超能纸带机

【问题描述】

超能纸带机（Capability Advanced Machine, CAM）是由一个简单的**程序**、一条无限长的**纸带**和一个**读写头**构成的。

纸带由一长列单元格构成，每个单元格上可以写 0~8 的这几种符号之一，也可以为空。为了方便，使用符号 9 表示对应的单元格为空。

程序开始运行前，纸带上已经有一些符号，这个符号序列称为**输入**。如下面就是一个输入：

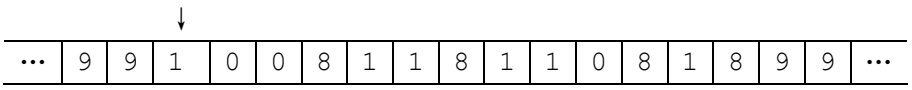
```
100811811081
```

输入必定是一串连续的非空符号，除了输入符号，纸带的其他地方都是空的。上面的输入在纸带上表示如下：

…	9	9	1	0	0	8	1	1	8	1	1	0	8	1	9	9	…

读写头始终指向纸带的一个单元格。它可以由程序控制左右移动，可以由程序控制向纸带上写入一个符号，还可以读出单元格上的符号作为程序运行的参数。

程序开始运行时，读写头指向纸带的第一个非空符号。如下面是程序开始运行时的读写头和纸带。



超能纸带机的程序非常简单，只有三种指令：

- L

```
L <符号 C>
```

该指令的作用是向读写头指向的纸带单元格上写入符号 C，并将读写头向左移动一个单元格。符号 C 可以是 **0~9 或?**之一，其中，C=9 表示向纸带写入一个空符号，而 C=?表示不改变纸带上的当前符号。

- R

```
R <符号 C>
```

该指令的作用是向读写头指向的纸带单元格上写入符号 C，并将读写头向右移动一个单元格。符号 C 可以是 **0~9 或?**之一，其中，C=9 表示向纸带写入一个空符号，而 C=?表示不改变纸带上的当前符号。

- LOOP-END

```
LOOP <符号表 H>
<循环体>
END <符号表 E>
```

该指令是一个循环指令，有两个符号表 H 和 E，符号表由 0~9 和?中的零个或多个符号构成（其中?为**通配符**，用以匹配任何符号）。指令中，循环体内可以是任何合法的零个或多个指令（循环可以无限层嵌套）。

指令的执行过程如下：

1. 首先判断当前读写头上的符号是否在符号表 H 中（注意?可以匹配任何符号，所以如果 H 中包含**?**，则条件一定成立），如果不在，则跳出循环，否则继续执行。注意：当 H 为空时，循环体将不会执行。

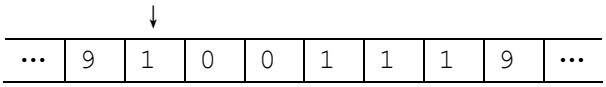
2. 执行循环体内的内容。

3. 当执行完循环体的内容后，判断当前读写头的符号是否在符号表 E 中（同样，E 中的**?**也可以匹配任何符号），如果不在，则跳出循环，如果在，则跳转到第 **1** 步。注意：当 E 为空时，则条件必然不满足，将跳出循环。

下面是一个 CAM 的程序（#和其后面的内容表示**注释**），其功能是：当读写头指向一个用二进制表示的整数的第一个符号时，该程序将此整数加 1。

```
# 将读写头移至该整数的最后一个单元格
LOOP 0 1
    R ?
END ?
# 上面的程序中，只要读写头处的符号是 0 或 1 就右移，所以现在指针指向的是
# 整数最后一个单元格的后一个单元格，还需要向左移动将读写头移回到整数最后一位。
L ?
# 将整数加 1，首先将末尾连续的 1 变成 0，然后将最后一个 0 变成 1
LOOP 1
    L 0
END ?
L 1
# 将读写头移回整数的起始位置
LOOP 0 1
    L ?
END ?
# 已经移出整数，需要向右移动一个单元格使读写头指向整数的第一个单元格
R ?
```

