**アカツキ・コーディング問題**

**dentaku**

**問題**

ある記法の数式が標準入力から渡されます。 この電卓は、数式に対して以下の操作を演算子がなくなるまで繰り返します。x, y, zを整数として、

* 任意の連続した「x y +」を「xとyの和」に置換します。
* 任意の連続した「x y -」を「xとyの差」に置換します。
* 任意の連続した「x y \*」を「xとyの積」に置換します。
* 任意の連続した「x ++」を「xに1足した整数」に置換します。
* 任意の連続した「x y z @」を「xy+yz+zxを計算した結果の整数」に置換します。

上記操作を繰り返した結果、正しい式であれば最終的に一つの数値が残り、それが数式の計算結果となります。 また、正しい式が与えられた場合、この操作の順序に関わらず計算結果が変わらないことが証明できます。

後述する解き方を参考に、与えられた数式を計算して結果を表示する(標準出力に出力する)プログラムを実装してください。

**入力と出力**

入力は以下の形式で標準入力から渡されます。

S

例えば 2 と 3 を加算する数式は 2 3 + で、これが標準入力から渡されます。

数式を計算した結果を出力してください。

前述の入力 2 3 + の計算結果は 5 となります。 この場合出力する結果は以下となります。

5

また、2 + 3, 1 2 + 3 など不正な数式が与えられる場合があります。 不正な数式が与えられた場合は invalid と出力してください。

* 数値、演算子はスペースで区切られています。
* 利用できる演算子は +, -, \*, ++, @ です。その他の演算子は入力されないことが保証されています。
* 数式に含まれる数値、計算過程はすべて整数であり、絶対値は 1 億以下です。
* 数値と演算子の個数は合計10万個以下です。
* 計算した結果、最終的に一つの数値が残るような数式に限り、正しい数式となります。

**解き方**

この記法の数式の計算はスタックというデータ構造を使うことで、簡単に処理できます。 スタックとはデータを後入れ先出しの構造で保持するデータ構造です。 Push という操作でデータを格納し Pop という操作でデータを一つづつ取り出します。 このとき後に格納されたデータが先に格納されていたデータより先に取り出されます。

参考: [Wikipedia スタック](https://ja.wikipedia.org/wiki/%E3%82%B9%E3%82%BF%E3%83%83%E3%82%AF)

このスタックというデータ構造を利用すると以下のように計算式を処理できます。 計算式の先頭から順に以下の処理を行うことで計算することができます。

* 数値であればスタックに Push する
* 演算子が来たらその演算子が必要とする項の数 Pop して値を取り出し演算子を適用し、計算結果を Push する。
* 計算式の終端に来たら Pop することで最終的な計算結果を得る。

**実装の詳細**

**CLI**

入力値が引数として渡されて、答えを標準出力に表示するCLIアプリケーションを実装してください。 詳細は末尾の「CLI アプリ作成用テンプレート」を参照ください。

**入出力例**

なお、期待する入出力は [test/basic\_testcases.json](https://app.tracks.run/editor/test/basic_testcases.json) ディレクトリにいくつか定義されています。 実装する際の参考にしてください。

**例 1**

入力

8 3 +

出力

11

**例 2**

入力

8 3 3 \* + 2 + 1 -

出力

18

**例 3**

入力

3 - 2

出力

invalid