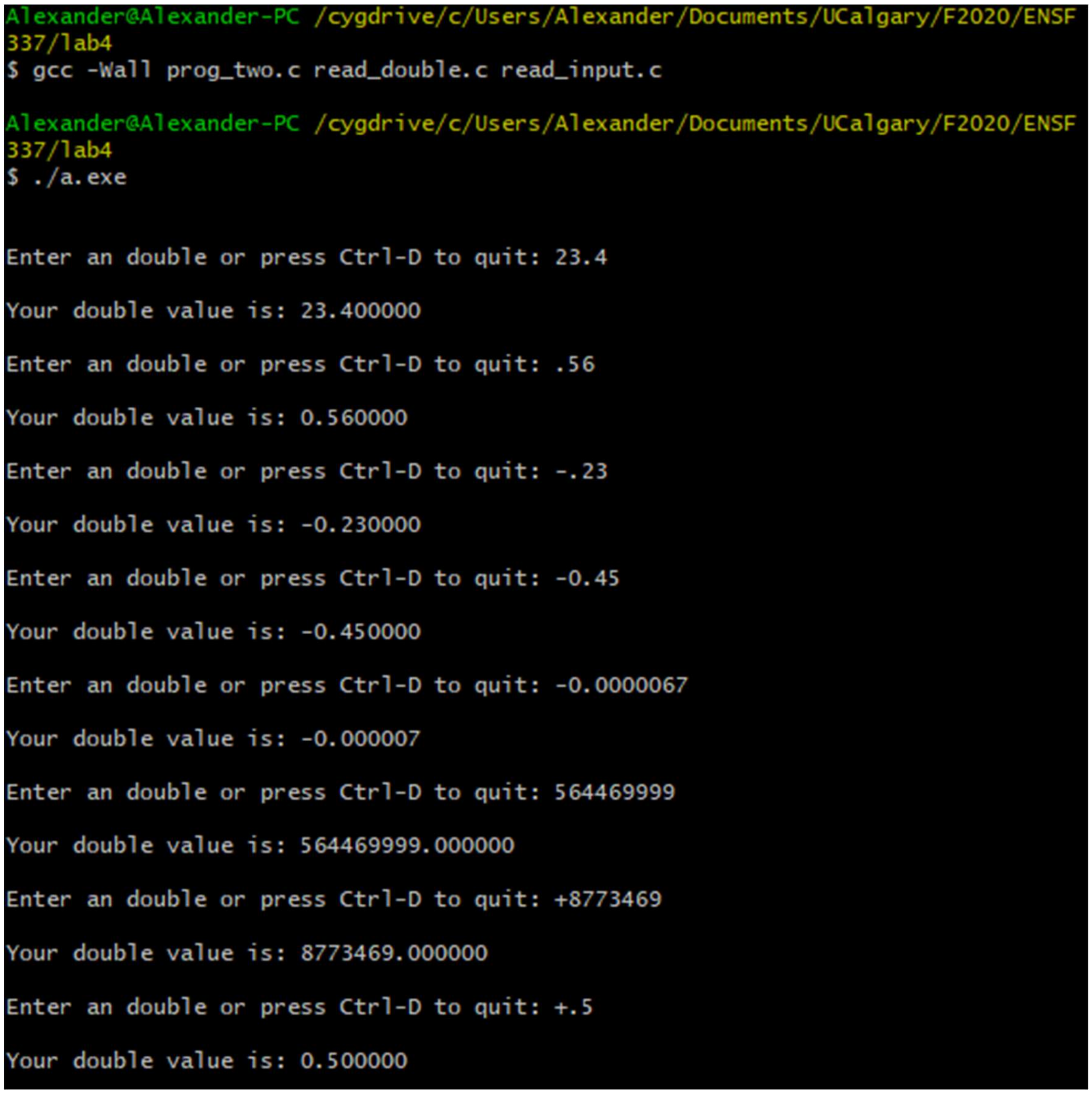


```c
/*
*  File Name: read_double.c
*  Assignment: Lab 4 Exercise E
*  Lab Section: B01
*  Completed by: Alexander Sembrat (30089188)
*  Submission Date: Oct 12th 2020
*/

#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 + 1);
    else if(digits[0] == '+')
      *num = convert_to_double(digits + 1);
    else
      *num = convert_to_double(digits);
    return 1;
  }

  return 0;
}

int is_valid_double(const char* digits){

     int valid = 1;
  int i;

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

  /* Must have at least one digit, and no non-digits. */
  if (digits[i] == '\0')
    valid = 0;
  else
    while (valid && (digits[i] != '\0')) {
    if((digits[i] >= '0' && digits[i] <= '9') || (digits[i] == '.' && digits[i+1] != '.') ){
          valid = 1;
    }
     else{
          valid = 0;
          break;
    }
```



```c
      i++;
    }

  return valid;
}

double convert_to_double(const char *digits){

  double total = 0;
  double  sign = 1;
  int x = 0;

  if (*digits == '-'){
    digits++;
    sign = -1;
  }

  while(*digits){

    if (*digits == '.'){
      x = 1;
    }
    int d = *digits - '0';

    if (d >= 0 && d <= 9){
      if (x) {
          sign = sign*0.1;
      }

      total = total * 10.0 + (double)d;
    }

    digits++;
  }
  return total * sign;
}
```

Outputs:



```
Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF
337/lab4
$ gcc -Wall prog_two.c read_double.c read_input.c

Alexander@Alexander-PC /cygdrive/c/Users/Alexander/Documents/UCalgary/F2020/ENSF
337/lab4
$ ./a.exe


Enter an double or press Ctrl-D to quit: 23.4

Your double value is: 23.400000

Enter an double or press Ctrl-D to quit: .56

Your double value is: 0.560000

Enter an double or press Ctrl-D to quit: -.23

Your double value is: -0.230000

Enter an double or press Ctrl-D to quit: -0.45

Your double value is: -0.450000

Enter an double or press Ctrl-D to quit: -0.0000067

Your double value is: -0.000007

Enter an double or press Ctrl-D to quit: 564469999

Your double value is: 564469999.000000

Enter an double or press Ctrl-D to quit: +8773469

Your double value is: 8773469.000000

Enter an double or press Ctrl-D to quit: +.5

Your double value is: 0.500000
```