C. String Manipulation 1.0

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One popular website developed an unusual username editing procedure. One can change the username only by deleting some characters from it: to change the current name s, a user can pick number p and character c and delete the p-th occurrence of character c from the name. After the user changed his name, he can't undo the change.

For example, one can change name "`arca`" by removing the second occurrence of character "`a`" to get "`arc`".

Polycarpus learned that some user initially registered under nickname t, where t is a concatenation of k copies of string s. Also, Polycarpus knows the sequence of this user's name changes. Help Polycarpus figure out the user's final name.

Input

The first line contains an integer k $(1 \leq k \leq 2000)$. The second line contains a non-empty string s, consisting of lowercase Latin letters, at most 100 characters long. The third line contains an integer n $(0 \leq n \leq 20000)$ — the number of username changes. Each of the next n lines contains the actual changes, one per line. The changes are written as "p_i c_i" (without the quotes), where p_i $(1 \leq p_i \leq 200000)$ is the number of occurrences of letter c_i, c_i is a lowercase Latin letter. It is guaranteed that the operations are correct, that is, the letter to be deleted always exists, and after all operations not all letters are deleted from the name. The letters' occurrences are numbered starting from 1.

Output

Print a single string — the user's final name after all changes are applied to it.

Examples

input

```
2
bac
3
2 a
1 b
2 c
```

output

```
acb
```

input

```
1
abacaba
4
1 a
1 a
1 c
2 b
```

output

```
baa
```

Note

Let's consider the first sample. Initially we have name "`bacbac`"; the first operation transforms it into "`bacbc`", the second one — to "`acbc`", and finally, the third one transforms it into "`acb`".