Problem A. Alice and Bob (and string)

You may notice that suffix automaton will be nothing but graph of described game. One can determine winning states by dfs. We say that the state is winning if there is transition to the losing state from it and it is losing if there is not. Now with dp on a directed acyclic graph one can count how many there are paths that lead to one of the winning states. Further, one can use this information to find the k-th lexicographically winning path in O (path length). Characters written on this path is the answer to the problem.

Problem B. Alice and Bob (and string) 2

Let's reverse string S, construct suffix automaton for it and calculate winning states like in previous problem. Now you can use the fact that suffix link tree in automaton is exactly suffix tree of reversed string (in this case it actually will be initial string because it became reversed). All strings on the same edge are winning or losing in the same time (see part about relation of automaton and suffix tree in lecture). Using this information, you can find the k-th winning string with dfs on suffix tree.

Problem C. Substring

Let the total length of words from queries be L. It means that amount of distinct lengths of such strings is O(sqrt(L)). Let's construct Aho-Corasick for this set of words. Additionally to standard suffix links, let's make additional link from each state (we will call it terminal link) to the closest terminal state which is reachable by suffix links. Now let's in each state create list of prefixes of text where string which corresponds to this state occurs as suffix. Initially all such lists will be empty.

Now let's feed text's characters to automaton one by one. After adding of each character we will have some state in automaton, let's move through its terminal link path and add current prefix to lists of all states in this path. Hence for each word we will find its list of occurrences. Now we can answer every query using binary search to check whether there is "good" position in list.

Problem D. Refrain

For each state of suffix automaton it is required to calculate the power of its right context (which equals to the number of occurrences of strings from this state in initial string). Then take the maximum product of length of largest string accepted by state and power of right context.

Problem E. Average Common Prefix

In the last analysis of the contest we told you how to use hashes to compare two strings in logn. With this comparison, we can sort an array of cyclic shifts in n * log n * log n. Then, for each pair of adjacent strings in the sorted array yo have to find the greatest common prefix in logn with hashes.

Problem F. Bacon's Cypher

This task is educational one. It is just a subproblem of problem from previous contest.

Problem G. Mnemonics and Palindromes 3

We will not analyze this problem because it is very simple. There is no more than 6 strings of such kind.