Passphrase Obfuscation (passphrase)

To better protect the server containing the future tasks of this competition, Luca is choosing a long *passphrase* to encrypt the tasks archive. Due to the current pandemic restrictions, Luca is unable to meet the rest of the team and directly communicate to them a newly chosen passphrase: he is left with no other option than sending it using a potentially insecure channel, and for this reason he must make it unrecognizable to external malicious eyes.

After a careful evaluation of potential attackers, Luca devised an infallible plan to safely communicate the passphrase: he is going to remove exactly K characters from the sequence of passphrases used by the team so far, joined together. These combined passphrases form a string of length N that everybody in the team knows; he can then only communicate the removed characters, making the new passphrase unintelligible to others while still being reconstructible by his colleagues.

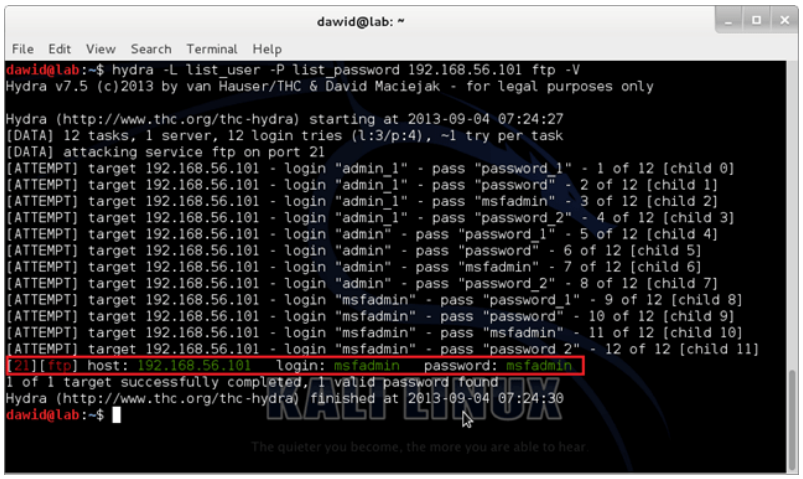

Figure 1: An example of a dictionary attack that Luca wants to prevent.

Additionally, a security expert consulted by Luca suggested that the new passphrase should be as close as possible to the end of a dictionary to improve its resistance to the so-called *dictionary attacks*. More formally, this means that he should prefer a passphrase p_1 over another passphrase p_2 whenever, in the lexicographic order, p_1 **comes after** p_2. What is the best (i.e., later in the dictionary) passphrase that Luca can produce by removing K characters to the given string of passphrases?

☞ Among the attachments of this task you may find a template file `passphrase.*` with a sample incomplete implementation.

Input

The first line contains two integers N and K. The second line contains the given string of passphrases.

Output

You need to write a single line containing the best passphrase that Luca can choose.

Constraints

- $2 \leq N \leq 30\,000$.
- $1 \leq K < N$.
- The given string consists of N lowercase letters of the English alphabet.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) $K = 1$.

– **Subtask 3** (30 points) $N \leq 50$.

– **Subtask 4** (35 points) $N \leq 2000$.

– **Subtask 5** (25 points) No additional limitations.

Examples

input	output
11 4 coronavirus	rovirus
16 2 programmingisfun	rorammingisfun

Explanation

In the **first sample case** we obtain the most secure passphrase by deleting characters `c`, `o` (at second position), `n`, `a`. No other passphrase, after removing 4 characters, comes lexicographically after this one.

In the **second sample case** we obtain the most secure passphrase by deleting characters `p` and `g`. No other passphrase, after removing 2 characters, comes lexicographically after this one.