CS201 – Fall 2020-2021 - Sabancı University

Take-Home Exam 3: A Simple Search Engine

Due November 25, Wednesday, 23:55 (Sharp Deadline)

Description

In this homework, you will write a simple program that searches for a string in another string. There are some special searching parameters and details about it will be given in later sections. First your program will get a string until the word "END" or "end" is present and then it will ask for a search string. Your program should be able to do multiple searches until the search string "QUIT" or "quit" is entered. The aim of this homework is to practice on string member functions and loops. Of course, you will need to use the topics that we have seen before such as if-else statements and functions, as needed.

Your take-home exams will be automatically graded using GradeChecker, so it is **very important to satisfy the exact same output given in the sample runs.** You can utilize GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check whether your implementation is working in the expected way. To be able to use GradeChecker, you should upload all of your files used in the take-home exam (**only** *your_main_cpp* file for this take-home exam). Additionally, you should submit all of your files to SUCourse. **Just a reminder, you will see a character ¶ which refers to a newline in your expected output.**

The name of your main source (cpp) file should be in the expected format: "SUCourseUsername_THEnumber.cpp" (all lowercase letters). Please check the submission procedures of the take-home exam, which are listed at the end of this document.

VERY IMPORTANT!

Your programs will be compiled, executed and evaluated automatically; therefore, you should definitely follow the rules for prompts, inputs and outputs. See **Sample Runs** section for some examples.

Order of inputs and outputs must be in the abovementioned format.

Following these rules is crucial for grading, otherwise our software will not be able to process your outputs and you will lose some grades in the best scenario.

IMPORTANT!

If your code does not compile, you will get **zero**. Please be careful about this and double check your code before submission.

Inputs

There are two inputs respectively; Source String and Search String. Both inputs are strings. You should note that there can be spaces between the words in the Source String (hint: use a while loop to read all words using cin). However, the source string cannot contain any characters other than numbers, letters and space in the Source String, such as punctuation.On the other hand, you can assume that there is no space or tab in the search string, a single word will be entered, you do not need to check input for this case. Search String must end with one of '+' , '.' , '*','**' characters. (See Sample Runs). Your program should get the Source String only once, but ask multiple Search Strings until "QUIT" or "quit" is entered.

Input Checks

All inputs must be checked. In the case of an invalid input, the program should ask for an input again and again until a correct input is entered (Hint: you may use while loop here). The rules of input check are given below:

- Source String must be concatenated with the words until the word "END" or "end" is present.
- Source String must be including only letters (Both upper and lower), digits and space character. (Hint: use ASCII table)
- Search String must end with one of the '+' , '.' , '*' characters.
- Search String cannot be empty.

Processing, Program Flow and Output

Your program should start with an introductory explanation and a prompt for the input(see sample runs). After all the inputs are entered correctly, your program should search for the string according to the rules and print out the index of the searched word.

Please note that all searches are **case sensitive**, which means that you shouldn't match *"Car"* with *"car"*.

Search Rules

1. **Search String ends with a '+':** The word must start with the search string.
   Example:
   ```
   Enter source string: Cars are fast END
   Enter search string: fa+
   index: 9 word: fast
   ```

2. **Search String ends with a '.':** The word ends with search string.
   Example:
   ```
   Enter source string: Cars are fast END
   Enter search string: ars.
   index: 1 word: Cars
   ```

3. **Search String ends with a '*':** The word that does **not** include the search string at the beginning and the end but may contain it in the middle.
   Examples:
   ```
   Enter source string: Cars are fast END
   Enter search string: as*
   index: 10 word: fast

   Enter source string: alan has a goal that study in alabama END
   Enter search string: a*
   index: 6 word: has
   index: 13 word: goal
   index: 18 word: that
   ```

4. **Search String ends with a '**':** Any word containing the search string regardless of location.
   Example:
   ```
   Enter source string: Cars are fast END
   Enter search string: r**
   index: 2 word: Cars
   index: 6 word: are
   ```

An example for same search string with different search operators:

```
Enter source string: Cars are fast END
Enter search string: s+
Enter search string: s*
index: 11 word: fast
Enter search string: s.
index: 3 word: Cars
Enter search string: s**
index: 3 word: Cars
index: 11 word: fast
```

An example for a different search string where search string occurs multiple times in a word:

```
Enter source string: Bu bayram da laylaylom gecti END
Enter search string: lay+
index: 13 word: laylaylom
Enter search string: lay*
Enter search string: lay**
index: 13 word: laylaylom
index: 16 word: laylaylom
Enter search string: lom+
Enter search string: lom.
index: 19 word: laylaylom
Enter search string: b+
index: 3 word: bayram
Enter search string: B+
index: 0 word: Bu
Enter search string: QUIT
```

Hint: You should use while loops and **find()** member function of the string class. Please also note that you can use **find(string searchWord, int position)** version of it to search after a given position in the string.

No abrupt program termination please!

You may want to stop the execution of the program at a specific place in the program. Although there are ways of doing this in C++, it is not a good programming practice to abruptly stop the execution in the middle of the program. Therefore, your program flow should continue until the end of the main function and finish there.

Important Remarks

Your program must be modular and you should avoid code duplication. Thus, you have to show your ability to use functions in an appropriate way. This will affect your grade. In general, **if your main function or any user-defined function is too long and if you do everything in main or in another user-defined function, your grade may be lowered. (Hint: implement string extraction and character deletion in separate functions) Please do not write everything in main and then try to split the task into some functions just to have some functions other than mail. This is totally against the idea of functional design and nothing but a dirty trick to get some points. Instead please design your program by considering necessary functions at the beginning.**

Try to use parametric and non-void functions wherever appropriate. Do NOT use any global variables (variables defined outside the functions) to avoid parameter use.
In this homework (and in the coming ones) you are not allowed to use instructions such as "exit" and "goto". These cause difficulties to control the flow of your programs. Thus, we do not approve using them. You are also not encouraged to use "break" and "continue". The use of "break" and "continue" prevent you from forming good readable loop conditions and hence prevent you from learning how to form good loops. Think cleverly in order not to use any of these instructions. If you don't know these commands, do not even try to learn them (we will explain "break" in class).

