# BWT Conversion

(Time Limit: 3 seconds)

**Problem Description**

The Burrows Wheeler transform (BWT) compression by Run-Length Encoding (RLE) is a widely-used string index for pattern matching. We explain the RLE-BWT below by using input text *mississippi* as an example. Let *t* be the size of the input text (e.g., *t* = 11). We obtain a matrix *M* with dimensions *t*\**t*. The first line of *M* corresponds to the input text. The *i*-th line, for *i* = 2,...,*t*, corresponds to a circular left shift of *i* - 1 positions with respect to the input text. For mississippi, we obtain the following circular rotations matrix *M*:  
mississippi  
ississippim  
ssissippimi  
sissippimis  
issippimiss  
ssippimissi  
sippimissis  
ippimississ  
ppimississi  
pimississip  
imississipp

Subsequently, we sort these 11 rotations lexicographically to obtain a new matrix *P*, which is like:  
imississipp  
ippimississ  
issippimiss  
ississippim  
mississippi  
…  
ssippimissi

The BWT text is then given by the last column of matrix *P* (e.g., *pssmipissii*). And the row of *P* which contains original text *mississippi* is 5. In order to further reduce memory, the BWT text is further compressed using RLE, which represent each consecutive letters with a single tuple (frequency, letter). For instance, given the BWT text *pssmipissii*, the RLE-BWT text is *1p2s1m1i1p1i2s2i*. This RLE-BWT can then be used for subsequent pattern matching.

In practice, in order to perform forward and backward pattern matching (e.g., “*mis*” and “*sim*” are forward and backward matching in *missisippi*, respectively), we have to construct a reverse index (representing *ippississim*) for backward matching from the forward index (representing *missisippi*). You are asked to construct a reverse RLE-BWT index from a forward one. Hints: given a forward RLE-BWT (e.g., *1p2s1m1i1p1i2s2i*) and the number of rows in *P* which contains the original string (e.g., 5). You may first decode the original string (e.g., *mississippi*) from the forward RLE-BWT, reverse the decoded string (e.g., *ippississim*), and finally compute the reverse RLE-BWT (i.e., *1s1m1s1p1i1p1i2s2i*) using above encoding rule.

**Technical Specification**

* + Each BWT character is from a to z or from A to Z.
  + The length of BWT text is from 1 to 500.
  + The frequency in the RLE-BWT is from 1 to 500.
  + The number of row which indicating the original string is from 1 to 500.

**Input Format**

The input file contains one or more instances of the problem, occurring consecutively in the input file, without any blank lines separating them. Each instance has two lines. The first line contains the RLE-BWT text. Each entry is a tuple of (count, letter), NOT separated by any blank spaces from its neighbor tuples. The second line contains the integer number corresponding to the row of *P* which contains the input text. The last instance of the input file consists of the two lines

END  
0

**Output Format**

For each instance of the problem, your program should print the reverse RLE-BWT in a single line.

**Example**

|  |  |
| --- | --- |
| **Sample Input:** | **Sample Output:** |
| 1p2s1m1i1p1i2s2i 5 2G1C1A1T3A 1  END 0 | 1s1m1s1p1i1p1i2s2i  1T2G4A1C |