1、多项式相加

我们经常遇到两多项式相加的情况，在这里，我们就需要用程序来模拟实现把两个多项式相加到一起。首先，我们会有两个多项式，每个多项式是独立的一行，每个多项式由系数、幂数这样的多个整数对来表示。

如多项式 $2x^{20}- x^{17}+ 5x^{9}- 7x^{7}+ 16x^{5}+ 10x^{4} + 22x^{2}- 15$

对应的表达式为：2 20 -1 17 5 9 - 7 7 16 5 10 4 22 2 -15 0。

为了标记每行多项式的结束，在表达式后面加上了一个幂数为负数的整数对。

同时输入表达式的幂数大小顺序是随机的。

我们需要做的就是把所给的两个多项式加起来。

时间限制：1000
内存限制：65536

输入
输入包括多行。 第一行整数 n,表示有多少组的多项式需要求和。(1 < n < 100) 下面为 2n 行整数，每一行都是一个多项式的表达式。表示 n 组需要相加的多项式。 每行长度小于 300。

输出
输出包括 n 行，每行为 1 组多项式相加的结果。 在每一行的输出结果中，多项式的每一项用“[x y]”形式的字符串表示，x 是该项的系数、y 是该项的幂数。要求按照每一项的幂从高到低排列，即先输出幂数高的项、再输出幂数低的项。 系数为零的项不要输出。

样例输入

```
2
-1 17 2 20 5 9 -7 7 10 4 22 2 -15 0 16 5 0 -1
2 19 7 7 3 17 4 4 15 10 -10 5 13 2 -7 0 8 -8
-1 17 2 23 22 2 6 8 -4 7 -18 0 1 5 21 4 0 -1
12 7 -7 5 3 17 23 4 15 10 -10 5 13 5 2 19 9 -7
```

样例输出

```
[ 2 20 ] [ 2 19 ] [ 2 17 ] [ 15 10 ] [ 5 9 ] [ 6 5 ] [ 14 4 ] [ 35 2 ] [ -22 0 ]
[ 2 23 ] [ 2 19 ] [ 2 17 ] [ 15 10 ] [ 6 8 ] [ 8 7 ] [ -3 5 ] [ 44 4 ] [ 22 2 ] [ -18 0 ]
```

提示
第一组样例数据的第二行末尾的 8 -8，因为幂次-8 为负数，所以这一行数据结束，8 -8 不要参与计算。

2、队列和栈

队列和栈是两种重要的数据结构，它们具有 push k 和 pop 操作。push k 是将数字 k 加入到队列或栈中，pop 则是从队列和栈取一个数出来。队列和栈的区别在于取数的位置是不同的。

队列是先进先出的：把队列看成横向的一个通道，则 push k 是将 k 放到队列的最右边，而 pop 则是从队列的最左边取出一个数。

栈是后进先出的：把栈也看成横向的一个通道，则 push k 是将 k 放到栈的最右边，而 pop 也是从栈的最右边取出一个数。

假设队列和栈当前从左至右都含有 1 和 2 两个数，则执行 push 5 和 pop 操作示例图如下：

push 5 pop

队列 1 2 -------> 1 2 5 ------> 2 5

push 5 pop

栈 1 2 -------> 1 2 5 ------> 1 2

现在，假设队列和栈都是空的。给定一系列 push k 和 pop 操作之后，输出队列和栈中存的数字。若队列或栈已经空了，仍然接收到 pop 操作，则输出 error。

时间限制：1000
内存限制：65536

输入
第一行为 m，表示有 m 组测试输入，m<100。 每组第一行为 n，表示下列有 n 行 push k 或 pop 操作。（n<150） 接下来 n 行，每行是 push k 或者 pop，其中 k 是一个整数。（输入保证同时在队列或栈中的数不会超过 100 个）

输出
对每组测试数据输出两行，正常情况下，第一行是队列中从左到右存的数字，第二行是栈中从左到右存的数字。若操作过程中队列或栈已空仍然收到 pop，则输出 error。输出应该共 2*m 行。

样例输入

```
2
4
push 1
push 3
pop
push 5
1
pop
```

样例输出

```
3 5
1 5
error
error
```

3、奇怪的括号

某天小 A 和同学在课堂上讨论到：“栈这种数据结构真是太优美了，既简单用途又广泛。”小 B 仰慕
小 A 许久，于是他拿出了自己在网上抄写的一道题问小 A，如何判断括号是否匹配呢

时间限制：1000
内存限制：65536

输入
多组数据，每组数据占一行，且都是由(、)、[、]、*、/这六种字符组成。

输出
每组数据输出一行，如果括号能匹配成功，输出 True，否则输出 False。括号匹配规则是：(和) 匹配 [和] 匹配 /* 和 */ 匹配 如果含有冗余字符也算匹配失败，例如 /***/ 是匹配失败的因为中间多了一个*。

样例输入

```
()/*[()]*/
*/**/
```

样例输出

```
True
False
```

4、中缀表达式的值

人们熟悉的四则运算表达式称为中缀表达式，例如(23+34*45/(5+6+7))。在程序设计语言中，可以利用堆栈的方法把中缀表达式转换成保值的后缀表达式（又称逆波兰表示法），并最终变为计算机可以直接执行的指令，得到表达式的值。 给定一个中缀表达式，编写程序，利用堆栈的方法，计算表达式的值。

时间限制：200
内存限制：65536

输入
第一行为测试数据的组数 N 接下来的 N 行，每行是一个中缀表达式。表达式中只含数字、四则运算符和圆括号，操作数都是正整数，数和运算符、括号之间没有空格。中缀表达式的字符串长度不超过 600。

输出
对每一组测试数据输出一行，为表达式的值

样例输入

```
3
3+5*8
(3+5)*8
(23+34*45/(5+6+7))
```

样例输出

```
43
64
108
```

提示

注意：运算过程均为整数运算（除法运算'/'即按照 C++定义的 int 除以 int 的结果，测试数据不会出现除数为 0 的情况），输出结果也为整数（可能为负）。 中间计算结果可能为负。