

## Task 3 - Spell caster

A very important rediscovery was made during this competition. If you were to rearrange the letters in a spell, it becomes several times more powerful! But not just some random rearrangements, no, it must be a very precise and well crafted piece of magic, using an ancient formula by a legendary spell caster. For that very reason, you must do what nobody has done in millennia - craft a rearranged spell! That is the third and final challenge in this year's Triwizard championship! To achieve this wizardry, you need to follow a two step algorithm.

### Step 1 - Extracting the characters

You are given a sentence that consists only of words of **Latin characters**. (no punctuation, no special symbols or numbers). First, you need to turn that sentence into a sequence of characters in the following way: until all words disappear, the last letter of each word (if exists) is removed from the word and is appended to the output sequence of letters.

For example, let's say you receive the sentence **"Fun exam right"**. You first split into 3 words: **["Fun", "exam", "right"]**. Then the last letters are extracted: **["Fun", "exam", "right"]** results into **"nmt"**. And you continue as such:

- **["Fu", "exa", "righ"] => "nmtuah"**
- **["E", "ex", "rig"] => "nmtuahExg"**
- **["", "e", "ri"] => "nmtuahFxgei"**
- **["", "", "r"] => "nmtuahFxgeir"**

After the extraction is finished, you are left with the sequence **"nmtuahFxgeir"**. It is now time to use that into the next step.

**NOTE:** *Completing just step 1 of this task does not reward any points in BGCoder!*

## Step 2 - Moving the characters

The next step in the ancient spell is to move each letter (from positions 0, 1, ..., n-1) to the right **N** times. The number **N** is the position of the letter in the Latin alphabet regardless of its casing ('a' => 1, 'b' => 2, ..., 'z' => 26). When a letter is moved to the right, if it is the last letter of the sequence, its next position is the first position in the sequence, just before all the letters.

The moving of letters starts from the sequence you reached in **step 1**. In the current example, that is **"nmtuahFxgeir"**. First, the letter **'n'** at position **0** is moved right **14** times:

**"nmtuahFxgeir" => "mtnuahFxgeir"**. Then the letter **'t'** at position **1** is moved right **20** times: **"mtnuahFxgeir" => "mnuahFxgetir"**. Then the process continues:

**"mnuahFxgetir" => "mnahFxgetiru" => "mnaFxgetiru" => "mnaFxgetiruh" => "gmnaFxetiruh" => "gmnaFxtiruhe" => "gmnaiFxtiruhe" => "gmrnaiFxtuhe" => "gmrnaiuFxthe" => "gmrnaihuFxte" => "gmrneaihuFxt"**

Finally, you are left with **"gmrneaihuFxt"**, which should be printed on the console.

## Input

- The input data should be read from the console.
- On the first line, you will receive a sequence of words, separated by a space.

## Output

- The output data should be printed on the console.
- The output data consists of a single text line, holding the obtained result.

## Constraints

- The input data will always be valid and in the format described.
- Each word will consist of only Latin letters. (no symbols or numbers)

## Examples

Input	Output	Input	Output
Telerik Academy	AymlTiedkaerec	Hi exam	maiHex