

程序功能说明：

本程序能计算一元多项式的加减运算，并将结果输出

程序运行所需环境和运行指令规范：

本程序需要在 java 虚拟机环境下运行

运行指令为：

```
java PolyCompute.class
```

```
java PolyCompute
```

程序的输入说明：

1. 输入仅有一行，由“0-9 + -, () { } ”这几种半角字符和“空格”组成，输入者通过键入回车键确认输入结束。正确的输入格式形如：

$-\{(3, 0), (2, 4), (1, 3), (4, 2), (3, 6)\} + \{(4, 4), (-3, 6), (1, 2)\} - \{(-3, 0), (-2, 1), (5, 2), (6, 3)\} - \{(-3, 0), (2, 1), (3, 8), (2, 6)\}$

2. 标准的输入由多组代表多项式的符号集合组成，形式如下“{多项式 1}+{多项式 2}-……+{多项式 n}”。

3. 每组花括号内为一个多项式字符串，所有花括弧前通过‘+’或‘-’代表多项式之间的加减关系，若第一组花括号前没有则默认为‘+’。

4. 花括号内的多项式由数对(c,n)组成，其中 c 为系数，n 代表次数。

5. 一个多项式的标准格式为： $\{(c_1, n_1), (c_2, n_2), \dots, (c_m, n_m)\}$ ，最外端由花括号与其它多项式区分开，内部是若干由括号包含的数对组成，括号之间由一个逗号‘,’分隔，两个数之间也由一个‘,’分隔。

6. 一个多项式内的数对中的 n 都不相同。

7. 组成全部多项式的字符串中的任何位置都可以存在空格。

8. 对于数对(c,n)，c 为系数，为十进制整数，取值范围为 $-999999 \leq c \leq 999999$ （即 c 前面可以有‘-’）；n 为该项的幂，为十进制整数，有 $0 \leq n \leq 999999$ 。所有数均可能出现前导 0，但 c 和 n 除去符号位之外的长度分别不超过 6 和 6。

9. 输入时，每个多项式限制为最多 50 个数对，多项式的个数限制为最多 20 个。

程序计算结果的输出规格：

正常运行结果输出为一个标准的多项式表达式字符串，多项式内的单项式次数为升序排列。

多项式中不存在系数为 0 的项，输出多项式中也不存在次数相等的两个项。

对于结果为 0 的输出，只输出一个字符 0。

对于输入的多项式系数的绝对值最大为 999999，然而对于输出的多项式的系数在合法输入下不会超过 int 所支持的最大整数，所以允许输出多项式的系数的绝对值超过 999999，不报错。

运行错误响应信息：

对于错误的输入，本程序会进行处理，并输出错误信息。

错误信息的第一行为 ERROR

第二行会出现以#开头的错误信息，下面为本程序能处理的所有错误：

"The input must be in the format: op{}op{}op{}...op{}!"

"The expn of each item in one input polynomial cannot be the same!"

"The number of polynomials does not match the number of operators!"

"The input polynomial must be in the format: {(n,n),(n,n),...,(n,n)}!"

"The decimal number in the input must be in the format : +/-xxx !(x means number)"

"The number of coefficients and indices in the polynomial does not match!"

"The exponent in the polynomial must be greater than or equal to 0!"








"The input can not be empty!"

"The absolute value of the number cannot exceed 999999!"

"The number of polynomials cannot exceed 20!"

"The number of pairs of a polynomial cannot exceed 50!"

程序类图：

- ▼  PolyCompute
 - polyList : ArrayList<Poly>
 - opList : ArrayList<Integer>
 - num : int
 -  PolyCompute()
 - parseTerm(String) : Poly
 - parsePoly(String) : int
 - compute() : Poly
 -  main(String[]) : void
- ▼  Poly
 - content : ArrayList<Term>
 -  Poly()
 - sort(boolean) : int
 - deleteZero() : void
 - add(Poly) : Poly
 - sub(Poly) : Poly
 - output() : void
- ▼  Error
 - str : String[]
 - outputError(int) : void
- ▼  Term
 - coeff : int
 - expn : int