Gorilla
Problem ID: 5gorilla

Introduction

How closely related are humans and gorillas? It is well known that out DNA-sequencies are very similar. But how do we actually know this? How do we even measure the similarity between two strings of DNA. And speaking of strings, how do we know how similar two strings are? Is it possible to use almost the same algorithm to align all kinds of strings? Of course not, guitar strings are aligned in a completely different way.

Aims

The goals of the lab are:

- Implementing a DP-algorithm.
- Debugging your code.
- Structuring your code in a logical fashion.
- Parsing a bit messy input.

Problem formulation

You are given two strings, which can be viewed as words or DNA-sequences. Furthermore you are given a matrix with the cost of aligning each pair of letters, i.e. the gain of aligning "A" with "B" and so on. Given these strings find an optimal alignment, such that the total gain of aligning the strings in maximized.

When aligning the two strings you are allowed to insert certain "*"-characters in one or both of the strings at as many positions in the string as you like. Each such insertion can be done to a cost of -4. No letters in either of the strings can be changed, moved or removed – the only allowed modification is to insert "*".

Input

The first line contains a number of space-separated characters, $c_1, \ldots, c_k$ – the characters that will be used in the strings. Then follows k lines with k space-separated integers where the j-th integer on the i-th row is the cost of aligning c_i and c_j. Then follows one line with an integer Q ($1 \leq Q \leq 10$), the number of queries that you will solve. Then follows Q lines, each describing one query. Each of these lines contain two space-separated strings, the strings that should be aligned with maximal gain. It is guaranteed that each string in all queries has size at most 3500.

Output

The output should contain exactly one line per query. On each of these lines two strings should be given, the strings from the input possibly with some "*" inserted. The strings have to be given in the same order as in the input.

Examination and Points of Discussion

To pass the lab make sure you have:

- Have successfully implemented the algorithm with the correct time complexity.
- Have neat and understandable code.
- Have descriptive variable names.
- Have filled in the blanks in the report.
- Have run the check_solution script to validate your solution.

During the oral presentation you will discuss the follwoing questions with the lab assistant:

- Is your solution recursive or iterative?
- What is the time complexity, and more importantly why?
- What would the time complexity of a recursive solution without cache be?
- Can you think of any applications of this type of string alignment?
- What could the costs represent in the applications?