AMERICAN COMPUTER SCIENCE LEAGUE

2018-2019 **Contest #4**

Intermediate Division - Prefix Evaluation

考号/Exam Code: _____ 姓名/Name:_____ 学校/School:_____

PROBLEM: Evaluate a prefix expression. The operands in the expression are single digit whole numbers. The operators are binary addition (+), subtraction (-), and multiplication (*); the trinary operator "switcher" (@); and the trinary operator "max" (>). The @ operator of *a*, *b*, and *c* returns *b* when *a* is positive; otherwise, it returns *c*. The > operator returns the largest of its 3 operands.

Each line of data is a valid prefix expression with operands and operators separated by at least one space.

Example 1: `* + 4 5 - 3 1` simplifies to `* 9 2`, which has a value of 18.

Example 2: `@ - 8 9 7 6` simplifies to `@ -1 7 6`, which has a value of 6.

Example 3: `+ > 8 * 2 7 9 6` simplifies to `+ > 8 14 9 6`, which simplifies to simplifies to `+ 14 6`, which has a value of 20.

INPUT: Five lines of data. Each line is a string, <= 128 characters, representing a valid prefix expression with operands and operators as described above. At least one space will separate operands and operators.

OUTPUT: Evaluate each prefix expression and print the answer.

SAMPLE INPUT:

```
* + 4 5 - 3 1
@ - 8 9 7 6
+ > 8 * 2 7 9 6
- @ - 3 5 7 * 2 4 > * 4 6 * 3 7 * 9 3
* 7 - + 4 6 > 0 - 2 3 1
```

SAMPLE OUTPUT:

#1. 18
#2. 6
#3. 20
#4. -19
#5 63

TEST INPUT

```
> * 4 2 + 8 5 - 9 2
@ - 7 9 * 3 - 4 7 * + 7 2 5
> @ 4 * - 3 7 - 8 9 6 - 7 9 + 3 2
* + 2 @ > - 4 7 - 6 8 - 5 9 + 7 * 8 2 - * 7 4 * 8 4 9
> 2 > - 4 * 3 2 * - 1 6 2 + - 7 9 - 4 1 @ @ 6 - 3 8 4 * 4 7 * 6 5
```

AMERICAN COMPUTER SCIENCE LEAGUE

2018-2019 **Contest #4**

Intermediate Division - Prefix Evaluation

考号/Exam Code: _____ 姓名/Name:_____ 学校/School:_____

PROBLEM（问题）: 计算前缀表达式。表达式中的操作数是单位整数。运算符是二进制加法（+）、减法（-）和乘法（*）；三元运算符“switcher”（@）和三元运算符“max”（>）。当 a 为正数时，a、b 和 c 的@运算符返回 b；否则，它返回 c。>运算符返回其 3 个操作数中最大的一个。

每行数据都是一个有效的前缀表达式，其中操作数和运算符至少由一个空格分隔。

Example 1: * + 4 5 - 3 1 简化为 * 9 2,其值为 18.

Example 2: @ - 8 9 7 6 简化为 @ -1 7 6, 其值为 6.

Example 3: + > 8 * 2 7 9 6 简化为 + > 8 14 9 6, 简化为+ 14 6, 其值为 20.

INPUT（输入）: 五行数据。每行是一个字符串，不超过 128 个字符，用上面描述的操作数和运算符表示有效的前缀表达式。至少有一个空格将分隔操作数和运算符。

OUTPUT（输出）: 计算每个前缀表达式并输出结果。

SAMPLE INPUT（示例输入）:

```
* + 4 5 - 3 1
@ - 8 9 7 6
+ > 8 * 2 7 9 6
- @ - 3 5 7 * 2 4 > * 4 6 * 3 7 * 9 3
* 7 - + 4 6 > 0 - 2 3 1
```

SAMPLE OUTPUT（示例输出）:

```
#1. 18
#2. 6
#3. 20
#4. -19
#5. 63
```

TEST INPUT

```
> * 4 2 + 8 5 - 9 2
@ - 7 9 * 3 - 4 7 * + 7 2 5
> @ 4 * - 3 7 - 8 9 6 - 7 9 + 3 2
* + 2 @ > - 4 7 - 6 8 - 5 9 + 7 * 8 2 - * 7 4 * 8 4 9
> 2 > - 4 * 3 2 * - 1 6 2 + - 7 9 - 4 1 @ @ 6 - 3 8 4 * 4 7 * 6 5
```