HIDDEN CODES

PROBLEM

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 subsequence of the text such that the first letter and the last letter of the subsequence 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 subsequence. 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:

a) the covering sequences do not overlap with each other;
b) a covering sequence does not exceed 1000 in length;
c) 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 are more than one solution you are required to report only one solution.

ASSUMPTIONS

- $1 \leq N \leq 100$ where N is the number of code words.
- The maximum length of a code word is 100 letters.
- $1 \leq$ length of the given text $\leq$ 1,000,000 letters.

- We say that a covering sequence *c* for a code word *w* is *right-minimal* if no proper prefix of *c* (a *proper prefix* of *c* 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
  a) for each position in the text the number of right-minimal covering sequences containing that position does not exceed 2500;
  b) the number of right-minimal covering sequences does not exceed 10,000.

INPUT

There are two input text files: **words.inp** and **text.inp**. The `words.inp` file contains a list of code words and `text.inp` contains the text.

- The first line of `words.inp` contains the value of *N*. Each of the following *N* input lines contains a code word which is a sequence of letters without any blank in between. The code words are identified by their order of appearance in the `words.inp` file: Integers 1 through *N* serve as the id-numbers for the code words.
- The `text.inp` file consists of a sequence of letters (terminated by an end-of-line character followed by the end-of-file symbol). This file does not include blanks.

RECOMMENDATION FOR PASCAL PROGRAMMERS

You are advised to declare the input file type as `text`, as opposed to a typed file for reasons of efficiency.

OUTPUT

The output must be a text file named **codes.out**.

- The first line will contain the sum obtained by your solution.
- Each of the following lines will determine an item in your solution: The line consists of three integers *i, s, 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.

EXAMPLE

words.inp:

```
4
RuN
RaBbit
HoBbit
StoP
```

codes.out:

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

text.inp:

```
StXRuYNvRuHoaBbvizXztNwRRuuNNP
```

(*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 on the output.)

EVALUATION

Your program will be allowed to run 10 seconds.
No partial credit can be obtained for a test case.