Sabanci University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Summer 2021-2022

Take-Home Exam 1 – Searching for Words
Due: 2 August 2022 11.55pm (SHARP)

DISCLAIMER:

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 cases is 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. Plagiarism will not be tolerated AND cooperation is not an excuse!

Introduction

The aim of this take-home exam (THE) is to recall CS201 material and practice on matrices (i.e., two dimensional vectors). You are asked to search for a list of words in a character matrix, just like you would do with Sunday morning puzzles. When searching for a word, you will calculate a score based on the existence/absence of that word in the matrix (please refer to the "Calculating the Score" section for more details).

Inputs to Your Program

All the inputs in this THE are given to the program through the standard input (cin). Your program will start by reading a positive integer number, *rows* that indicates the number of rows in the matrix of characters; followed by "*rows*" number of lines which represent the actual matrix of characters (each input line/row is guaranteed to have the same number of characters, i.e., all matrix rows will have the same length, you don't need to check for that). Please note that, while reading the matrix, **you must store it in a vector of vectors. *Otherwise, your submission will be graded as zero (0)***. Then, your program will read another positive integer number, *words* which indicates the number of words that will be read and checked for their existence in the matrix; followed by "*words*" number of lines where each line contains one word. The inputs for the number of rows and the number of words will always be positive integers; meaning that you don't need a validity check on these inputs.

Format of the Inputs

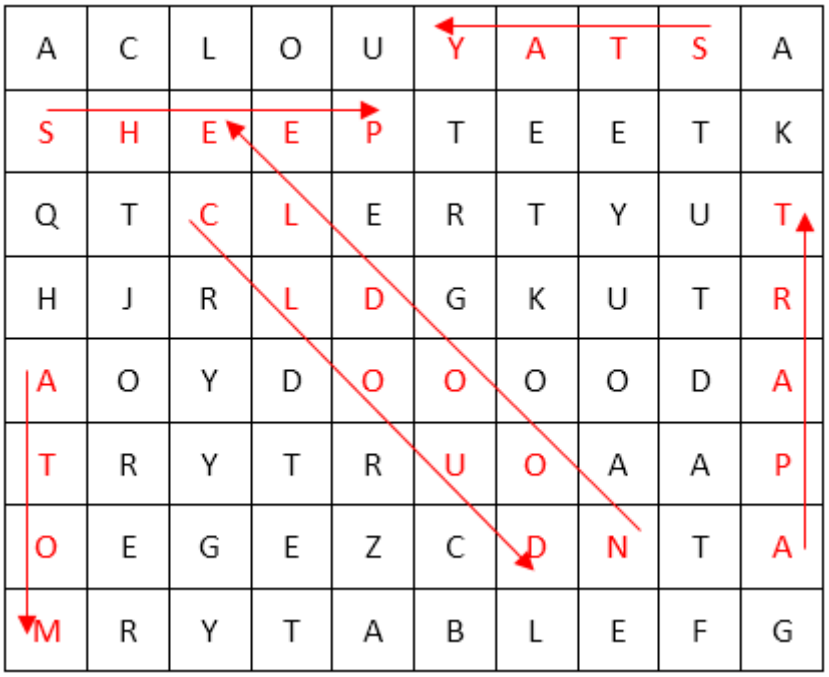
All the character inputs (and words) in this THE are uppercase English letters with no spaces, tabs or special characters. You can assume that this will be true for all the test cases that will be used, hence you do not need to do any extra check. Regarding the words, you can also assume that all the words that will be used in all the test cases will be of length greater than two (2).

Task of Searching

This section clarifies the algorithm you need to develop.

After reading the matrix and the words to search for, your program will search for each word (that it gets from the standard input) in the 2D vector that your program had constructed from the matrix.
This search can be done vertically, horizontally or diagonally (*Note*: anti-diagonal[1] direction is not included for the search). It is also possible to go in the backward directions of these three directions as well.

The following illustration shows possible directions of search in the matrix:



More explicitly, your program can do the search to the north ↑ (APART), south ↓ (ATOM), east → (SHEEP), west ← (STAY), northwest ↖ (NOODLE) or southeast ↘ (CLOUD) of the first letter of the word to be found. You should base your search technique on this rule. Combinations of different directions are not allowed within a word, meaning that you should be searching a word in *one direction* only. A word that your program is searching for may not be in the matrix at all and this is also a case that your program should handle (kindly refer to the sample runs and scoring criteria).

