Possible Passwords

After learning recursion, you've decided to look for some applications of the new problem solving technique you've learned. It turns out that there's an escalating rivalry between a couple student clubs on campus, "Unkempt Freshmen" and "Frustrated Student Underlings". The SGA at UCF suspects that both clubs are up to some fairly nefarious activities. In order to check up on the clubs, your boss has asked you to write a program that can guess passwords for the email accounts of each club. Luckily, after gathering some data, you know exactly how long each password is and what the possible letters are for each slot. As an example, it's possible you might have narrowed down a particular password to be three letters long where the first letter is from the set {'a', 'b', 'c'}, the second letter is from the set {'x', 'y'} and the third letter is from the set {'d', 'm', 'n', 'r'}. From this data, there are 24 possible passwords. You will have to write a program that can iterate through each possible password, in alphabetical order. Since printing out each of the passwords might create unnecessarily long output, to check to see that your program works, you'll only be asked to output specific alphabetically ranked possible passwords from the list, instead of the whole list itself.

The Problem

Given the length of a password, a list of possible letters for each letter in the password, and a desired alphabetical rank, determine the possible password of the given rank.

The Input

The first line of the input file will contain a single positive integer, c $(c \leq 100)$, representing the number of input cases. The input cases follow, one per line. The first line of each input case will contain a single positive integer, $m (m \leq 20)$, the length of the password. The following m lines will contain strings of distinct lowercase letters in alphabetical order representing each of the possible letters for each letter in the password. The i^{th} line in this set will store the possible letters for the i^{th} letter in the password, in alphabetical order. The last line of each test case will contain a single positive integer, r $(r \leq 1048576)$, representing the rank of the possible password to output. (You are also guaranteed that the product of the lengths of these m lines won't exceed 1,000,000,000.)

The Output

For each case, output the correct possible password for the query, in all lowercase letters. It is guaranteed that all queries will be for a valid ranked password.

Sample Input

```
2
3
abc
xy
dmnr
10
2
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
676
```

Sample Output

```
bxm
zz
```