Sabanci University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Summer 2019-2020

Take-Home Exam 1 – Word Seek
Due: 13 July 2020 23:55 (SHARP)

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 from the provided samples; however checking these test cases are highly encouraged and recommended, as well.

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

After this take-home exam deadline, all students will be called for a demo to explain how they approached and solved the problem.

Introduction

The aim of this take-home exam is to practice on and recall matrices. You are asked to implement a game where the user searches for a list of words in a character matrix, just like the one you would do with Sunday morning puzzles. A score will be calculated at the end of the program according to the existence of the words and their directions in the matrix.

Inputs to Your Program

Your program should prompt for an input file name and read the file name from the standard input (cin). This file will have the characters of a matrix. The file could look like this:

```
8 8
A H E A R L O W
L O L Z I L O A
G E S A E S E N
D I S T A N T Y
W H U B P A L O
B R M M I R Q N
B X A U K E B E
G V A N Q Z U K
```

matrix1.txt

If the input file cannot be opened successfully, your program should then ask the user for the file name again and try opening it repeatedly until the file is successfully opened.

While reading the file, **you <u>must</u> store the information in a matrix of chars (vector of vector of chars)**. **<u>Refusing to do so will result in a grade of zero</u>**. In other words, you **<u>must not</u>** read the file more than once or use any other data structure than a vector of vector.

Your program will also read words, for which the existence will be checked within the provided matrix, from the standard input, which are further explained in the next section.

Format of the Inputs

In the first line of the data file containing the matrix of characters, there will be the dimensions of the respective matrix, i.e. the number of rows and the number of columns, in this given order, separated by a single space character. In the remaining lines of this file, there will be the matrix of characters. Here, the characters of the same row will be separated by a single space character, and the rows will be separated by a newline ('\n') character, as seen above in *matrix1.txt*. Also, there will not be any empty lines in this file. You may assume that this file won't contain any characters other than uppercase English letters, spaces, newlines and EOF. You may assume that the row and column count given at the beginning of the file will always be correct. Hence, you do not need to check its correctness, but just read the file as the format suggests.

The words to be searched will be asked to the user through the standard input (cin). This process will continue until either the user enters a word of length less than 3 or the puzzle is cleared. You may assume that there will only be uppercase English letters in the input words, but no spaces, tabs or anything else.

For any further details, you can see the sample runs.

Program Flow and Search

Your program should start by asking the filename for the matrix file. While doing so, your program should make sure that the file with the given name actually exists. Once the file is successfully opened, your program should read the file, store the characters in a **matrix of characters**, and display the content of this matrix.

After that, your program should read words from the user and each word should be searched through the puzzle matrix. Search is only possible horizontally or vertically (diagonal search is not included). For both horizontal and vertical search, you should also check if the word can be found in the reverse direction. If the word is found, then the user will earn some points, which can be calculated as shown in the table below, where `Number_of_Chars` is the number of characters of the found word. Otherwise, i.e. if the word is not found, the user will lose 5 (five) points. Please be reminded that the user starts with 0 (zero) points.

In case that there are multiple occurrences of a word in the matrix, the priority of search directions is as given in the table below, in the decreasing order. For example, an occurrence in regular horizontal direction has the highest priority, whereas an occurrence in reverse vertical direction has the lowest priority. If a word occurs in the same direction more than once at a given time, the one whose starting point is closest to the top of the puzzle should be picked. If a word occurs in the same direction more than once at a given time and there are multiple occurrences whose starting points lay on the same distance from the top of the puzzle, then the one whose starting points is closest to the left boundary of the puzzle should be picked. Long story short, we advise you to scan the matrix from top-left to bottom-right in row-based manner 4 (four) times and look for occurrences in one direction in each scan.

Search Type	**Direction on Matrix**	**Points Earned**
Regular Horizontal	Left to Right	`Number_of_Chars`
Reverse Horizontal	Right to Left	`Number_of_Chars * 2`
Regular Vertical	Top to Down	`Number_of_Chars + 2`
Reverse Vertical	Down to Top	`Number_of_Chars * 2 + 2`

Once a word is found (either horizontally or vertically), its respective row (if found horizontally) or column (if found vertically) should be deleted from the puzzle matrix. This is applied such that the user will not be able to use the same word for the same location again.

