## **Expression Evaluator**

Write a program that evaluates a string expression. You will be given that **string expression** on the first line in the form of **numbers** and **operators** separated with a **single space** from each other. Your job is to take that string expression and calculate the result after evaluating it.

To do that, you have to follow a simple rule. If, for example, we have this string **"2 4 \* 1 3 -"**, the first time we meet an operator (**\***), we should take **all the numbers** we have so far (**2, 4**), apply that operation to them, and **save** the result (**8**). The next time we meet an operator (**-**), we again take **all the numbers** we have (**8, 1, 3**) and apply the operator to them **in that order** (**8 - 1 - 3 = 4**). In the **end**, we **print** the result.

All the numbers will always be **integers,** and the possible operators are **"\*", "+", "-", "/"**. It is important to **keep the order** of the numbers (especially for "**/"** and "**-"** because the **order matters**). When having a **division**, you should **round** the result to the **lower integer**.

### Input

* **Single line**: a string containing integers and operators

### Output

* **Single number**: the result after the evaluation

### Constrains

* When reaching an operator, it is sure that you will have a **minimum of one number** to evaluate
* The string will always **end** with an **operator**, so you get one number as a result
* Operators and numbers will **always** be **valid**
* There will be **no** case of **division by zero**
* There might be **negative numbers** in the string

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 6 3 - 2 1 \* 5 / | 1 | 6 - 3 = 3  3 \* 2 \* 1 = 6  6 / 5 = 1 |
| 2 2 - 1 \* | 0 | 2 - 2 = 0  0 \* 1 = 0 |
| 10 23 \* 4 2 / 30 10 + 100 50 - 2 -1 \* | 164 | 10 \* 23 = 230  230 / 4 / 2 = 28  28 + 30 + 10 = 68  68 - 100 - 50 = -82  -82 \* 2 \* -1 = 164 |