4114 RC Codes

After seeing the movie Die Hard 4, Rakesh, a 14 year old school student, became very interested in Cyber Security. He started reading up a lot on the subject. He came up with a coding scheme for characters that he called “RC”, in which each character is stored as a variable length code. (A variable length code is where the length of the code is not necessarily the same even for plaintexts of the same length.)

Rakesh wanted to represent RC codes as binary trees (called RTree), and he introduced a rule for the creation of RC codes called the prefix property. In an RC code no character code is a prefix of any other character code. For example, character strings over the alphabet {m,n,o,p} could be stored internally using the RC Code {m= 00, n = 010, o = 011, p = 1}.

Because of the prefix property, an RC can be represented as a binary tree (RTree) whose leaves are the alphabet characters and whose left and right branches are (implicitly) labeled 0 and 1, respectively. The leaves of the tree are arranged so that the code of a character is read as the string of branch labels leading from the root to the leaf.

Write a program that inputs an RTree (representing an RC code) and a bit string consisting of contiguously stored character codes, and that outputs the bit string decoded into its corresponding string of characters.

Input

The first line of input is an integer number N $(0 < N < 20)$ indicating the number of test cases.

The next line contains an RTree. If a tree is a single leaf, it appears as a single character; otherwise, if it has sub trees, it appears as a left parenthesis, followed by the left RTree sub tree, followed by a comma, followed by the right sub RTree sub tree, followed by a right parenthesis. You cas assume every non-leaf tree (tree that is not a leaf) has two non-empty sub trees. Also there will be at least two leaves. For example, the RTree representig the RC code in the second paragraph would appear as ‘`((m,(n,o)),p)`’. The next line again an integer number M $(0 < M < 20)$ that indicates the number of lines of binary codes to decode (whose lenght is at most 1000 characters). The subsequent lines each contain a nonempty binary string to be decoded. With that, the input for a test case ends.

If there are more than one test case the data for the next test case continues just below the end of the first test case. You may assume that all inputs to your program will be valid cryptograms that can be successfully decoded. We will use only the 26 English lower case characters.

Output

For each of the binary strings to be decoded, print the string of characters that the binary string represents, one per line, without intervening or terminating blank lines.

Sample Input

```
2
((a,(b,c)),(d,e))
2
111001101000
000100000100110100111001110111011 0011
((a,(b,c)),d)
```

```
2
0101011001101001100
0001101100110101010000011
```

Sample Output

```
edcba
ababcbcdcdedeae
bdcaddbca
accaddbdbac
```