Appleby Contest '19 P3 - A Recursion Problem

Time limit: 2.0s **Memory limit:** 128M

There are many ways to represent arithmetic expressions.

We commonly use infix notation where operations are put in between values (i.e. $1 + 2 \times 3 = 7$), but another less well-known method is prefix notation. This is where operations are put before values. For example, if we want to add two numbers we would write `+ x y` instead of `x + y`. Furthermore, brackets are used to enforce order of evaluation.

The formal definition of prefix notation we will be using is as any one of the following options:

- `x`, where `x` is an integer.
- `(+ x y)`, where `x` and `y` are valid prefix notation expressions. The result of this expression is $x + y$.

Your objective today is to evaluate prefix notation expressions **that only involve addition**.

Input Specification

The first and only line of input contains a valid prefix notation expression. You can expect the expression to only consist of the following characters: `0123456789()+-` (and the space: ` `)

Output Specification

The value of that expression.

Constraints

Any integer x in the given expression will satisfy the following inequality: $-10^4 \leq x \leq 10^4$.

$1 \leq |s| \leq 10^5$, where $|s|$ denotes the length of the prefix notation expression.

Sample Input

```
(+ 1 (+ (+ (+ 3 4) -2) 5))
```

Sample Output

```
11
```

Sample Explanation

Here is the sample input being simplified:

- `(+ 1 (+ (+ (+ 3 4) -2) 5))`
- `(+ 1 (+ (+ 7 -2) 5))`
- `(+ 1 (+ 5 5))`
- `(+ 1 10)`
- `(11)`
- `11`