CS201 – Fall 2020-2021 - Sabancı University

Take-Home Exam 4: Spell Check

Due December 13th, Sunday, 23:55 (Sharp Deadline)

Brief Description

In this THE, you will write a simple program that corrects spelling mistakes in a given text file with respect to the word pool stored in another file (see "Structure of Input Files" section for details). The main idea behind this THE is you will read the inputs from files , process them and print out your output on the console. (see "Name and Structure of Output File" section for details). After successfully printing out the output on the console, your program will terminate.

The names of the input files will be taken from the user, and you should perform an input check on these inputs; which will be explained in the "Input Check" section.

Please see the "Sample Runs" section below.

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 (**cpp, header and txt files for this take-home exam**). Additionally, you should submit all of your source files (**cpp and header**) 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.

IMPORTANT!

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

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 mentioned format.

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

Structure of the Input Files

As mentioned before, there are two input text files in this THE. One contains the word pool, and the other one contains the main text that will be corrected. These two files will be input files to your program. (i.e. your program will read some data from both files)

Input Check and Flow

Name of both input files will be entered by the user of your program. At the beginning of your program, the user will be asked to enter an input file name for the word pool. If the user enters a name for a file that does not exist, your program should display an appropriate message and ask for the name of a new input file name until the file is successfully opened. After the first input file is opened successfully, your program will ask for another input file name for the main text. The same file opening control will be performed for this file too. If a user enters a name for a file that does not exist, your program should display an appropriate message and ask for the name of a new input file name until the file is successfully opened.

Word Pool File

The word pool file contains correctly spelled words. There may be several lines in this file, and a single line may contain more than one word. You may also assume that there is at least one word in each line, and please note that the word pool file is not sorted/ordered.

You should note that the word pool is case-insensitive, so it would be useful to convert it to lower case letters to perform necessary operations. You may assume that the word pool is not going to be empty and it doesn’t contain any repeated words.

Main Text File

This file contains the main text which your program should perform spell check on. There may be several lines in this file, and a single line may contain more than one word. You may also assume that there is at least one word in each line.

You should note that the main text is also case-insensitive, so it would be useful to convert it to lower case letters to perform necessary operations. You may assume that the main text file is not going to be empty and it doesn't contain any repeated words.

Important! You may assume that all words in both files contains only the letters of the English alphabet. So, you do not need to make any input check on the words.

The structure of the input files and the associated rules and assumptions explained above are fixed such that you cannot change them or make any other assumptions. **You can assume that the input files will not be empty.** You may examine the sample input files provided in the Google Drive folder of THE-4. Please note that we have multiple input files for multiple scenarios, therefore your code should also work with all these possible outcomes.

Important! There might be any number of white spaces (i.e. blanks) between and after each data item in both files. You should handle it.

Spellcheck

Spellcheck will basically be reading each word from the main text file separately, searching the word pool for corrections and writing the output (either correction(s) or the exact word) to a console.

To do so, you must first begin reading words from the main text file. For each word, you should make a search in the word pool file and compare them if they have the same length. Then the result will be written to the console for each word.

Sample pool words: kedi, balkan, baston, fayton

- If an exact match is found in the pool, the result is the given word exactly.
    - Case: If the input word is "kedi" then since there is an exact match in the Sample Pool, you should output "kedi" to the console.

- Otherwise, the result should be the closest match if the number of spelling mistakes is equal to or less than the half of the given word's length.
    - Case: If the input word is "balkon", since it matches more than half of the characters in "balkan" you should output "balkan".

- Please note that, there may be more than one word that matches with the fewest number of spelling mistakes, in that case the result should be written in parentheses with commas between the matching words.
    - Case: If the input word is "kanton" then you should output: (baston,fayton)

- If there are no matches with spelling mistakes equal to or less than the half of the given word's length, then the result should be the given word in curly brackets.
    - Case: If input word is "kara" you should output: {kara}

Please refer to the Output section to see the details about output's format.
When the spellcheck is finished, display a message (to the screen) that informs the user about the end of the. The message format is as follows:
"`You may find the results below:`"

Structure of Output

Output will have a similar structure to the main text file. For each word in the main text, its spellcheck result is written to the console. Position of this result will be the same as the actual word's position in the main text file. In other words, spellcheck results should be in the same order as taken from the main text. As mentioned before, there may be several white spaces between words in a line, but in the output file there should only be only one white space between spellcheck results in a line.

