LAB 4 — Arrays and Pointers

Problem A

A.1 Specification

Write a C program that contains two functions, one to input two *non-empty* strings and the other to compare them. The comparison returns an integer that indicates the ***first*** position (array index) where the two strings differ.

A.2 Implementation

- The program to be submitted is named `lab4a.c`. **Use the given template `lab4a.c`** and fill in your code. **Submit only file `lab4a.c`.**
- The first function to be implemented is `myStrInput()`. See file `lab4a.c` for its specification. Use `getchar` and a loop to read a line of characters, and store the input characters into the array. The loop terminates when a new line character '`\n`' is entered. The new line character '`\n`' is NOT part of the line (i.e., discard the new line character '`\n`').
- The second function to be implemented is `myStrCmp()`. See file `lab4a.c` for its specification. The function returns an integer that indicates the ***first*** position (array index) where the two strings differ. Consider the following two special cases:
    - Two strings are equal. In that case the return value is –1.
    - One string is a substring of the other (e.g., CSE2031 and CSE2031E3.0). In that case, the return value is the length of the shorter string (i.e., the index of the null character in the shorter string).
- In both functions, **do not use array indexing** such as `s[i]`. **Use only pointers and address arithmetic** to manipulate the array elements. If you use array indexing in your code, your program will not be marked and given zero point.
- Do not modify the function definitions in file `lab4a.c`.

A.3 Sample Inputs/Outputs

```
indigo 51 % a.out
This is CSE2031.
CSE2031E3.0
0
indigo 54 % a.out
I'll go now.
I will go soon.
1
indigo 49 % a.out
This is 2031.
This is 2011.
10
indigo 50 % a.out
abc
abc
-1
indigo 52 % a.out
CSE2031
CSE2031E3.0
7
indigo 53 % a.out
It is going to snow tomorrow.
It is going to rain tonight.
15
```

Problem B

B.1 Specification

Write a C program to input a line of characters in the form of a floating-point number, convert the line of characters into an actual floating-point number, and display on the standard output the floating-point number.

B.2 Implementation

- The program is named `lab4b.c`. **Use the given template `lab4b.c`** and fill in your code.
- You are given an array of characters of size `MAX_SIZE` where `MAX_SIZE` = 100. The array is named `my_strg`.
- Use `getchar` and a loop to read a line of characters, and store the input characters into array `my_strg`. The loop terminates when a new line character '`\n`' is entered. The new line character '`\n`' is NOT part of the line (i.e., discard the new line character '`\n`').
- The input line contains only characters '0' to '9' and the dot character '**.**' in the form of a valid positive floating point number of the following format: **[integer part] . [fractional part]**
- Convert the input line of characters to a **double** floating-point number, which is stored in variable `my_number`.

- Display on the standard output the double floating-point number `my_number` using the `printf` statement as follows:

```
printf( "%.6f\n", my_number );
```

- Assume that the input line of characters represents a valid floating point number of the form [integer part] **.** [fractional part]

B.3 Sample Inputs/Outputs

```
indigo 360 % lab4b
24.5
24.500000
indigo 361 % lab4b
76.24
76.240000
indigo 362 % lab4b
100.0
100.000000
indigo 363 % lab4b
0.255
0.255000
indigo 364 % lab4b
12.9999999999
13.000000
indigo 365 % lab4b
1.00000000099
1.000000
indigo 366 % lab4b
```

Common Notes

All submitted files should contain the following header:

```
/**************************************
*     EECS2031 – Lab 4                *
*     Filename:  Name of file         *
*     Author: Last name, first name   *
*     Email: Your email address       *
*     Login ID: Your login ID         *
**************************************/
```

In addition, all programs should follow the following guidelines:

- Include the `stdio.h` library in the header of your `.c` files.
- Use `printf` to print text and outputs according to the required formats.
- End each output result with a new line character '`\n`'.
- Do not use any C library functions except `getchar(), putchar(), scanf()` and `printf()`.
- **Assume that all inputs are valid (no error checking is required on inputs).**