面向对象设计与构造第二次作业

第一部分：训练目标

通过对表达式结构进行建模，完成多项式的括号展开与函数调用、化简，进一步体会层次化设计的思想。

第二部分：预备知识

- 1、Java 基础语法与基本容器的使用。
- 2、扩展 BNF 描述的形式化表述。
- 3、正则表达式或其他解析方法。

第三部分：基本概念

一、基本概念的声明

- **带符号整数 支持前导 0的十进制带符号整数（若为正数，则正号可以省略），无进制标识。如：`+02`、`-16`、`19260817`等。**
- **因子**
  - **变量因子**
    - **幂函数**
      - **一般形式** 由自变量 x，指数符号 `**` 和指数组成，指数为一个符号不是 `-` 的整数，如：`x ** +2`。
      - **省略形式** 当指数为 1 的时候，可以省略指数符号 `**` 和指数，如：`x` 。
    - **三角函数**
      - **一般形式** 类似于幂函数，由`sin(<因子>)`或`cos(<因子>)` 、指数符号`**`和指数组成，其中：
        - `<因子>` 在本次作业中只可能是**常数因子**或者变量因子中的**幂函数**，如 `sin(1)` ， `sin(x**2)` 。
        - 指数为符号不是 `-` 的整数，如：`sin(x) ** +2`。
      - **省略形式** 当指数为 1 的时候，可以采用省略形式，省略指数符号`**`和指数部分，如：`sin(x)`。
      - 本指导书范围内的"三角函数"**仅包含`sin`和`cos`**。
    - **自定义函数**
      - 自定义函数的**定义**形如 `f(x, y, z) = 函数表达式` ，比如 `f(y) = y**2`，`g(x, y) = sin(x)*cos(y)`，`h(x, y, z) = x + y + z` 。
        - `f` 、`g`、`h` 是函数的**函数名。**在本次作业中，保证函数名**只使用** `f` ，`g`，`h`，且**不出现同名函数的重复定义**（因此每次最多只有 3 个自定义函数）。更具体的约束信息请看第六部分中的数据限制部分。
        - `x`、`y`、`z` 为函数的**形参。**在本次作业中，**形参个数为 1~3 个。**形参**只使用** `x`，`y`，`z`，且同一函数定义中不会出现重复使用的形参。
        - 函数表达式为一个关于形参的表达式。函数表达式的一般形式参见**形式化定义。**

        - 自定义函数的**调用**形如 `f(因子，因子，因子)`，比如 `f(x**2)`，`g(sin(x), cos(x))`，`h(1, 0, -1)`。
            - `因子` 为函数调用时的**实参**，只包含幂函数因子、三角函数因子和常数因子。
            - 例如有自定义函数 `f(x, y, z) = x + y + z`，函数调用 `f(sin(x), cos(x), x)`，则其结果为 `sin(x) + cos(x) + x`。保证函数调用的结果中只包含自变量 `x`
        - **求和函数**
            - 求和函数的一般形式为 `sum(i, s, e, t)`，其含义为 $\sum_{i=s}^{e} t$。
                - `i` 是**循环变量**，本次作业中循环变量只能是字符 `i`。
                - `s` 与 `e` 为求和的下限和上限，二者都应该是**常数因子**。在循环时，循环变量 `i` 的初值为 `s`，每次迭代 `i` 自增 1 并计算表达式的值，当 `i` **大于** `e` 时停止计算，并将计算结果求和。（若初始 `s > e`，则该函数结果为 `0`）。
                - 求和表达式 `t` 是一个因子，其中可以包含循环变量 `i`。其具体形式请参考**形式化定义**。
            - 例如，`sum(i, 1, 3, i)` 的化简结果是 `1 + 2 + 3` 或 `6`，`sum(i, 1, 5, x)` 的结果是 `5*x`，`sum(i, 1, 5, (i*x))` 是 `15*x`。
    - **常数因子** 包含一个带符号整数，如：`233`。
    - **表达式因子** 用一对小括号包裹起来的表达式，可以带指数，且指数为符号不是 `-` 的整数，例如 `(x**2 + 2*x)**2`。表达式的定义将在表达式的相关设定中进行详细介绍。
- **项** 由乘法运算符连接若干因子组成，如 `x * 02`。此外，**在第一个因子之前，可以带一个正号或负号**，如 `+ x * 02`、`- +3 * x`。注意，**空串不属于合法的项。**
- **表达式** 由加法和减法运算符连接若干项组成，如：`(-1 + x ** 233)* sin(x**2) ** 06 - cos(x) * 3 * sin(x)`。此外，**在第一项之前，可以带一个正号或者负号**，表示第一个项的正负，如：`- -1 + x ** 233 + -2 + x ** 1113`。
- **空白字符** 在本次作业中，空白字符仅包含空格 `<space>`（ascii 值 32）和水平制表符 `\t`（ascii 值 9）。其他的空白字符，均属于非法字符。

