## Problem 1 – Durankulak Numbers

In Durankulak we have a special way to write numbers. We use the following 168 digits:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | … | Z |
| 0 | 1 | 2 | … | 25 |
| … | | | | |
| аA | аB | аC | … | аZ |
| 26 | 27 | 28 | … | 51 |
| … | | | | |
| bA | bB | bC | … | bZ |
| 52 | 53 | 54 | … | 77 |
| … | | | | |
| fA | … | fJ | fK | fL |
| 156 | … | 165 | 166 | 167 |

We write the numbers as sequences of digits. The last digit of the number (the most right one) has a value as shown in the above table. The next digit on the left has a value 168 times bigger than the shown in the above table, the next digit on the left has 168\*168 times bigger value than the shown in the table and so on. Your task is to write a program to **convert a Durankulak-style number into its corresponding decimal representation**.

### Input

The input data consists of a single string – a Durankulak-style number.

The input data will always be valid and in the described format. There is no need to check it explicitly.

### Output

The output data consists of a single line holding the calculated decimal representation of the given Durankulak-style number and should be printed at the console.

### Constraints

* The input string will have between 1 and 16 characters.
* Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| U | 20 |  | bM | 64 |  | BaG | 200 | CfI | 500 |