Problem C. Clean Up!

Time limit:	2 seconds
Memory limit:	512 megabytes

Once Charlie decided to start a new life by deleting all files in his Downloads directory. It's easy to do that using `bash` shell! It has two useful features: the "`rm`" command, which removes all files given as arguments, and patterns, which are replaced with the list of files matching them before executing the command.

Charlie ran "`rm *`", but received an "`Argument list too long`" response. Unfortunately, after `bash` replaced "`*`" with the names of all files in the Downloads directory, it failed to run the command because it had too many arguments.

After some experiments, Charlie realized he can execute "`rm abc*`" to delete all files with names starting with "`abc`" if there are at most k such files. If more than k files match this pattern, none of them will be deleted. Of course, he can replace "`abc`" with any string.

Help Charlie to find the smallest number of "`rm`" commands needed to delete all files. Assume that he can only use the "`rm`" command as "`rm <prefix>*`", where `<prefix>` consists of lowercase English letters (and can be empty).

Input

The first line contains two integers n and k — the number of files to delete, and the maximum number of files that can be deleted by one "`rm`" command ($1 \le n, k \le 3 \cdot 10^5$).

Each of the next n lines contains a single string, denoting a file name. All file names are distinct, non-empty, and consist of lowercase English letters. The total length of all file names doesn't exceed $3 \cdot 10^5$.

Output

Print a single integer — the smallest number of "`rm`" commands needed to delete all files.

Examples

standard input	standard output
4 2 a abc abd b	2
4 2 d c ab a	2
5 3 please remove all these files	3

Note

In the first example test, Charlie can execute "`rm ab*`" to delete files "`abc`" and "`abd`", and then execute "`rm *`" to delete files "`a`" and "`b`". Note that he can't just run "`rm *`" immediately, because initially all four files match an empty prefix.