设我们有一个整数 39：



对于下面的输入，该程序的运行过程如下：

步骤	纸带和读写头状态	步骤	纸带和读写头状态
开始	… 9 **1** 0 0 1 1 1 9 …	8	… 9 1 0 0 1 1 0 9 …
1	… 9 1 **0** 0 1 1 1 9 …	9	… 9 1 0 0 1 0 0 9 …
2	… 9 1 0 **0** 1 1 1 9 …	10	… 9 1 0 0 0 0 0 9 …
3	… 9 1 0 0 **1** 1 1 9 …	11	… 9 1 0 1 0 0 0 9 …
4	… 9 1 0 0 1 **1** 1 9 …	12	… 9 1 0 1 0 0 0 9 …
5	… 9 1 0 0 1 1 **1** 9 …	13	… 9 1 0 1 0 0 0 9 …
6	… 9 1 0 0 1 1 1 **9** …	14	… 9 1 0 1 0 0 0 9 …
7	… 9 1 0 0 1 1 1 9 …	15	

对一组只包含整数的算术表达式求值，表达式运算包含**加法**、**减法**和**乘法**。乘法优先级最高，但可以通过使用**小括号**来改变优先级。对于每个表达式，其参与运算的都是**变量**或者**常数 1**。变量最终会用整数代替，在代入变量的值后表达式的任何一步不会出现负数或 0 的情况。

表达式中最多包含三个运算符（注意是三个而不是三种，所以 a+b+c 是包含两个运算符的表达式），且不会出现两个乘号。

写一个程序，对于每一个表达式产生一个 CAM 程序，该程序的输入是变量的值，在运行完该程序后，纸带上**只剩下**表达式的运算结果。为了简化问题，运算结果中可以保留多余的前导 0。使用下面的格式表示变量的值：

首先将第一个字母(a)的值表示成二进制，然后将第二个字母(b)的值表示成二进制，在两个字母前用一个符号 8 连接，接着是第三个字母，第四个字母。如果表达式中只有三个字母，则第四个字母不会输入，同样，如果只有 k 个字母，则第 k+1 个字母不会输入。例如，一个表达式中有 a,b,c 三个字母。如果 a=10, b=4, c=7，则输入为

```
1010810081111
```

如果 a=1, b=100, c=3，则输入为

```
1811001008111
```

如果表达式中没有任何字母，则输入为空，这时候，纸带上的每个单元格里都是空字符（9）。

字母的值不会是 0 或负数，变量的值都不会超过 1023，但运算的过程和结果有可能超过 1023。

【输入文件】

输入文件 camp.in 仅一行，为一个四则运算表达式。该表达式满足：

- 至多有三个运算符
- 至多有一个乘号
- 可能包含一个或多个括号，但括号不会直接被括号包含，如((a+b))*c 不会出现，正确的出现方式是(a+b)*c。注意，这不代表括号不会嵌套，如 a+(b+(c+d))是合法的。
- 表达式中运算对象仅包含字母和 1
- 如果第 i 个字母在表达式中出现，则第 i-1 个字母一定在表达式中出现。如：如果表达式中有 c，那么表达式中一定有 b 和 a。
- 如果按照先括号内再括号外，先乘除，后加减，同级运算从左到右的顺序运算，则在运算的任何一步，你都可以假设值为正数。所以类似(1-1)的表达式是不可能出现的。但是(a-b)这样的表达式可能会出现，因为 a,b 的值由输入给定，而可以设定输入使 a-b 的值为正。

【输出文件】

输出文件 camp.out 包含一个 CAM 的程序，该程序仅包含上面所述的几种指令。你的输出格式必须满足：

- 每行至多一条指令
- 所有的指令标识符(LOOP, END, L, R)都大写
- LOOP 和 END 后面的符号表每两个符号之间用至少一个空格分隔，符号表的第一个符号与指令标识符之间用至少一个空格分隔。符号表可以不按大小顺序。同一个符号可以重复，但重复的符号只算一次。如下面是一个正确的 LOOP END 指令

```
LOOP  6  1  3
…
END  7  1  ?  1
```

