**Problem "Search filters"**

The main topics of problem "Search filters" are: algorithms on strings, data structures. Possible partial solutions are based on searching for a substring in a string, iterating over all variants, hashing.

Note that since the server name and partition name are no more than 5 parts each, there are only a small number of filters for each address that it can potentially fit. In particular, the address is suitable for filters, where the server filter matches the server name or contains a number of its final parts, preceded by an asterisk, a total of no more than 6 options (without an asterisk, leave 1, 2,..., 5 end parts). Similarly, the partition filter can either match the name of the server, or contain a number of its initial parts, followed by an asterisk, a total of no more than 7 options (without an asterisk, leave 0, 1, 2,..., 5 initial parts).

It turns out that for each address there are no more than 42 different filters under which it can fit.

By placing the filters in the appropriate data structure in which you can quickly search for the presence of a filter, such as in a forest or in a structure std::set, we can check for a filter in O(1) or o(*log n*).

Note also that the condition does not guarantee that all filters are different, and the solution should take into account that there may be several identical filters.

To solve subtask 1, it is enough to randomly check for each filter, after removing the end stars, whether it is a substring of the address. The most effective solutions should act in one of two ways: either quickly check the occurrences of a substring in a string, for example, by the *Knuth-Morris-Pratt algorithm*, or take into account that the" junction " of the server name and the partition name is unique and you can only check the occurrences where it is correctly located. In tests to subtask 2 for each address there can be only one filter under which it approaches and completely coinciding with it. You can apply any standard data structure to the solution, for example, add all filters to map<string, int> and search for the number of suitable filters by one query to this structure.

To solve subtask 3, you must apply the approach described in the main solution above. The time limits in this problem are quite strict, various non-asymptotic inaccuracies in the implementation or the unsuccessful use of hashing for string comparison lead to the fact that some tests may not pass, such solutions receive only a partial evaluation.