  对于空白字符，有以下几点规定：
    - 带符号整数内不允许包含空白字符，注意带符号整数本身的符号与整数之间也不允许包含空白字符。
    - 指数运算符内不允许包含空白字符，如 `*  *` 不合法。
    - 函数保留字内不允许包含空白字符，即 `sin`，`cos`，`sum` 关键字内不可以含有空白字符。

二、设定的形式化表述

- 表达式 → 空白项 [加减 空白项] 项 空白项 | 表达式 加减 空白项 项 空白项
- 项 → [加减 空白项] 因子 | 项 空白项 '*' 空白项 因子
- 因子 → 变量因子 | 常数因子 | 表达式因子
- 变量因子 → 幂函数 | 三角函数 | 自定义函数**调用** | 求和函数
- 常数因子 → 带符号的整数
- 表达式因子 → '(' 表达式 ')' [空白项 指数]
- 幂函数 → (函数自变量|'i') [空白项 指数]
- 三角函数 → 'sin' 空白项 '(' 空白项 因子 空白项 ')' [空白项 指数] | 'cos' 空白项 '(' 空白项 因子 空白项 ')' [空白项 指数]
- 指数 → '**' 空白项 ['+'] 允许前导零的整数 （与上次作业不同）
- 带符号的整数 → [加减] 允许前导零的整数
- 允许前导零的整数 → (0|1|2|...|9){0|1|2|...|9}
- 空白项 → {空白字符}
- 空白字符 → ' '（空格） | `\t`
- 加减 → '+' | '-'

- 自定义函数**定义** → 自定义函数名 空白项 '(' 空白项 函数自变量 空白项 [',' 空白项 函数自变量 空白项 [',' 空白项 函数自变量 空白项]] ')' 空白项 '=' 空白项 函数表达式
- 函数自变量 → 'x' | 'y' | 'z'
- 自定义函数**调用** → 自定义函数名 空白项 '(' 空白项 因子 空白项 [',' 空白项 因子 空白项 [',' 空白项 因子 空白项]] ')'
- 自定义函数名 → 'f' | 'g' | 'h'
- 求和函数 → 'sum' '(' 空白项 'i' 空白项',' 空白项 常数因子 空白项 ',' 空白项 常数因子 空白项 ',' 空白项 求和表达式 空白项 ')'
- 函数表达式 → 表达式
- 求和表达式 → 因子

其中

- `{}` 表示允许存在 0 个、1 个或多个。
- `[]` 表示允许存在 0 个或 1 个。
- `()` 内的运算拥有更高优先级，类似数学中的括号。
- `|` 表示在多个之中选择一个。
- 上述表述中使用单引号包裹的串表示字符串字面量，如 '(' 表示字符 `(`。

式子的具体含义参照其数学含义。

若输入字符串能够由"表达式"推导得出，则输入字符串合法。具体推导方法请参阅 **"第一单元形式化表述说明"文档**。

除了满足上述形式化表述之外，我们本次作业的输入数据的**额外限制**请参见**第六部分：输入/输出说明 的数据限制部分**。

第四部分：题目描述

本次作业中需要完成的任务为：读入**一系列自定义函数的定义**以及一个包含简单幂函数、简单三角函数、简单自定义函数调用以及求和函数的**表达式**，输出**恒等变形展开所有括号后**的表达式。

在本次作业中，**展开所有括号**的定义是：对原输入表达式 E 做**恒等变形**，得到新表达式 E'。其中，E' 中不再含有自定义函数与求和函数，且只包含**必要的括号**。

