PROBLEM SET

This exercises are intended to show ***how*** students face a given problem, in both small and big scales. It does not matter if a problem is ***actually solved*** since main objective is to stimulate critical thinking on parallel ways of problem solving.

Yet this problems are small in scale, both represents actual challenges on real life topics, used for cryptography and DNA sequencing, for instance.

Enjoy it! Try to solve and test. Test cases are provided per request

PROBLEM A

Longest Common Subsequence

Statement

A Common Substring of two given strings is a series of characters in which all elements appears in both sequences in the same order. For instance, both “aall” and “rel” are common subsequences of “parallel” and “armalel”

Given a sequence of symbols $S=\langle a_0,a_1,...,a_n\rangle$, a subsequence S’ of S is obtained by removing zero or more symbols from S. For example, given $K=\langle a,b,c,d,e\rangle$ then $K'=\langle b,d,e\rangle$ is a subsequence of K.

A Longest Common Substring (LCS) of two strings A and B is a substring that has maximum length and, of course, is a substring of both A and B. Finding a longest common subsequence of two distinct sequences is a very important challenge to various computer science areas, for instance genetics and cryptography. Hint: The difference in size between two such sub-sequences is a generalization of Levenshtein’s distance

Write a program to calculate the size of the Longest Common Subsequence between two sequences. You don’t need to print the actual sub-strings

Input Format

The program must read two sequences from standard input, containing only capital letters and numbers. Strings can be as long as 5000 characters

Output Format

A number showing the size of the Longest Common Substring.

Example

Input:
```
QWAgN5LKPR
2A5KP7R
```

Output for the input example:
5

REMARKS

1) How different would be your solution… if you know some pattern of the LCS?
2) … if you have to ***print*** actual subsequences of maximum length?
3) … If you have to find the LCS over an arbitrary number of strings?
4) … if you have to deal with an arbitrary increasing number of strings UNTIL find a given maximum number for LCS?

PROBLEM B

On solving weird puzzles

Statement

Many real-world problems are theoretically similar, or can be mapped to games as “crosswords”. In this game, the player try to find as most words as possible from a “letter soup”, a bi-dimensional set of characters where words can be constructed from left to right, right to left or in diagonal way

Thus, given a set of patter words, the problem lies in find such words into the character matrix. In this version, words can be found in one of eight possible directions (assuming each word is not reversible):

1. left-to-right horizontally
2. left-to-right 45 degrees upperwise
3. bottom-to-top vertically
4. right-to-left 45 degrees upperwise backward
5. right-to-left horizontally
6. right-to-left 45 degrees downwise backward
7. top-to-bottom vertically
8. left-to-right 45 degrees downwise

Each word can be more than once in the matrix. This kind of problem is somehow similar to signal processing, when tools searches for patterns inside waves (sound, music, seismic waves, etc)

Input Format

In the first line, there are three numbers, indicating:

1. The number of pattern words
2. The length (in characters) of each line of the puzzle
3. The number of lines of the puzzle

Then, the program reads a series of patter words, an then a matrix of characters. Characters can be assumed as only uppercase characters from English 'A' to 'Z'. Pattern words can be as long as 100 characters. Letter matrix can be as much as 5000 lines of as much 5000 letters each.

You can assume this:

1. Each word is not reversible (for instance: 'AABAA' is not possible)
2. If a word if subset of other, you only must show a response for the longest one if needed.
   For instance, if words 'AAB' and 'AABCD' are in the problem, you can report a result if you find 'AAB'. BUT, if 'AABCD' appears, you must take the latest.

3. Each word can appear more than once

Output Format

You must show an output with all the appearances of each word, followed by three numbers, indicating the coordinates for the first character, and the type of direction the word is into the puzzle. Assume the first character of first line has coordinates 0 0.

Example

Input:

```
4 20 10
ROBIN
GILBER
CARLITO
OBI
AAAAAARAAGILBOAAAAAA
AAAAAROBINAAAATAAAAA
AAAAAABAAAAAACAIAAAA
RAAGAAIAAAAAAAAALAAA
EAAAIANAAAAAARAAARAA
BAAAALAIAAAAALAAAAAA
LAAAAABAAREBLIGAAAAC
IAAAAOAEAAAAATAAAAAA
GAAAAABARAAAAOAAAAAA
AAAAAAAIAAAAAAAAAAAA
```

Output for the input example:

```
ROBIN 5 1 1
ROBIN 6 0 7
GILBER 3 3 8
GILBER 0 8 3
GILBER 14 6 5
CARLITO 9 6 4
CARLITO 13 2 7
OBI 5 7 2
OBI 5 7 8
```

REMARKS

1) How different would be your solution… if words can be reversed?
2) … if you have to ***print*** responses in a given order?
3) … If you have to find the solutions dynamically over an undetermined number of lines?
4) … If there can be accepted 'jumps' inside words? For instance CARLI**X**TO?