小高的樹形轉換術 (Convert)

問題敘述

在某堂課上，陽光從教室窗戶斜斜地灑進來，粉筆在黑板上劃過，留下一串白色的運算式。

小高半趴在桌上，眼神在老師畫的二元樹和算式之間來回游移。

「前序、中序、後序……這些東西到底有什麼不一樣啊？」他心裡嘀咕著。

就在老師把一棵樹分別寫成三種不同順序的瞬間，小高腦袋突然「叮！」地響了一下——

「欸欸！原來一棵樹的節點順序可以有前序、中序、後序三種欸！」

他立刻在筆記本上寫下各種順序的節點序列，想看看它們是怎麼互相轉換的。

同桌的小睿探頭一看，皺著眉說：「你這是在研究樹，還是在發明密碼啊？」

老師走過來瞄了一眼，扶額嘆氣：「小高，與其手動一個一個對照，不如寫個程式幫你轉換比較快吧。」

小高眼睛一亮：「對喔！我就來寫一個，可以把中序表示法轉成前序表示法的程式，這樣電腦就能輕鬆計算出結果了！」

現在，你的任務就是幫小高完成這個程式：

1. 輸入：一個運算式的中序表示法，長度不超過 10000，字串中可能會包含括號字元。

2. 輸出：該運算式的計算結果與前序表示法。

3. 運算規則：遵守數學的優先順序（先乘除，後加減）。

4. 保證運算過程不會超過 integer 的範圍。

5. 輸入字串的數字不一定只有一位數哦！

小高和同學們常常被中序式的運算順序搞得暈頭轉向，請幫他們轉換成前序式，解開這個難題吧！

輸入說明

運算式是一個由運算元和運算子組成的序列。

運算元：運算式中被操作的對象，例如數字。

運算子：執行運算操作的符號，例如加號 (+)、減號 (-)、乘號 (*)、除號 (/)。

輸入為一個帶括號的中序表示法字串，運算元、運算子與括號之間均用括號隔開。

題目保證：

1. 運算子僅包含加（+）、減（–）、乘（*）、除（÷）四種。

2. 除法運算（/）皆採向零取整。在 C 語言中，運算子 / 預設即為向零取整，且不會出現除以 0 的情況。

3. 運算元均為非負整數，且數值不超過 int 範圍。

補充說明：

1. 在計算過程與最終結果中，可能會出現負數。

2. 括號的巢狀層數至多 100 層，且僅出現 (跟)。

測資限制

- $1 \leq$ 字串長度 ≤ 10000。

輸出說明

輸出兩行：第一行為運算結果，第二行是輸入字串的前序表示法。

範例測資

範例輸入 1

```
1 + 2 - 3 * 4
```

範例輸出 1

```
-9
- + 1 2 * 3 4
```

範例說明 1

錯誤範例：$+\ 1\ -\ 2\ *\ 3\ 4$

這個的計算結果雖然也是 -9，但它不能作為正確答案。因為依照加減乘除的運算規則，當運算子優先級相同時，計算必須從左至右進行。因此，只有題目所示的輸出才是正確的！

範例輸入 2

```
( 1 + 2 ) * 3
```

範例輸出 2

```
9
* + 1 2 3
```

範例輸入 3

```
12 + 34
```

範例輸出 3

```
46
+ 12 34
```

恭喜你看完這道題目！

為表達祝賀，最後附上輸入模板供大家使用 $>\ <$

連結：https://reurl.cc/DOjQeE

針對 TODO: 的地方修改即可！

Convert

Problem Description

In a class, sunlight slanted through the classroom window, and the chalk traced across the blackboard, leaving a string of white mathematical expressions.

Xiao Gao slouched halfway over his desk, his gaze shifting between the binary tree the teacher had drawn and the equation on the board.

”Preorder, inorder, postorder... What’s the difference between all these?” he muttered to himself.

Just as the teacher rewrote the same tree in three different orders, Xiao Gao’s mind suddenly lit up—

”Hey! So a tree’s nodes can actually be arranged in preorder, inorder, and postorder!”

He quickly scribbled down each ordering of the nodes in his notebook, curious to see how they could be converted into one another.

His desk mate, Xiao Rui, leaned over, frowned, and asked, ”Are you studying trees⋯ or inventing some kind of secret code?”

The teacher walked by, glanced at his notebook, and sighed while rubbing his forehead. ”Xiao Gao, instead of matching them by hand one by one, why not just write a program to

do the conversion for you?"

Xiao Gao' s eyes lit up. "Right! I' ll write something to convert from an inorder expression to a preorder one—then the computer can handle the calculation easily!"

Now, your task is to help Xiao Gao finish this program:

1. Input: An infix (inorder) expression, up to length 10000, which may contain parentheses.

2. Output: The evaluated result of the expression and its preorder representation.

3. Evaluation rules: Follow standard operator precedence (multiplication/division before addition/subtraction).

4. It is guaranteed that the computation will not exceed the range of an integer.

5. Numbers in the input may have more than one digit.

Xiao Gao and his classmates often get confused by the order of operations in infix expressions. Please help them convert the expressions into prefix form to solve this problem!

Input Format

An expression is a sequence composed of operands and operators.

Operand: The object being acted upon in an expression, such as a number.

Operator: A symbol that performs an operation, such as the plus sign (+), minus sign (−), multiplication sign (*), or division sign (/).

The input is a parenthesized infix expression string, where operands, operators, and parentheses are all separated by spaces.

The problem guarantees the following:

1. Operators include only addition (+), subtraction (−), multiplication (*), and division (/).

2. Division (/) is performed as truncation toward zero. In C, the / operator by default truncates toward zero, and division by zero will not occur.

3. Operands are non-negative integers and will not exceed the int range.

Additional notes:

1. During computation and in the final result, negative numbers may appear.

2. Parentheses can be nested up to 100 levels, and only (and) may appear.

Constraints

- $1 \leq$ string length ≤ 10000

Output Format

Output two lines:

The first line contains the evaluation result.

The second line contains the prefix (preorder) representation of the input expression.

Example Test Case

Sample Input 1

```
1 + 2 - 3 * 4
```

Sample Output 1

```
-9
- + 1 2 * 3 4
```

Explanation of Sample 1

Incorrect conversion result: $+\ 1\ -\ 2\ *\ 3\ 4$

This expression also evaluates to -9, but it cannot be considered the correct answer. According to the rules of arithmetic, when operators have the same precedence, calculations must be performed from left to right. Therefore, only the output given in the problem is correct.

Sample Input 2

```
( 1 + 2 ) * 3
```

Sample Output 2

```
9
* + 1 2 3
```

Sample Input 3

```
12 + 34
```

Sample Output 3

```
46
+ 12 34
```

Congratulations on finishing this problem!

To celebrate, here’ s an input template for everyone to use > <

link：https://reurl.cc/DOjQeE

Just make changes to the parts marked with TODO.