第五部分：设计建议

- 在 Java 内，不建议使用静态数组。推荐使用 `ArrayList` 、 `HashMap` 、 `HashSet` 一类的数据结构，快速管理和调配手中无序的数据。
- 对于使用一般输入手动解析的同学，处理字符串时可以考虑使用**正则表达式**，相关的 API 可以了解 `Pattern` 和 `Matcher` 类；另外也可以考虑使用**递归下降**等方法进行解析，相关教程我们会放在实验或者训练中。
- 这次作业看上去似乎很难，其实找对了方法后并不难。关键思想是"化整为零"，可以这样考虑：
  - 对于每一种函数（常数、幂函数、三角函数、求和函数、自定义函数），分别建立类
  - 对于每一种运算规则（乘法、加减法），分别建立类
  - 对于自定义函数，可以先将其 `定义表达式` 展开，将展开后的结果代入进行计算。
  - 对于求和函数，可以类似于自定义函数进行代入

第六部分：输入/输出说明

一、公测说明

输入格式

本次作业的输入数据包含若干行：

- 第一行为一个整数 n $(0 \leq n \leq 3)$，表示**自定义函数定义的个数。**
- 第 2 到第 $n+1$ 行，每行为一行字符串，表示一个自定义函数的定义。
- 第 $n+2$ 行，一行字符串，表示待展开表达式。

读入方法

在读入时，我们要求同学们**必须使用**课程组提供的官方读入工具包。

官方读入工具包提供两种读入模式：**一般读入模式**和**预解析读入模式。**

- **一般读入模式**会从标准输入中一次读入一行输入的自定义函数定义或者表达式，并原封不动地返回，其行为类似于使用 `scanner` 从标准输入中读取一行字符串。
- **预解析读入模式**会从标准输入中读入输入的自定义函数定义以及表达式，接着对该表达式以及自定义函数定义**预先做一定的解析**以方便同学们之后的处理。通过调用官方包提供的方法得到**若干行**，表示解析后得到的**子表达式个数**以及**每个子表达式字符串。**其中，解析后的每个**子表达式**的格式形如 `标签 运算符 若干操作数或已有的标签` 。**标签**指的是每行字符串的唯一的 id，保证每行字符串的标签不同；**运算符**用字符串表示，涉及到的所有运算符的定义如下：

运算符	原表达式	运算符字符串表示	操作数
+	x + 1	add	x 1
+	+ x	pos	x
-	x - x	sub	x x
-	- x	neg	x
*	x * x	mul	x x
**	x ** 3	pow	x 3
sin	sin(x)	sin	x
cos	cos(x)	cos	x
null(无运算符)	193	null(无运算符)	193

（官方包预解析模式本次已经把求和函数 `sum` 做了展开，因此上表中没有求和函数运算符）

操作数可以是 `x`，也可以是带符号整数。保证已有标签值小于当前标签值，即已有标签在当前子表达式之前已出现。**最后**一个标签（`fn`）对应的值即为需要求解的值。