[1] Anti-diagonal is the direction which is perpendicular to the main diagonal of the matrix.

You can also assume that a particular word will not occur in the matrix more than once, so you can safely stop the search regarding that word when the program finds it once.

Don't bother: The words may intersect in the matrix (i.e., two words may use a common letter) and this is totally okay for the game. You do not need to check against this. An example for this case exists in the example matrix given above, where SHEEP and NOODLE share the letter 'E' successfully. Besides, even one word can contain another (APART and PART can both be accepted in a game).

Calculating the Score

Each word that your program can or cannot find in the matrix affects the score. Following are the rules of the game to calculate the score:

1. Words that lay horizontally or vertically increase the score by their number of letters (i.e., the word 'MONEY' would contribute 5 points).
2. Words that are found (vertically, horizontally, or diagonally) and are longer than 5 characters contribute 2 extra points to the overall score.
3. Words that lay diagonally increase the score by their number of letters multiplied by two (i.e., the word 'BIKE' would contribute 8 points (=4*2)).
4. For each word that cannot be found in the matrix, the total score will be deducted by 5.

Please note that the total score is allowed to go below zero (please check test case #5).

Output of Your Program

The output of your program should be an integer number which represents the final score of the game (it might be positive, negative or zero based on the score calculation). You can see the format of the output in the sample runs. Please note that your program's output should be exactly the same as given in the sample runs, i.e., no extra spaces, empty lines, text output, etc.

Sample Runs

Below, we provide some sample runs of the program that you will develop along with an explanation for them. Your program should read the inputs as in the *Input* column and should print the final score as an integer value as in the *Output* column.

TC#	Input	Output	Explanation
1	5 HJRLDGKU AOYDOOOO TRYTRUOA OEGEZCDN MRYTABLE 1 ATOM	4	5 represents the number of rows of the matrix, therefore it is followed by 5 lines of input. Then, 1 represents the number of words that will be searched for in the matrix, therefore it is followed by 1 line of input. The output (score) is 4, as follows: ● The word *ATOM* is found vertically in the matrix as highlighted in the cell on the left. So, as per rule number 1 for calculating the score, the score is increased by the number of letters in the word *ATOM*. Thus, the final score is 4 as shown in the output column.
2	9 QJCMPVSOY FDOBORFEC UANOTPTDZ KLSLSCTND TXTMFJXGT QERIOZJSW LYUTYMIEE BLCNDHDUG TCTFGVYUB 2 STOP CONSTRUCT	15	The output (score) is 15 (= 4+9+2), as follows: ● The word *STOP* can be found vertically (in reverse direction) as highlighted in the sample input. ● For the word *CONSTRUCT* (length = 9), it can be found vertically as highlighted in the sample input. ● And the last +2 points because the word *CONSTRUCT* is longer than 5 characters (according to rule # 2).
3	7 HRPIEQGVEOZKVEVUFU LYOURYLAUEHLOZEYPB JTXNYIANINCLUDEHMB CARJGZDLAFRBWPJZZV QFQFKWEZNETTSOPHTN HXEUGKDZGLGSQGLWRK WOSCHOLARSHIPXFNQS 3 CAR INCLUDE SCHOLARSHIP	25	The output (score) is 25 (=3+7+11+2+2), as follows: ● The words *CAR*, *INCLUDE*, and *SCHOLARSHIP* can be found horizontally. Therefore the score is the addition of their lengths ● +2 points are added for finding the word *SCHOLARSHIP* and +2 points are added for finding the word *INCLUDE* as they are longer than 5 characters (rule #2).
4	9 KEEPTFWSM KYZCZISRS BRKEYFAST SNIJQWIRT NECTIKLMH WLRPEWLNT YIAMAERCS STWUDYNOY KELMQZGAH 3 KEEP ELITE CREAM	14	The output (score) is 14 (=4+5+5) as, follows: ● The word *ELITE* can be found vertically. The word *KEEP* can be found horizontally. ● The word *CREAM* can be found horizontally (in reverse direction). Therefore the score is the addition of their lengths.

