In this assignment you will implement (that is, write a working program for) two algorithms based on the paradigm of dynamic programming.

Please read these instructions carefully before posting any questions to Piazza. We hope all clarification questions you have will already be answered here. If not, please read previous questions on Piazza before posting new ones.

Reread Problem Set 4 (and perhaps also the solutions to Problem Set 4, posted on October 16) before beginning this one.

1. [15 marks]

   Implement your algorithm to solve the word search problem which you developed in Problem Set 4, Question 3, in one of C++, Java, or Python. You will be marked on correctness and efficiency.

   Your program should read from standard input and write to standard output.

   Your program should expect, as input, the following:
   - a line containing a single word \( w \) (with no spaces or special characters). The line can be arbitrarily long, but will be less than 1000 characters.
   - then a blank line
   - then a number of lines representing a square matrix of characters, with one row per line. Each line will have the same number of characters, but it might be different from \( w \). The number of rows will be the same as the number of characters in the first row. Every line will be less than 500 characters.
   
   (Don’t bother checking for inputs that don’t adhere to the specs. This is not a software engineering course!)

   The output should be either the single number 0 – indicating not found – or the single number 1 – indicating found.

   For example, if the input is
the output should be

1\n
Here the \n denotes a carriage return.

Your main program should be called A5Q1.java (if it is in Java) or A5Q1.cpp (if it is in C++) or A5Q1.py (if it is in Python).

2. [15 marks]

The same as above, except now your algorithm should work for Assignment 4, Question 4. If you did not succeed in solving this problem, read the solutions (available on LEARN) carefully before proceeding, and make sure you understand the idea.

Your program should expect, as input, the following:
– a line containing a word (with no spaces or special characters)
– then a blank line
– then a number of lines representing a square matrix, with one row per line.

(Don’t bother checking for inputs that don’t adhere to the specs.)

The output should be either the single number 0 (followed by \n as before) – indicating not found – or four numbers: the row and column (starting from index 1 for both) where the starting position of the word was found, and then the row and column where the end position of the word was found. (If there are multiple occurrences of the word, you only need to return the four numbers corresponding to one single occurrence.)

For example, if the input is

HORSE
AABCDE
BAHOT9
DhcRS4
aQAAE6
321458
EFGHIJ

the output should be

2 3 4 5\n
Again, the \n denotes a carriage return.
There should be exactly one space between each of the numbers. Your main program should be called A5Q2.java (if it is in Java) or A5Q2.cpp (if it is in C++) or A5Q2.py (if it is in python).

Other information
If you use Java, the main method must be inside a public class called “A5Q1” (for Problem 1) and “A5Q2” (for Problem 2). Note that capitalization and spelling of these names are extremely important for us to properly mark your assignment. If you don’t follow the naming conventions we list here, you risk getting a 0.

Your solution must be entirely original. You may not use any code for these problems or solutions that you find online, other than the solutions to the last problem set posted on the course web page. Of course, you can use any standard built-in features of the language you choose, (including, for example, the built-in arithmetic operations like +, -, *, /, <, >, %, etc.), vectors, strings, Java.Util.ArrayList, Java.Util.Scanner, built-in math functions, std::stack, ceiling and max functions).

Submission
Only submit a solution in one of the two programming languages; if you submit multiple times, we will only mark the last one. We will compile your program in the Linux environment using the command

g++ -std=c++11 A5Q1.cpp   (for C++)
or
javac A5Q1.java   (for Java)
or
python A5Q1.py   (for python),
and similarly for A5Q2.

Please make sure your code compiles and runs properly on the student environment before submitting it (preferably before the due date, in case there are any problems).
Submit your solution through Marmoset: at
https://marmoset.student.cs.uwaterloo.ca
Submit only one file for each question. Name your code files as discussed above.

Once you submit your solution, you can see whether your program passes the public tests, which consist of 1 or 2 easy test cases. This is for you to test your output format. Don’t trust the public test cases exclusively! You are strongly recommended to test your code with your own test cases. Your final mark will be the marks for your last submission, tested by the public test cases and others (not public) that we design.

Marking
Your results will be automarked, so be sure that your output is precisely in the right form.

If your program takes more than 10 seconds of cpu time, it will time out and we will stop running it. Unless you are doing something very, very wrong, this is unlikely to happen.

Here's how you will be evaluated on this part. We will test both correctness and efficiency.

For correctness, we will run your program on some inputs and check that you answer correctly. You get 10 marks if your program passes all our tests, and 0 otherwise. We won’t check things like empty input.

For efficiency, we will use the running time as a proxy. You only get marks for efficiency (up to 5) on a problem if you received full marks for correctness on that problem. Your algorithm for Problem 1 should run in $O(n^2)$ time, where the matrix is $n \times n$. Similarly, your algorithm for Problem 2 should run in $O(n^3)$ time.