IOI '99 P2 - Hidden Codes

Time limit: 1.0s **Memory limit:** 1G

IOI '99 - Antalya-Belek, Turkey

A set of code words and a text are given. The text is supposed to contain a message made up of the code words embedded in the text in a peculiar (and possibly ambiguous) way.

The code words and the text are sequences made up of the upper and lower case letters of the English alphabet only. Case-sensitivity is assumed. The *length* of a code word is defined in the usual way: For example, the code word **ALL** has length 3.

The letters of a code word do not have to occur consecutively in the given text. For example, the code word **ALL** will always occur in the text within a subsequence in the form of **A**u**L**v**L** where u and v denote arbitrary (possibly empty) sequences of letters. We say that **A**u**L**v**L** is a *covering sequence* for **ALL**. In general, *a covering sequence* for a code word is defined as a substring of the text such that the first letter and the last letter of the substring are the same as those of the code word and it is possible to obtain the code word by deleting some (possibly none) of the letters of the substring. It should be noted that a code word may occur within one or more covering sequences or may not occur in the text at all, and that a covering sequence may be a covering sequence for more than one code word.

A covering sequence is identified by its start position (position of its first letter) and its end position (position of its last letter) in the text. (The first letter of the text is at position 1.) We say that two covering sequences, say c_1 and c_2, *do not overlap* if the start position of c_1 is greater than ($>$) the end position of c_2 or vice versa. Otherwise we say that the two covering sequences *overlap*.

To extract the message hidden in the text you undertake the task of forming a *solution*. A solution is a set of *items*, each consisting of a code word and a covering sequence for this code word, so that the following conditions are all satisfied:

1. the covering sequences do not overlap with each other
2. a covering sequence does not exceed $1\,000$ in length
3. the sum of the lengths of the code words is maximal. (Each item contributes the length of the code word it contains to the sum.)

In case there is more than one solution you are required to report only one solution.

Input Specification

The first line of input contains the number of code words, N $(1 \le N \le 100)$. Each of the following N lines contains one code word. Each code word has a numerical ID. IDs are assigned in increasing order starting from 1 (hence the first code word has the ID 1, the second 2, and so on). No code word has a length greater than 100.
The final line of input is the text, whose length is at least 1 character and no more than $1\,000\,000$ characters.

We say that a covering sequence c for a code word w is *right-minimal* if no proper prefix of c (a proper prefix is an initial subsequence of c that is not equal to c) is a covering sequence for w. For example, for the code word **ALL, AAALAL** is a right-minimal covering sequence whereas **AAALALAL** is also a covering sequence, but not right-minimal.

It is guaranteed that in the given text:

1. for each position in the text the number of right-minimal covering sequences containing that position does not exceed $2\,500$
2. the number of right-minimal covering sequences does not exceed $10\,000$.

Output Specification

The first line of output should contain the sum obtained by your solution. Each of the following lines will determine an item in your solution. A line consists of three space-separated integers: i, s, and e. Here i is the ID-number of the code word that occurs within the covering sequence identified by the start position s and end position e. The order of the output lines that follow the first line is not important.

Sample Input

```
4
RuN
RaBbit
HoBbit
StoP
StXRuYNvRuHoaBbvizXztNwRRuuNNP
```

Sample Output

```
12
2 9 21
1 4 7
1 24 28
```

(Remark: The hidden message that could be extracted from the solution is `RuN RaBbit RuN`. (An alternative solution would yield `RuN HoBbit RuN`). Be reminded that the message is not to appear in the output.)