TC#	Input	Output	Explanation
5	9 KEEPTFWSM KYZCZISRS BRKEYFAST SNIJQWIRT NECTIKLMH WLRPEWLNT YIAMAERCS STWUDYNOY KELMQZGAH 6 GOOGLE KEEP EMAIL ELITE CREAM UNIVERSITY	-1	The output (score) is -1(=-5+4-5+5+5-5), as follows: ● The words *UNIVERSITY, EMAIL,* and *GOOGLE* were not found in the matrix, therefore the score is deducted by -5 for each one of them. ● Note: this is the same input matrix as the above test case, but there are extra three words to be searched for that happen not to be in the matrix.
6	5 SEARCH DAPIJY QTVAMK NRAELQ LGIFTZ 5 SAVE SEARCH AIM LEARN GIFT	31	The output (score) is 31(=4*2+(6+2)+3*2+5+4), as follows: ● The words *SEARCH* and *GIFT* can be found horizontally. ● The word *LEARN* can be found horizontally in reverse direction. ● The words *SAVE* and *AIM* can be found diagonally. Therefore, each one of them contributes to the overall score by their length multiplied by 2 (rule #3)
7	14 IJRFGRRSHLTEVC YDWOTGRRUWFAGF DIEJIGSETTSFLW LWTNFPGGLBCFBJ LBWGTPXKXKBXTK WYRIZIPQKLZXDL VGYZNGFRVMCBWW QDVPDMRIYRLPMT URCCQKTVCQXALH LAEUBNPQTAGEZN LJZSQMWLPLTSRA QMZLUKHVVBIIMF FCYZSLAAJLGYOK DMOGJKTMMCAEPN 4 ESTABLISHMENT IDENTIFICATION GYM RESULT	45	The output (score) is 45(= -5 + (14*2+2) + (3*2) + (6*2+2)), as follows: ● The word *GYM* can be found diagonally in the reverse direction. Therefore it contributes to the score with 6 (=2*3), according to rule #3. ● The word *IDENTIFICATION*, can be found diagonally therefore it contributes to the score with 28 (=14*2), according to rule #3. Also, it is longer than 5 characters. Therefore it contributes to the score with 2 extra points (rule #2). ● Same for the word *RESULT*, it can be found diagonally and is longer than 5 characters. Therefore it contributes to the score with 14 (=2*6+2) ● The word *ESTABLISHMENT* can not be found in the matrix. Therefore, the score is deducted by 5 points.

Some Important Rules

In order to get full credit, your program must be efficient, modular (with the use of functions), well commented and properly 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 THEs, we pay attention to these issues. Moreover, **we may test your programs with very large test cases**. Hence, take into consideration the efficiency of your algorithms other than correctness.

Sample runs give a good estimate of how correct your implementation is, however, we will

test your programs with different test cases and **your final grade may conflict with what you have seen on CodeRunner**. We will also **manually** check your code, indentations and so on, hence do not object to your grade based on the **CodeRunner** 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 CodeRunner or the sample runs**. The cases that you *do not need* to consider are also given throughout this documentation.

Submit via SUCourse ONLY! Paper, e-mail or any other methods are not acceptable.

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 THE on time:

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

What and where to submit (PLEASE READ, IMPORTANT)

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 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
```

You should copy the full content of the .cpp file and paste it into the specified “Answer” area in the relevant assignment submission page on SUCourse. ***Please note that the warnings are also considered as errors on CodeRunner, which means that you should have a compiling and warning-free program***.

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 (0). Any tiny change in the output format will result in your grade being zero (0), so please test your programs yourself, and against the sample runs that are available at the relevant assignment submission page on SUCourse.

In the CodeRunner, there are some visible and invisible (hidden) test cases. You will see your final grade (including hidden test cases) before submitting your code. There is no re-submission. You don't have to complete your task in one time, you can continue from where you left last time but you should not press submit before finalizing it. Therefore, you should make sure that it’s your final solution version before you submit it. Also, we still do not suggest that you develop your solution on CodeRunner but rather on your IDE on your computer.

You may visit the office hours if you have any questions regarding submissions.

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.

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 NOT send any part of your code to your friends by any means or you might be charged as well, although you have done your THE by yourself. THEs 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!
Ahmed Salem, Duygu Karaoğlan Altop