# Problem J. The Paladin

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 1024 mebibytes |

A Paladin, a warrior adept at holy magic, needs your help forming spells. Being a Paladin, every spell must also be a Palindrome, meaning that it is the same string when read forwards and backward. The cost of constructing a new spell is based on rune pair costs (costs of adjacent letters) that occur in the final spell. Rune pairs not presented in the input are not allowed in the final spell.

The total cost of a spell is the sum of all rune pair costs in the spell for each time the pair occurs in the spell. For example, if the spell is `abacaba`, then the cost is the sum of the costs of `ab` + `ba` + `ac` + `ca` + `ab` + `ba`.

Determine the smallest cost to make a new Paladin spell of exactly a given length.

## Input

The first line of input contains two integers $n$ ($1 \le n \le 676$) and $k$ ($2 \le k \le 100$), where $n$ is the number of rune pairs and $k$ is the desired spell length in runes (i.e., letters).

Each of the next $n$ lines contains a string $s$ ($s$ consists of exactly two lower-case letters, which may be the same or different) and an integer $c$ ($1 \le c \le 100$), where the string $s$ is a rune pair, and $c$ is the cost of that rune pair. All rune pairs are distinct.

## Output

Output a single integer, which is the smallest possible cost for constructing a spell of length $k$ from the given runes, or $-1$ if it isn't possible.

## Example

| standard input | standard output |
|---|---|
| 5 9<br>ab 4<br>ba 1<br>bd 3<br>db 100<br>bc 4 | 20 |