2716 Data Compression

Data compression is the process of transforming a body of data consisting of a character stream to a smaller size using certain codes, from which the original data can be restored by decompression. There exist many complex data compression algorithms that can sometimes achieve phenomenal compression. A simple dictionary-based compression technique is stated below. The technique is based on the concept of a dictionary that stores repeating code sequences as words. You are required to write a program to compress a given body of data using the technique and decompress the data when required. Assume that the complete set of distinct characters present in the body of data (including the blank character if present) is also given as a string of characters.

The dictionary is organized as an indexed list of words. A word in the dictionary is associated with its index (a distinct nonnegative integer), its coded representation (an ordered pair of integers) and its length (the number of characters in the word). Initially each character in the given string is stored in the dictionary as a word in the order in which it appears in the string. The index of the first entry in the dictionary is assumed to be 0. If the index of an entry in the dictionary is n then the index of the next entry is $(n + 1)$. The entire body of data is encoded replacing each character by its index in the dictionary.

The encoded data is compressed successively creating a new word in the dictionary by concatenating two existing words and encoding each appearance of the new word in the body of encoded data by its index until no further compression is possible. Since a new word is created by concatenating two existing words, each word in the dictionary is encoded as an ordered pair of integers (i, j) where i and j are respectively the indexes of the two existing words, which appear in the left and in the right, to form the word by concatenation. Each distinct character stored initially in the dictionary, say with index k, is encoded as $(-1, k)$.

The process of successive compression is stated below:

1. *Scan the encoded data from left to right and select the most frequently occurring word W. If there are two or more such words then the tie is broken first by selecting words having the largest length and then by selecting the word having the smallest index.*

2. *Let w be a new word that can be formed by concatenating an existing word, say E, to the left or to the right of W, i.e., the word w is either EW or WE and it may also be the word WW. Find the number of times each w occurs while scanning the encoded data from left to right.*

3. *Select the most frequently occurring new word w with frequency two or more. If there are two or more such words then the tie is broken first by selecting words w having the largest length, then selecting E with the smallest index and finally if necessary selecting the prefix EW instead of the postfix WE.*

4. *Add the selected new word w in the dictionary. Let its index be n. Scan the body of encoded data from left to right and replace each occurrence of w by its index n.*

5. *In case no w exists, consider the next possible W and repeat the process until no new word can be created.*

At the termination of the process the body of encoded data represents the given body of data in compressed form and the dictionary contains all words used in the compressed data. The compressed data may be decompressed using the dictionary and the given string of characters.

Input

The input may contain multiple test cases.

For each test case, the first line contains the test case number p and the option number q, which is either 1 (for compression) or 2 (for decompression). The second line gives a string of complete set of distinct characters that appear in the body of data. The string terminates with the character '$'.

For data compression the body of data follows the second line in one or more lines and is terminated by the character '$'.

For decompression the dictionary and the body of encoded data follow the second line. A blank line separates the dictionary and the body of encoded data. Each word in the dictionary is given in a single line in the order it appears in the dictionary. The line contains the ordered pair of integers representing the word in encoded form. The body of encoded data appears in one or more lines as a sequence of integers. The character '$' terminates the body of encoded data.

The entire input set terminates with an input '0' for each of p and q. The input is illustrated in sample input.

Output

For each test case the first output line contains the test case number p and the option number q. If the option is data compression then the output is in the form of input for decompression. For decompression the body of decompressed data terminated by the character '$', follows the first output line.

The output is illustrated in sample output.

Sample Input

```
1 1
abc$
abcbaacbabbcaccabca$
2 2
01 $
-1 0
-1 1
-1 2
0 1
3 2
0 2

4 1 0 5 1 3 1 2 5 2 4 0 $
3 1
01 $
01 100 1011 0  01 0$
0 0
```

Sample Output

```
1 1
abc$
-1 0
-1 1
-1 2
0 1
3 2
```

```
0 2

4 1 0 5 1 3 1 2 5 2 4 0 $
2 2
01 100 1011 0  01 0$
3 1
01 $
-1 0
-1 1
-1 2
0 1
3 2
0 2

4 1 0 5 1 3 1 2 5 2 4 0 $
```