After each guess, your program should display the result of the search and the resulting points. Please check the Sample Runs section (below) to see the exact format and prompts. Only after successful guesses, the updated puzzle matrix should be displayed, too.

The End of the Game

The game will end once the user enters a word of length less than 3 (three) or the puzzle is cleared. If the user clears the puzzle (dimensions of the puzzle becomes 0 x 0), then they earn 20 additional points. There is no other condition that stops the game.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are the standard input (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/spaces as here; in other words, there must be an exact match!

We will be automatically grading your take-home exam using GradeChecker, so it is very important to satisfy the exact same output given in the sample runs. You can utilize GradeChecker (http://sky.sabanciuniv.edu:8080/GradeChecker/) to check whether your code 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 **without zipping them**. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

Sample Run 1

```
Welcome to Word Seek.
Please enter a filename for the puzzle: matrix
Failed..
Please enter another filename for the puzzle: matrix1
Failed..
Please enter another filename for the puzzle: matrix1.txt
Success!
Puzzle:
A H E A R L O W
L O L Z I L O A
G E S A E S E N
D I S T A N T Y
W H U B P A L O
B R M M I R Q N
B X A U K E B E
G V A N Q Z U K
Please enter a word: ANYBODY
The word cannot not be found in the puzzle.
5 points are deducted.
Total points: -5.
Please enter a word: ANYONE
Word ANYONE is found.
```

```
8 points are earned.
Total points: 3.
Puzzle:
A H E A R L O
L O L Z I L O
G E S A E S E
D I S T A N T
W H U B P A L
B R M M I R Q
B X A U K E B
G V A N Q Z U
Please enter a word: LOL
Word LOL is found.
3 points are earned.
Total points: 6.
Puzzle:
A H E A R L O
G E S A E S E
D I S T A N T
W H U B P A L
B R M M I R Q
B X A U K E B
G V A N Q Z U
Please enter a word: DISTANT
Word DISTANT is found.
7 points are earned.
Total points: 13.
Puzzle:
A H E A R L O
G E S A E S E
W H U B P A L
B R M M I R Q
B X A U K E B
G V A N Q Z U
Please enter a word: HEAR
Word HEAR is found.
4 points are earned.
Total points: 17.
Puzzle:
G E S A E S E
W H U B P A L
B R M M I R Q
B X A U K E B
G V A N Q Z U
Please enter a word: EASE
```

```
Word EASE is found.
8 points are earned.
Total points: 25.
Puzzle:
W H U B P A L
B R M M I R Q
B X A U K E B
G V A N Q Z U
Please enter a word: NUMB
Word NUMB is found.
10 points are earned.
Total points: 35.
Puzzle:
W H U P A L
B R M I R Q
B X A K E B
G V A Q Z U
Please enter a word: A
Game has ended.
Total points: 35.
```

Sample Run 2

```
Welcome to Word Seek.
Please enter a filename for the puzzle: matrix2.txt
Success!
Puzzle:
Y V X B X X V
R E S O O N E
R R B R R D R
E Y S E X X Y
V E O A V V C
E A O K E E O
R R N U R R O
Y L X P Y Y L
Please enter a word: VERY
Word VERY is found.
6 points are earned.
Total points: 6.
Puzzle:
Y X B X X V
R S O O N E
R B R R D R
E S E X X Y
```

```
V O A V V C
E O K E E O
R N U R R O
Y X P Y Y L
Please enter a word: SOON
Word SOON is found.
4 points are earned.
Total points: 10.
Puzzle:
Y X B X X V
R B R R D R
E S E X X Y
V O A V V C
E O K E E O
R N U R R O
Y X P Y Y L
Please enter a word: VERY
Word VERY is found.
6 points are earned.
Total points: 16.
Puzzle:
X B X X V
B R R D R
S E X X Y
O A V V C
O K E E O
N U R R O
X P Y Y L
Please enter a word: VERY
Word VERY is found.
6 points are earned.
Total points: 22.
Puzzle:
X B X V
B R D R
S E X Y
O A V C
O K E O
N U R O
X P Y L
Please enter a word: COOL
Word COOL is found.
6 points are earned.
Total points: 28.
Puzzle:
```

```
X B X
B R D
S E X
O A V
O K E
N U R
X P Y
Please enter a word: VERY
Word VERY is found.
6 points are earned.
Total points: 34.
Puzzle:
X B
B R
S E
O A
O K
N U
X P
Please enter a word: SOON
Word SOON is found.
6 points are earned.
Total points: 40.
Puzzle:
B
R
E
A
K
U
P
Please enter a word: BREAKUP
Word BREAKUP is found.
9 points are earned.
Total points: 49.
Puzzle is cleared. 20 extra points are earned.
Total points: 69.
Game has ended.
Total points: 69.
```