预解析读入模式对读入的自定义函数以及表达式处理后得到的完整输出的例子请见**样例部分。**

输出格式

输出包括**两行**，第一行是**设置官方包读入模式后自动输出的标签**， 第二行表示展开括号之后的表达式。

数据限制

- 输入表达式**一定满足**基本概念部分给出的**形式化描述。**
- `sin(<因子>)`、`cos(<因子>)` 中的 `<因子>`，在本次作业中只能是**常数因子**或**幂函数。**在待展开表达式中出现 `sin(<因子>)` 和 `cos(<因子>)` 时，幂函数自变量只能是 `x`。
- 求和函数的**求和表达式中**不能出现**求和函数**或者**自定义函数。**
- **自定义函数定义**满足以下限制：
  - 至多只有一层多余括号。（除三角函数调用的括号（`sin()` 与 `cos()`）外，均属于多余括号）
  - 不会出现重名函数。
  - 函数表达式中不允许使用求和函数，不允许调用其他自定义函数或自己（即不允许递归）
  - 函数形参不能重复出现，即无需考虑 `f(x,x)=x**2+x` 这类情况。
- 对于规则 “自定义函数**调用** → 自定义函数名 '(' 空白项 因子 空白项 [',' 空白项 因子 空白项 [',' 空白项 因子 空白项]] ')'”，其中的因子**仅包含常数因子、幂函数因子和三角函数因子，不包含表达式因子。**且本条规则中幂函数自变量只能是 `x`（不包含 `y` 等）。
- 对于规则 “指数 → ** 空白项 ['+'] 允许前导零的整数”，我们本次要求**输入数据的指数不能超过 8。**（**本条与上次作业不同**）
- 最后一行输入的待展开表达式中的多余括号至多只有一层，且只含有自变量 `x`（不含 `y` 等）。其**有效长度**至多为 100 个字符，每个单个自定义函数**定义**的**有效长度**至多为 100 个字符。其中**有效长度**指的是去除掉所有**空白符**后剩余的字符总数。
- 输出的结果的**有效长度**至多为 10000 个字符，如果超过该限制，那么该输出结果将会被判为错误。下面我们会介绍一个完成本次作业的基准思路，如按照本思路进行设计和实现程序，则可以保证**公测**中你的程序的输出不会超过长度限制。在此基础上，你还可以进行力所能及的优化。

基准思路

- 对于形如 $(\sum_i a_i) + (\sum_j b_j)$ 的部分，将其展开为 $\sum_i a_i + \sum_j b_j$。
- 对于形如 $(\sum_i a_i) - (\sum_j b_j)$ 的部分，将其展开为 $\sum_i a_i + \sum_j (-b_j)$。
- 对于形如 $(\sum_i a_i) \times (\sum_j b_j)$ 的部分，将其展开为 $\sum_i \sum_j a_i b_j$，且不需要进行同类项的合并。
- 对于形如 a^b 的部分，若 a 含有括号，则将其展开为 $a \times a \times a \ldots \times a$，共有 b 个 a，接着使用第三条的思路进行展开。
- 对于形如 $\mathrm{sum}(i, \mathrm{begin}, \mathrm{end}, \mathrm{expr}(i))$ 的表达式，其中 expr 中包含循环变量 i，则将其展开为 $\mathrm{expr}(\mathrm{begin}) + \mathrm{expr}(\mathrm{begin}+1) + \ldots + \mathrm{expr}(\mathrm{end})$。若每个 $\mathrm{expr}(i)$ 中仍然含有括号，则需要按照前面的化简规则进行化简。
- 对于形如 $\mathrm{sum}(i, \mathrm{begin}, \mathrm{end}, \mathrm{expr})$ 的表达式，其中 expr 中**不**包含循环变量 i，则需要展开成 $\mathrm{expr} + \mathrm{expr} + \ldots + \mathrm{expr}$，共有 $\mathrm{end} - \mathrm{begin} + 1$ 个。若每个 expr 中仍然含有括号，则需要按照前面的化简规则进行化简。
- 对于形如 $\sin(\mathrm{expr}), \cos(\mathrm{expr})$ 的表达式，使用上述规则将 expr 的括号展开即可。

判定模式

本次作业中，对于每个测试点的判定分为**正确性判定**和**性能判定**，并会根据**所选的读入模式**对得分进行修正。其中，正确性判定为 80 分，性能判定部分为 20 分，二者之和为总分。对于使用了**预解析读入模式**的同学，其最后得分会在原始分数基础上乘以 0.85。

注意：**获得性能分的前提是，在正确性判定环节被判定为正确。**如果被判定为错误，则性能分部分为 0 分。

正确性判定：

- 输出的表达式须符合表达式的**形式化描述**，需要**展开所有括号**且与保持原表达式**恒等。**

- **展开所有括号**的定义：对原输入表达式 E 做**恒等变形**，得到新表达式 E'。其中，E' 中不再含有自定义函数与求和函数，且只包含**必要的括号**。
- 本次作业中**必要的括号**为三角函数调用中包含的括号：`sin()` 与 `cos()`。
- 本次作业中对于恒等的定义：设 $f(x)$ 的定义域为 D_1，D_1 包含于 R，$g(x)$ 的定义域为 D_2，D_2 包含于 R，对任意 $x \in D_1 \cap D_2$，$f(x) = g(x)$ 成立。

