# SWE2024-43: System Programming Lab (Fall 2021)

Programming Assignment #1

Due: **October 12th** (Tue.), **11:59 PM**

## 1. Introduction

You may get used to use file I/O and data structure with this assignment.

## 2. Specification

In this assignment, your goal is to read a movie scenario file with name entered as a command line input and write a code that performs a specific search function. After your program starts, your program waits for user's keyword input and performs searching function mentioned below.

### 2-1 Definitions of term

Before starting the description, we define the terms to use.

1) **Word**: List of characters (not empty) that do not include **white space (tab, space, new line)**. Words does not distinguish between upper and lowercase letters.

**Ex) input: He's very good boy. → words: [He's, very, good, boy.]**

2) **Line**: List of words separated by **'₩n'** on input. **The line number of the first line on input is 1.**

**You must count empty line.**

3) **Index**: The position of the character on the line. **The index of the first character on each line is 0.**

**You must count white spaces and writing symbols**.

### 2-2 Implementation

① **Searching single word locations**

**If your program receives input with a single word,** search for a word's location in a given movie scenario.

● Find the word your program received in the movie script and print it out on stdout in the form below.

**[line number]:[start index of the word]**

② **Searching several words locations**

**If your program receives input with multiple word (separate with a single space and no wrapper),** search for lines containing both words. More than 2 words can be given as input.

● Find the line containing both words entered in the movie scripts and print it on stdout in the form below.

**[line number]**

③ **Searching several consecutive words locations**

**If your program receives a phrase input wrapped in “”,** search the lines containing the phrase. More than 2 words can be given as input.

● Find the line containing received phrase in the movie scripts and print it on stdout in the form below.

**[line number]:[start index of the phrase]**

④ **Searching a simple regular expressing keyword locations**

**If your program receives input which two words are formed as [word1] * [word2]**, print the location of the keyword, which **contains one or more words** between word1 and word2 **in a line**. Input value is always two words.

● Find the location of keyword that explained above, and print it on stdout in the form below.

**[line number]**

● If the received keywords appear multiple times in the movie scenario, then you have to print them all.

● For the case of ① and ③, you have to **print all locations of even if they exists in the same line**. But the case of ②, ④, you just print once for duplicated line number.

● **You must add a single space after a single keyword location. And you have to print new line when searching is done. For example:**

```
15:23 17:10 23:4
15 17 23
```

● Search results should be output sequentially from the top of the file.

● **You may follow and refer the details about keyword input "7. Example" section.**

## 3. Score Policy

**1) Source code (100%)**

**We evaluate your assignment with many different movie scenarios.**

**You have to follow the format explained above, and you can't get any point if you don't.**

## 4. Restriction

- You have to do your assignment on **Linux environment**.

*** You must not use stdio.h, string.h library. (If you use or employ them, you will get 0 point)**

- Input text file name is given as **command line argument**(argv[1]).

- The word means, a string separated by **white spaces (tab, space, new line)**, and we don't distinguish between upper and lowercase letters.

## 5. Hand in instructions

* Your program must be written in one source code(**student_id.c**).

* Source code must be compiled using **make** command. So, you should write your own **Makefile**.

* Compress your **source code** and **Makefile** into "**student_id.tar.gz**" and submit it to **iCampus**.

In your tar.gz file, only "**student_id.c**" and "**Makefile**" should exist. (no scenario file, no directory)

* If you don't follow submission format or code is not compiled with make, **you will get 0 point**.

## 6. Precautions

- Late submission **can be reduced by 10% or up to 30% after the deadline** **(0 points after 3 days).**
- You can discuss the task together, but you have to write the program source code yourself.
- If you copy someone else's assignment, **you both do a zero-point job**, even if you copy the source code you found **on the Internet**.
- If you have any question, please use **Q&A board** in iCampus (not email or message), so that the question can be shared.

## 7. Example (Red highlighted words are inputs)

```
$ ./assignmetn1.out 500-Days-of-Summer_s.txt

500 days

6 6419

he is

50 1039 1822 1955 2256 2315 3494 3503 4353 4360 4445 4831 5101 5885 6325

"he is"

1955:33 2315:42 5885:22

he*is

2315 4445 5101

she he

1370 1512 1513 3423 3473 3478 3550 4255 4510 4515 5413 5672 6154 6188

loved

106:30 1122:9 1150:24 3961:25 4739:17 5921:28

…
```

```
$ ./assignmetn1.out 500-Days-of-Summer_s.txt > result.out
he

hey

she

tom

summer is

"summer is"

landscape

together

we*her

no much

immediately

no*much

quarterback

we

  (ctrl + c)


$ diff –bsq result.out answer.out

Files result.out and answer.out are identical

$ diff result.out answer.out
```

**(If your result file is equal to answer file, nothing is printed)**