Sabanci University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Summer 2017-2018

Homework 3 – The Hard Way
Due: 23 July 2018 11.55pm (Sharp Deadline)

DISCLAIMER:

Your program should be a robust one such that you have to consider all relevant user mistakes and extreme cases; you are expected to take actions accordingly!

Only checking the sample run cases might not be sufficient as your solution will be checked against a variety of samples different than the provided samples; however checking these cases are highly encouraged and recommended.

You can NOT collaborate with your friends and discuss your solutions with each other. You have to write down the code on your own. <u>Plagiarism will not be tolerated</u> AND <u>cooperation is not an excuse</u>!

Introduction

The aim of this homework is to practice on stack structures. In this homework, you will implement a program for finding the common words of two given files. It might seem like a trivial CS201 task at first but you **must** do this by using **<u>dynamic stacks</u>**.

Input to Your Program

Your program will first ask for two file names. Afterwards, it will ask for an option question, to which the user can only respond with a “1” or “2”.

Format of the Inputs

At the very beginning, your program will ask for the name of the first input file. Your program should keep asking the same question until a valid file name is given. The same procedure shall be applied for the second file name as well. Once you get both file names correctly, you can start parsing the information out of the files.

The files may consist of more than one line and there might be empty lines as well. Besides, the words within the files might be separated by spaces, tabs or multiple of these. However, there won't be any punctuation marks and/or digits within the files. They will just consist of a bunch of words. You have some example files with the homework bundle.

Last, your program will ask the user to choose the option "1" or "2". Option "1" means that the common words will be displayed in the order that they appear in the first file, whereas Option "2" means that they will be displayed in the order that they appear in the second file. If the option is invalid, you program should print an appropriate error message and ask again until a valid option is entered by the user. Details of these operations can be inspected in the Sample Runs section.

Data Structure to be Used

You **<u>must use dynamic stacks</u>** for this homework. Indeed, you are not allowed to use any other aggregate data type other than dynamic stacks (**array, vector, queue, static stack etc. may not be used anywhere in your code**). Moreover, you are **<u>not allowed to read the files more than once</u>**. You need to think on how to utilize dynamic stacks to store linear information. We inspect your codes other than testing them with test cases; so any attempt of using an alternative data structure or reading the files more than once will be easily spotted and your grade will be set to zero.

Note: A dynamic stack is a stack which is implemented in a dynamic linked list manner with dynamic memory allocation and deallocation operations. You are not allowed to implement a static stack which uses arrays or vectors internally, for this homework.

Details of the Tasks and Outputs of Your Program

Your program should find the words that appear in both of the files. For this, your program should first read the files and store the words in dynamic stack(s). You may want to design your dynamic stack class in a way that you can store the occurrences of the words along with them, because your program should be displaying that information in the end as well.

Depending on the user's choice, you may display the common words in two different orders. If the user chooses "1", your program will display the common words in the order that they appear in the first file. If option "2" is chosen, the same will apply for the second file. Keeping the information in stacks will help you not to lose track of the order of those words in the files.

In the end, your program will display the words which occur in both of the files and their **<u>minimum occurrence count</u>** (i.e if a word occurs 5 times in a file and 3 times in another file, you should display that the word occurs at least 3 times.) The detail is given in Sample Runs.

You have the implementation of the DynIntStack class that you know from the lectures in the homework bundle; however this class is not fully suitable for your task. You should be deciding on what kind of information you want to store in your dynamic stacks and edit the class methods and such accordingly. Moreover, you **must implement a destructor** for the dynamic stack class such that it will deallocate the dynamically created memory for that dynamic stack object.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are the standard inputs (cin) taken from the user (i.e., like ***this***). You have to display the required information in the same order and with the same words as here.