性能判定：

- 在本次作业中，性能分的唯一评判依据是**输出结果的有效长度**，有效长度的定义在**数据限制部分**已经给出。
- 设某同学给出的**正确答案**的有效长度为 L_p，目前**所有人**给出的正确答案中有效长度**最小的**为 L_{min}。

  记 $x = \frac{L_p}{L_{min}}$，则该同学**性能分百分比**为：

$$r(x) = 100\% \cdot \begin{cases} 1 & x = 1 \\ -31.8239x^4 + 155.9038x^3 - 279.2180x^2 + 214.0743x - 57.9370 & 1 < x \leq 1.5 \\ 0 & x > 1.5 \end{cases}$$

  举例来说，就是这样：

x	$r\left(x\right)$
1.0	100.0%
1.05	79.9%
1.1	60.5%
1.2	29.0%
1.3	10.9%
1.4	4.5%
1.5	0.0%

  该答案得到的性能分即为 $r(x) \times 20$。

二、互测说明

互测时，你可以通过提交**输入数据**和**期望得到的正确的输出**，该组数据会被用来测试同一个互测房间中的其他同学的程序。

注意，你提交的输出只需要包含一行，即输出的表达式，**不必包含官方包输出的标签**。

数据限制

- 输入表达式**一定满足**基本概念部分给出的**形式化描述**。
- 互测中不允许出现**自定义函数**和**求和函数**。
- `sin(<因子>)`、`cos(<因子>)` 中的 `<因子>`，在本次作业中只能是常数因子或者自变量，且如果是在待展开表达式中出现 `sin(<因子>)` 和 `cos(<因子>)`，则自变量只能是 `x`。（**本条与公测部分的限制不同**）
- 待展开表达式中自变量只可能出现 `x`，且多余括号至多只有一层。

- 对于规则 “指数 → ** 空白项 ['+'] 允许前导零的整数”，**本次对指数的大小无额外限制**，但是过大的指数可能导致答案过长。（**本条与上次作业不同**）
- 输入表达式的**有效长度**至多为 **50** 个字符。其中输入表达式的**有效长度**指的是输入表达式去除掉所有**空白符**后剩余的字符总数。（**本条与公测部分的限制不同**）
- 除此之外，为了限制不合理的 hack，我们要求输入表达式的代价 `Cost(Expr) <= 10000`，其中表达式代价的计算方法如下（**本条与公测部分的限制不同**）：
  - `Cost(常数) = max(2, len(常数))`
  - `Cost(x) = 2`
  - `Cost(a + b) = Cost(a - b) = Cost(a) + Cost(b) + 1`
  - `Cost(a * b) = Cost(a) * Cost(b) * (Cost(a) + Cost(b))`（多项相乘时从左到右计算）
  - `Cost(a ** b)= Cost(a)^b * b`
  - `Cost(sin(a)) = Cost(cos(a)) = Cost(a) + 5`
  - `Cost(+a) = Cost(-a) = Cost(a) + 1`
- 满足上述要求后，我们会用官方的**基准思路**程序对输入数据进行检验，输出有效长度不超过 **1000** 的视为合法互测数据。（**本条与公测部分的限制不同**）

如果提交的数据不满足上述数据限制，则该数据将被系统拒绝，且不会用来对同屋其他被测程序进行测试。

三、样例

一般读入模式

#	输入	输出	说明
1	1 f(x)=x**2 (x+f(x))*x	x**2+x**3	带入 f(x)之后得到(x+x**2)*x，展开括号可得结果
2	1 f(x)=x*(sin(x)+cos(x)) x-(f(x)+x)	-x*sin(x)-x*cos(x)	带入 f(x)之后展开可得到结果
3	1 f(x)=(x+1)**2 f(sin(x))+cos(x)	sin(x)**2+2*sin(x)+cos(x)+1	f(x)的实参是三角函数变量因子
4	1 f(x,y,z)=x+y**2+z**3 f(sin(x)**2,cos(x),x)	sin(x)**2+cos(x)**2+x**3	多变量自定义函数的情况，保证最后变量只有 x
5	2 f(x,y)=x+y g(x,y)=x*y x*f(sin(x),cos(x))+g(sin(x),cos(x))	x*sin(x)+x*cos(x)+sin(x)*cos(x)	多个自定义函数
6	0 x*(sum(i,1,10,(x+1)))	10*x**2+10*x	求和函数
7	1 f(x,y)=(x+1)**2+(y-1)**2 f(x,x)+sum(i,1,5,x)	2*x**2+5*x+2	既包含自定义函数又包含求和函数