Sample Run 3

```
Welcome to Word Seek.
Please enter a filename for the puzzle: matrix3.txt
Success!
Puzzle:
Y R R E V E R Y
V E R Y E A R L
X S B S O O N X
B O R E A K U P
X O R X V E R Y
X N D X V E R Y
V E R Y C O O L
Please enter a word: VERY
Word VERY is found.
4 points are earned.
Total points: 4.
Puzzle:
V E R Y E A R L
X S B S O O N X
B O R E A K U P
X O R X V E R Y
X N D X V E R Y
V E R Y C O O L
Please enter a word: SOON
Word SOON is found.
4 points are earned.
Total points: 8.
Puzzle:
V E R Y E A R L
B O R E A K U P
X O R X V E R Y
X N D X V E R Y
V E R Y C O O L
Please enter a word: VERY
Word VERY is found.
4 points are earned.
Total points: 12.
Puzzle:
B O R E A K U P
X O R X V E R Y
X N D X V E R Y
V E R Y C O O L
Please enter a word: VERY
Word VERY is found.
```

```
4 points are earned.
Total points: 16.
Puzzle:
B O R E A K U P
X N D X V E R Y
V E R Y C O O L
Please enter a word: ONE
Word ONE is found.
5 points are earned.
Total points: 21.
Puzzle:
B R E A K U P
X D X V E R Y
V R Y C O O L
Please enter a word: VERY
Word VERY is found.
4 points are earned.
Total points: 25.
Puzzle:
B R E A K U P
V R Y C O O L
Please enter a word: COOL
Word COOL is found.
4 points are earned.
Total points: 29.
Puzzle:
B R E A K U P
Please enter a word: BREAKUP
Word BREAKUP is found.
7 points are earned.
Total points: 36.
Puzzle is cleared. 20 extra points are earned.
Total points: 56.
Game has ended.
Total points: 56.
```

Some Important Rules

Although some of the information is given below, please also read the take-home exam 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 take-home exams, we pay attention to these issues. Moreover, in order to observe the real performance of your code, 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 SHOULD USE GRADE CHECKER FOR THIS TAKE-HOME EXAM!

You should 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 .txt files, too.

GradeChecker 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 take-home exam. If you do not consider these situations, you may get a lower score (even zero) even if your program works correctly with MS Visual Studio.

Common Errors Page: http://sky.sabanciuniv.edu:8080/GradeChecker/commonerrors.jsp

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

GradeChecker 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 GradeChecker.** We will also **manually** check your code (comments, indentations and so on), hence do not object to your grade based on the GradeChecker 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 GradeChecker 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, either.

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 take-home exam on time:

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

What and where to submit (PLEASE READ, IMPORTANT)

You should test your program using GradeChecker. We will use the same UNIX based C++ compiler that Grade Checker uses for grading your take-home exam.

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 GradeChecker tool explained above.

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

  "SUCourse+UserName_the1.cpp"

  Your SUCourse+ username 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_the1.cpp"**

- Please make sure that this file is the latest version of your take-home exam program.

- You should upload all the .txt files to SUCourse+ as well.

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

- Submit your work **<u>through SUCourse+ only</u>**! You can use the 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+.

- 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.

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 take-home exam by yourself. Take-home exams 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