Some Hints

Maybe you have already realized that the simplest way of processing the input file is reading the data line by line and parsing each line. In order to perform such a processing, the ideal mechanism is to use a combination of getline function and input string streams (istringstream). We have given/will give examples of such a processing in the lecture and in recitations.

Just a reminder: When you are done with a file, do not forget to close it.

Please see sample runs for examples.

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.

Sample Runs

You will find the sample input and output files in the Google Drive folder of THE-4. Below, we provide a sample run of the program that you will develop. The italic and bold phrases are inputs taken from the user. You should follow the input order in these examples.

Sample Run1

pool1.txt

```
baston fayton balkon
balkan
bant jant bal kan
```

text1.txt

```
boston kanton
bardak
pendik
tane kara
```

Command Line Window

```
Please enter a filename for the word pool: pool
Cannot open the word pool file.
Please enter a filename for the word pool: pooool.txt
Cannot open the word pool file.
Please enter a filename for the word pool: pool1.txt
Please enter a filename for the main text: m
Cannot open the main text file.
Please enter a filename for the main text: maintext.text
Cannot open the main text file.
Please enter a filename for the main text: text1.txt
You may find the results below:

baston (baston,fayton)
balkan
{pendik}
(bant,jant) {kara}
```

Sample Run2

Pool2.txt

```
charmender charmeleon
charizard
sandshrew sandslash clefairy
clefable
psyduck golduck poliwag
poliwhirl poliwrath
exeggcute exeggutor
vaporeon
jolteon flareon espeon umbreon
leafeon glaceon sylveon
```

text2.txt

```
claneon flareon
neoneon poliwryyy
politoedd dyspuck yspduck psyduck
churmundur charmender
```

Command Line Window

```
Please enter a filename for the word pool: pool2.txt
Please enter a filename for the main text: text2.txt
You may find the results below:

(flareon,glaceon) flareon
leafeon poliwrath
{politoedd} {dyspuck} psyduck psyduck
charmender charmender
```

Sample Run3

pool3.txt

```
all not from by the be
a is are does do nee
WAS were that this these those
SHALL
gold  fire frost
light dark
old new
dea
LOST found Nor
```

text3.txt

```
Aal that is gold does net glitter
Not all those who wander are last
```

Command Line Window

```
Please enter a filename for the word pool: pool3.txt
Please enter a filename for the main text: text3.txt
You may find the results below:

all that is gold does (not,nee,new) {glitter}
not all those {who} {wander} are lost
```

Sample Run4

pool4.txt

```
kedi


zurafa
koala
Keci kas      kel kis
```

text4.txt

```
koala
koola
kedi

kidi zurafa zulafa kus
```

Command Line Window

```
Please enter a filename for the word pool: pool4.txt
Please enter a filename for the main text: text4.txt
You may find the results below:

koala
koala
kedi

kedi zurafa zurafa (kas,kis)
```

Sample Run5

pool5.txt

```
kelam karar kemal kaleM kagiT
MaMa maYa kefil tefal
```

text5.txt

```
kalem kagit masa kefal
```

Command Line Window

```
Please enter a filename for the word pool: pool5.txt
Please enter a filename for the main text: text5.txt
You may find the results below:

kalem kagit (mama,maya) (kemal,kefil,tefal)
```

General Rules and Guidelines about THE's

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

When submitting your Take-Home Exam to gradechecker and sucourse, you should submit all your cpp, header and txt files.

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_the4*.cpp*** (please only use lowercase letters).
  - **Please submit all .cpp and header (.h) files you used during the take-home exam (ex: strutils prompt date randgen 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