预解析读入模式

#	输入	解析后输入	输出	说明
1	1 f(x)=x**2 (x+f(x))*x	3 f1 pow x 2 f2 add x f1 f3 mul f2 x	x**2+x**3	带入 f(x)之后得到(x+x**2)*x，展开括号可得结果
2	1 f(x)=x*(sin(x)+cos(x)) x-(f(x)+x)	6 f1 sin x f2 cos x f3 add f1 f2 f4 mul x f3 f5 add f4 x f6 sub x f5	-x*sin(x)-x*cos(x)	带入 f(x)之后展开可得到结果
3	1 f(x)=(x+1)**2 f(sin(x))+cos(x)	5 f1 sin x f2 add f1 1 f3 pow f2 2 f4 cos x f5 add f3 f4	sin(x)**2+2*sin(x)+cos(x)+1	f(x)的实参是三角函数变量因子

4	1 f(x,y,z)=x+y**2+z**3 f(sin(x)**2,cos(x),x)	7 f1 sin x f2 pow f1 2 f3 cos x f4 pow f3 2 f4 pow f3 2 f5 add f2 f4 f6 pow x 3 f7 add f5 f6	sin(x)**2+cos(x)**2+x**3	多变量自定义函数的情况，保证最后变量只有 x

5	2 f(x,y)=x+y g(x,y)=x*y x*f(sin(x),cos(x))+g(sin(x),cos(x))	8 f1 sin x f2 cos x f3 add f1 f2 f4 mul x f3 f5 sin x f6 cos x f7 mul f5 f6 f8 add f4 f7	x*sin(x)+x*cos(x)+sin(x)*cos(x)	多个自定义函数
6	0 x*(sum(i,1,10,(x+1)))	20 f1 add x 1 f2 add x 1 f3 add f1 f2 f4 add x 1 f5 add f3 f4 f6 add x 1 f7 add	10*x**2+10*x	求和函数

		f5 f6 f8 add x 1 f9 add f7 f8 f10 add x 1 f11 add f9 f10 f12 add x 1 f13 add f11 f12 f14 add x 1 f15 add f13 f14 f16 add x 1 f17 add f15 f16 f18 add x 1 f19 add f17 f18 f20 mul x f19		

7	1 f(x,y)=(x+1)**2+(y-1)**2 f(x,x)+sum(i,1,5,x)	10 f1 add x 1 f2 pow f1 2 f3 sub x 1 f4 pow f3 2 f5 add f2 f4 f6 add x x f7 add f6 x f8 add f7 x f9 add f8 x f10 add f5 f9	2*x**2+5*x+2	既包含自定义函数又包含求和函数

注意：由于本作业可被判定为正确的答案不唯一，以上样例的输出**仅保证正确性，但并不一定为性能最优解。**

第七部分：提示与警示

一、提示

- Java 内的原生整数类型有 `long` 和 `int`，长度分别为 64 位和 32 位。
- 如果觉得上述数据类型不够用，可以搜索一下 Java 内可以怎样快速处理这个问题，也可以回顾一下 pre 第 2 弹的指导书。
- **不要重复造轮子！不要重复造轮子！不要重复造轮子！**~~重要的事情说三遍~~
- 我们鼓励大家通过 Baidu、Google、Stack Overflow 等方式自行学习和解决问题。
- 如果还有更多的问题，请到讨论区提问。但是**请善用讨论区**，并在此之前认真阅读包括但不限于课程要求文档、指导书、搜索引擎结果等的内容。关于如何提问。

二、警示

- 正式评测时官方包模式选定后的标签输出**会被替换并加密**，因此**请不要尝试 hack 官方输入包。**
- 如果在互测中发现其他人的代码疑似存在**抄袭**或者 **hack 官方包**等行为，可向课程组举报，课程组感谢同学们为 OO 课程建设所作出的贡献。