它和下面的指令是等效的

```
LOOP 1 3 6
…
END ?
```

- 可以在程序的任意位置放任何个数的空格或制表符，但这些分隔符不能将指令标识符分开。如：

```
LOOP        ?
         END
```

是可以的，但

```
LO        OP        ?
          END
```

是不行的，因为 LOOP 被分开了。

- 可以用#号标识一条行注释，但如果在#前有指令，#和指令之间至少用一个空格分隔
- 输出的程序不能超过 100000 行。

【样例输入/输出】

输入(camp.in)	输出(camp.out)
a+1	LOOP 0 1 R ? END ? L ? LOOP 1 L 0 END ? L 1

【样例说明】

上面的样例和问题描述中的例子一样，这里我们去掉了注释（你可以加上，不影响正确性）和最后将指针移动到整数开始的部分。这对结果没有影响。

【怎样测试你的程序】

在你的用户目录下，有一个程序 camp_r，你可以使用这个程序来运行你的输出。该程序是一个 CAM 的模拟程序，它能从 camp.out 中读入程序，从 tape.in 中读入输入串，然后将运行的结果输出到 run.log 中。

维了正确执行 camp_r，你需要建立一个 tape.in 文件，为初始的输入。tape.in 的第一个数为输入的长度 e，后面有用空格分隔的 e 个数，为输入。如下面是一个合法的 tape.in：

```
5
1 0 0 1 1
```

使用下面的命令运行你的程序：

```
./camp_r
```

在 run.log 中会出现运行的结果

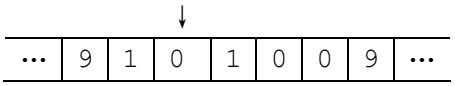
```
Total time: 25 Cycles
Tape:
 1H0 1 0 0
```

其中，第一行表示运行了多少步，第三行表示运行到最后时纸带和读写头的状态。

该状态的表述方法是：按顺序输出纸带上从第一个非空符号到最后一个非空符号之间的所有符号，如果一个符号前有 H，则表示当前读写头在此位置，如：

```
1H0 1 0 0
```

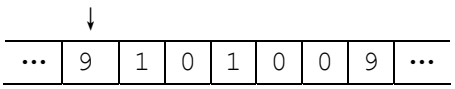
表示的状态：



如果读写头在第一个符号之前或最后一个符号之后，则对应的 9 也可能输出，如：

```
H9 1 0 1 0 0
```

表示



【如何调试你的程序】

为了方便调试程序，你可以给 camp_r 指定参数，参数可以是：

- -S<k>：输出从第 k 步开始的执行过程
- -E<k>：输出到第 k 步为止的执行过程，通过与-S 参数配合，可以输出执行的中间步骤
- -LALL：输出所有的执行过程。该参数只能单独使用。

如下面将显示在执行的第 10 步到执行的 12 步之间的过程：

```
./camp_r -S10 -E12
```

如果指令了输出过程，在 run.log 中将会有对应的过程，每一步一行。如使用该命令执行样例的输出，将会得到：

```
     10:  1 0 0H1 1
     11:  1 0 0 1H1
     12:  1 0 0 1H1
Total time: 25 Cycles
Tape:
 1H0 1 0 0
```

【评分标准】

本题一共有 10 组测试数据，每组数据占分数的 10%。对于你的每个输出，我们将设计一个或多个纸带的输入串，每个输入串有一定的分值，如果你的程序在给定的输入下运行 100000 步内（包括 100000 步）结束并将正确的解保留在纸带上，你就可以得到对应的分值。

【数据规模】

输入的数据范围如下：

输入编号	运算符个数	可能的运算符	是否可能有括号
1	1	+,-	否
2	1	+,-	否
3	1	+,-	否
4	≤2	+,-	否
5	≤3	+,-	是
6	≤3	+,-	是
7	1	+,-,*	否
8	≤2	+,-,*	否
9	≤3	+,-,*	是
10	≤3	+,-,*	是