Please remark that there are two inputs described above in the Inputs part. While input order should be exactly the same as the sample runs, also no other input is allowed (such as a name for the introduction or anything else not mentioned in this homework specification). Since your submissions are processed automatically, extra inputs cause problems and delays in the processing and grading. If you do not follow this rule, your grade may be lowered.

SAMPLE RUNS ARE IN THE NEXT PAGE

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user. You have to display the required information in the same order and with the same words and characters as below.

Sample Run 1 (Normal Run)

```
Enter source string: Cars are fast END
Enter search string: fast+
index: 9 word: fast
Enter search string: r*
index: 2 word: Cars
index: 6 word: are
Enter search string: s.
index: 3 word: Cars
Enter search string: r+
Enter search string: s**
index: 3 word: Cars
index: 11 word: fast
Enter search string: e.
index: 7 word: are
Enter search string: t.
index: 12 word: fast
Enter search string: fa+
index: 9 word: fast
Enter search string: quit
```

Sample Run 2

```
Enter source string: There are way too much homeworks in this class end
Enter search string: are+
index: 6 word: are
Enter search string: a.
Enter search string: a*
index: 11 word: way
index: 43 word: class
Enter search string: t+
index: 14 word: too
index: 36 word: this
Enter search string: th+
index: 36 word: this
Enter search string: o*
index: 24 word: homeworks
index: 28 word: homeworks
Enter search string: o.
index: 16 word: too
Enter search string: o**
index: 15 word: too
index: 16 word: too
index: 24 word: homeworks
index: 28 word: homeworks
Enter search string: quit
```

Sample Run 3 (Remember you should store the string up to word END)

```
Enter source string: there are good times
and there are bad times
END
Enter search string: are+
index: 6 word: are
index: 31 word: are
Enter search string: re*
Enter search string: re.
index: 3 word: there
index: 7 word: are
index: 28 word: there
index: 32 word: are
Enter search string: o*
index: 11 word: good
index: 12 word: good
Enter search string: oo*
index: 11 word: good
Enter search string: tim+
index: 15 word: times
index: 39 word: times
Enter search string: Tim+
Enter search string: Oo+
Enter search string: quit
```

Sample Run 4 (You should only ask for the search string when a valid source string is given)

```
Enter source string: invalid_input end
Enter source string: AnyPunctiation..,~#$£> END
Enter source string: + +2 '4 ' 3 END
Enter source string: Correct formatted input string 23 END
Enter search string: C+
index: 0 word: Correct
Enter search string: quit
```

Sample Run 5 (You should only search when a suitable search operator is present. Also ask for search string until QUIT is given)

```
Enter source string: this is a long long homework END
Enter search string: invalid_input
Enter search string: long
Enter search string: wrongoperator$
Enter search string: quit
```

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all take-home exams, unless otherwise noted.

- **How to get help?**

You can use GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

You may ask questions to TAs (Teaching Assistants) or LAs (Learning Assistants) of CS201. Office hours of TAs/LAs are at the course website.

- **What and Where to Submit**

You should prepare (or at least test) your program using MS Visual Studio 2012 C++ (Windows users) or using Xcode (macOS users).

It'd be a good idea to write your name and last name in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). If your name and last name is "Gülşen Demiröz", and if you want to write it as comment; then you must type it as follows:
// Gulsen Demiroz

Submission guidelines are below. Since the grading process will be automatic, students are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be 0.

- Name your submission file as follows:
  - Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
  - Name your cpp file that contains your program as follows: "**SUCourseUsername_THEnumber.cpp**"
  - Your SUCourse user name is actually your SUNet username, which is used for checking sabanciuniv emails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name (**use only lowercase letters**). For example, if your SUCourse username is "**altop**", then the file name should be: **altop_the3*.cpp*** (please only use lowercase letters).
  - **Please submit both cpp and header files you used during homework (strutils etc.).**
  - Do not add any other character or phrase to the file name.
- Please make sure that this file is the latest version of your take-home exam program.

- Submit your work **through SUCourse only**! You can use GradeChecker only to see if your program can produce the correct outputs both in the correct order and in the correct format. It will not be considered as the official submission. You must submit your work to SUCourse. You will receive no credits if you submit by any other means (email, paper, etc.).
- If you would like to resubmit your work, you should first remove the existing file(s). This step is very important. If you do not delete the old file(s), we will receive both files and the old one may be graded.

Grading, Review and Objections

Be careful about the automatic grading: Your programs will be graded using an automated system. Therefore, you should follow the guidelines on the input and output order. Moreover, you should also use the same text as given in the "Sample Runs" section. Otherwise, the automated grading process will fail for your take-home exam, and you may get a zero, or in the best scenario, you will lose points.

Grading:

- There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse time and your computer time may have 1-2 minutes differences. You need to take this time difference into consideration.
- Successful submission is one of the requirements of the take-home exam. If, for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
- If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
- Please submit your **own** work only. It is really easy to find "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the Syllabus or on the course website.

Plagiarism will not be tolerated!

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your take-home exam from the email address provided in the comment section of your announced take-home exam grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the take-home exam tab to see the problem with your take-home exam.
- Download the file you submitted to SUCourse and try to compile it.
- Check the test cases in the announcement and try them with your code.
- Compare your results with the given results in the announcement.

Good Luck!
Anıl Özdemir & Barış Altop & Gülşen Demiröz