Sample Run 1

```
This program finds the common words of two files using stacks.
---
Enter the first file name: file
Enter the first file name: incorrect
Enter the first file name: file1_v3.txt
Enter the second file name: file
Enter the second file name: invalid
Enter the second file name: file2_v3.txt
Choose with respect to which file the result will be sorted to (1: first file, 2: second file): 1
---
The word "remember" occurred at least 1 time(s) in both files.
The word "spicy" occurred at least 1 time(s) in both files.
The word "gifted" occurred at least 1 time(s) in both files.
The word "watch" occurred at least 1 time(s) in both files.
The word "caring" occurred at least 1 time(s) in both files.
The word "temper" occurred at least 1 time(s) in both files.
The word "harsh" occurred at least 1 time(s) in both files.
The word "mice" occurred at least 1 time(s) in both files.
The word "representative" occurred at least 1 time(s) in both files.
The word "bell" occurred at least 4 time(s) in both files.
The word "stretch" occurred at least 1 time(s) in both files.
The word "yam" occurred at least 1 time(s) in both files.
The word "theory" occurred at least 1 time(s) in both files.
The word "paper" occurred at least 1 time(s) in both files.
The word "listen" occurred at least 1 time(s) in both files.
The word "profuse" occurred at least 1 time(s) in both files.
The word "nest" occurred at least 4 time(s) in both files.
The word "notebook" occurred at least 1 time(s) in both files.
The word "dime" occurred at least 1 time(s) in both files.
The word "turn" occurred at least 1 time(s) in both files.
The word "face" occurred at least 1 time(s) in both files.
The word "foamy" occurred at least 2 time(s) in both files.
The word "nonchalant" occurred at least 3 time(s) in both files.
The word "muddled" occurred at least 1 time(s) in both files.
The word "health" occurred at least 2 time(s) in both files.
The word "garrulous" occurred at least 2 time(s) in both files.
The word "vanish" occurred at least 1 time(s) in both files.
The word "labored" occurred at least 1 time(s) in both files.
The word "excited" occurred at least 1 time(s) in both files.
```

Sample Run 2

```
This program finds the common words of two files using stacks.
---
Enter the first file name: file2_v3.txt
Enter the second file name: file4_v3.txt
Choose with respect to which file the result will be sorted to (1: first file, 2: second file): aaa
---
Invalid choice
---
Choose with respect to which file the result will be sorted to (1: first file, 2: second file): 0
---
Invalid choice
---
Choose with respect to which file the result will be sorted to (1: first file, 2: second file): -1
---
Invalid choice
---
Choose with respect to which file the result will be sorted to (1: first file, 2: second file): 2
---
```

Sample Run 3

```
This program finds the common words of two files using stacks.
---
Enter the first file name: file3_v3.txt
Enter the second file name: file2_v3.txt
Choose with respect to which file the result will be sorted to (1: first file, 2: second file): 2
---
The word "bell" occurred at least 3 time(s) in both files.
The word "nest" occurred at least 5 time(s) in both files.
The word "lackadaisical" occurred at least 1 time(s) in both files.
The word "yam" occurred at least 1 time(s) in both files.
The word "excited" occurred at least 1 time(s) in both files.
The word "learn" occurred at least 1 time(s) in both files.
The word "jar" occurred at least 1 time(s) in both files.
The word "offer" occurred at least 1 time(s) in both files.
The word "theory" occurred at least 1 time(s) in both files.
The word "soup" occurred at least 1 time(s) in both files.
The word "perform" occurred at least 1 time(s) in both files.
The word "future" occurred at least 1 time(s) in both files.
The word "playground" occurred at least 1 time(s) in both files.
The word "vanish" occurred at least 1 time(s) in both files.
The word "pot" occurred at least 1 time(s) in both files.
The word "lumpy" occurred at least 1 time(s) in both files.
The word "stretch" occurred at least 1 time(s) in both files.
The word "outrageous" occurred at least 1 time(s) in both files.
The word "crash" occurred at least 1 time(s) in both files.
The word "intend" occurred at least 1 time(s) in both files.
The word "terrify" occurred at least 1 time(s) in both files.
The word "important" occurred at least 1 time(s) in both files.
The word "watch" occurred at least 1 time(s) in both files.
```

Sample Run 4

```
This program finds the common words of two files using stacks.
---
Enter the first file name: file3_v3.txt
Enter the second file name: file2_v3.txt
Choose with respect to which file the result will be sorted to (1: first file, 2: second file): 1
---
The word "watch" occurred at least 1 time(s) in both files.
The word "yam" occurred at least 1 time(s) in both files.
The word "excited" occurred at least 1 time(s) in both files.
The word "learn" occurred at least 1 time(s) in both files.
The word "jar" occurred at least 1 time(s) in both files.
The word "offer" occurred at least 1 time(s) in both files.
The word "theory" occurred at least 1 time(s) in both files.
The word "soup" occurred at least 1 time(s) in both files.
The word "perform" occurred at least 1 time(s) in both files.
The word "future" occurred at least 1 time(s) in both files.
The word "playground" occurred at least 1 time(s) in both files.
The word "vanish" occurred at least 1 time(s) in both files.
The word "pot" occurred at least 1 time(s) in both files.
The word "lumpy" occurred at least 1 time(s) in both files.
The word "stretch" occurred at least 1 time(s) in both files.
The word "outrageous" occurred at least 1 time(s) in both files.
The word "crash" occurred at least 1 time(s) in both files.
The word "intend" occurred at least 1 time(s) in both files.
The word "nest" occurred at least 5 time(s) in both files.
The word "bell" occurred at least 3 time(s) in both files.
The word "terrify" occurred at least 1 time(s) in both files.
The word "important" occurred at least 1 time(s) in both files.
The word "lackadaisical" occurred at least 1 time(s) in both files.
```

Some Important Rules

Although some of the information is given below, please also read the homework submission and grading policies from the lecture notes of the first week. In order to get a full credit, your program must be efficient, modular (with the use of functions), well commented and indented. Besides, you also have to use understandable identifier names. Presence of any redundant computation, bad indentation, meaningless identifiers or missing/irrelevant comments may decrease your grade in case that we detect them.

When we grade your homeworks, we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we are going to run your programs in Release mode and **we may test your programs with very large test cases**. Hence, take into consideration the efficiency of your algorithms other than correctness.

How to get help?

You may ask your questions to TAs or to the instructor. Information regarding the office hours of the TAs and the instructor are available at SUCourse.

YOU CAN USE GRADE CHECKER FOR THIS HOMEWORK!

You can use GradeChecker (http://sky.sabanciuniv.edu:8080/GradeChecker/) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

Make sure you upload the sample txt files, class header and cpp files, too.

Grade Checker and the automated grading system use a different compiler than MS Visual Studio does. Hence, you should check the “Common Errors” page to see some extra situations to consider while doing your homework. If you do not consider these situations, you may get a

lower score (even zero) even your program works correctly with MS Visual Studio.
Common Errors Page: http://sky.sabanciuniv.edu:8080/GradeChecker/commonerrors.jsp

Grade Checker can be pretty busy and unresponsive during the last day of the submission. Due to this fact, leaving the homework for the last day generally is not a good idea. You may wait for hours to test your homework or make an untested submission, sorrily..

Grade Checker and Sample Runs together give a good estimate of how correct your implementation is, however we may test your programs with different test cases and **your final grade may conflict with what you have seen on Grade Checker.** We will also **manually** check your code, indentations and so on, hence do not object to your grade based on the Grade Checker results, but rather, consider every detail on this documentation. **So please make sure that you have read this documentation carefully and covered all possible cases, even some other cases you may not have seen on Grade Checker or Sample Runs**. The cases that you *do not need* to consider are also given throughout this documentation.

Submit via SUCourse ONLY! **Grade Checker is not considered as a submission**. Paper, e-mail or any other methods are not acceptable, neither.

The internal clock of SUCourse might be a couple of minutes skewed, so make sure you do not leave the submission to the last minute. In the case of failing to submit your homework on time:

"No successful submission on SUCourse on time = A grade of 0 directly."

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework.

It'd be a good idea to write your name and lastname 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 full name is "Duygu Karaoğlan Altop", and if you want to write it as comment; then you must type it as follows:

```
// Duygu Karaoglan Altop
```

Submission guidelines are below. Since the grading process will be automatic, you are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be zero. The lack of even one space character in the output will result in your grade being zero, so please test your programs yourself and with the Grade Checker tool explained above.

- Name your cpp file that contains your program as follows:

"SUCourseUserName_hw3.cpp"

Your SUCourse user name is actually your SUNet username which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SU e-mail address is **atam@sabanciuniv.edu**, then the file name must be: **"atam_hw3.cpp"**

- Please make sure that this file is the latest version of your homework program.

- You should upload the sample txt files and your class header and cpp files to SUCourse as well.

- Do <u>not</u> zip any of the documents but upload them as separate files only.

- Submit your work **<u>through SUCourse only</u>**! You can use the Grade Checker 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 may visit the office hours if you have any questions regarding submissions.

Plagiarism

Plagiarism is checked by automated tools and we are very capable of detecting such cases. Be careful with that...

Exchange of abstract ideas are totally okay but once you start sharing the code with each other, it is very probable to get caught by plagiarism. So, do <u>NOT</u> send any part of your code to your friends by any means or you might be charged as well, although you have done your homework by yourself. Homeworks are to be done personally and you have to submit your own work. **Cooperation will NOT be counted as an excuse.**

In case of plagiarism, the rules on the Syllabus apply.

Good Luck!
Tolga Atam, Duygu K. Altop