2021 年第 4 次作业：语法制导翻译

Hollow Man

1

（50 分）写一个 SDD，完成下面的题目：

给非终结符 E 一个综合属性 v，其值可取 lvalue 或 rvalue，分别表示 E 是左值表达式和右值表达式，那么语法制导定义如下（无输出则表示无错）：

	产生式	语法规则
1)	E -> E1 + E2	E.v = rvalue
2)	E -> (E1)	E.v = E1.v
3)	E -> E1++	if E1.v == rvalue then printf("invalid lvalue in increment");
		E.v := rvalue
4)	E -> id	E.v := lvalue
5)	E -> num	E.v := rvalue

2

（50 分）以作业二中的后缀表达式文法为基础：

S -> S S + | S S * | id

设计一个语法制导定义（SDD），将每一个输入的后缀表达式转换为等价的中缀表达式，但不带冗余括号。如：输入 ab*cd++ ，输出 a*b+(c+d) ；输入 ab*cd+*e+ ，输出 a*b*(c+d)+e 。

属性的含义：
code：生成的中缀表达式。

op: 当前表达式的符号。

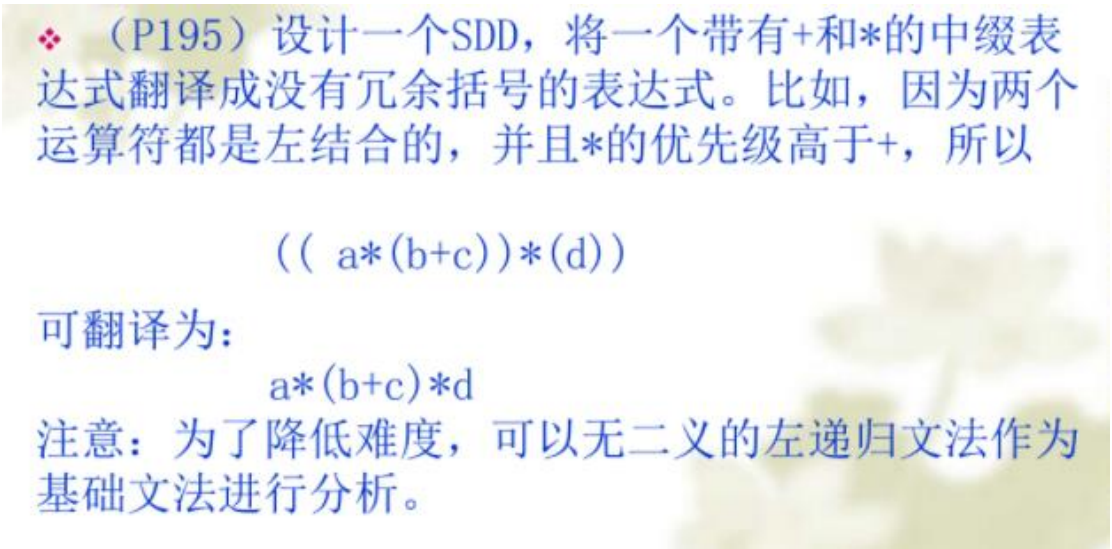
	产生式	语法规则
1)	L -> Sn	printf(Sn.code)
2)	Sn -> S1 S2 +	if S2.op=="+" then Sn.code=S1.code "+ (" S2.code ")"
		else Sn.code=S1.code "+" S2.code
		Sn.op="+"
3)	Sn -> S1 S2 *	if S2.op=="*" then S2.code=(" S2.code ")"
		if S1.op=="+" && S2.op=="+" then Sn.code="(" S1.code ") * (" S2.code ")"
		else if S1.op=="+" then Sn.code="(" S1.code ") *" S2.code
		else if S2.op=="+" then Sn.code=S1.code "* (" S2.code ")"
		else Sn.code=S1.code "*" S2.code
		Sn.op="*"
4)	Sn -> id	Sn.code=id
		Sn.op=""

二.论述题 (共 1 题,33.4 分)

1

（100 分）如果觉得第一大题的两道题目有点牛刀小试的感觉，可以考虑玩玩下面的这道题（5.1 节 PPT 中有）。

注意：全部完成第一大题的两道小题即视为完成本次作业，只完成第二大题同样视为完成本次作业；两大题都正确完成者可以申请加分（微信单独申请，直接加平时成绩的总分）。大题题号后的分数是系统加的，不要管它。



参考： https://github.com/fool2fish/dragon-book-exercise-

answers/blob/master/ch05/5.3/5.3.md#532-

使用语法：

L -> En

E -> E1 + T

E -> T

T -> T1 * F

T -> F

F -> (E)

F -> digit

其中 En 包括了 E 和 E1。

属性的含义：

precedence：令 +，*和单 digit 的优先级分别为 0，1，2。 如果表达式最外层有括号，则为去掉括号后最后被计算的运算符的优先级，否则为表达式最后被计算的运算符的优先级。

expr：表达式。

cleanExpr：去除了冗余括号的表达式。

	产生式	语法规则
1)	L -> E	printf(E.cleanExpr)
2)	E -> E1 + T	E.precedence = 0
		E.expr = E1.expr "+" T.expr
		E.cleanExpr = E1.cleanExpr "+" T.cleanExpr
3)	E -> T	E.precedence = T.precedence
		E.expr = T.expr
		E.cleanExpr = E.cleanExpr
4)	T -> T1 * F	T.precedence = 1
		T.expr = T1.expr "*" F.expr
		T.cleanExpr = (T1.precedence >= 1 ? T1.cleanExpr : T1.expr) "*" (F.precedence > 1 ? F.cleanExpr : F.expr)
5)	T -> F	T.precedence = F.precedence
		T.expr = F.expr
		T.cleanExpr = F.cleanExpr
6)	F -> (E)	F.precedence = E.precedence
		F.expr = "(" E.cleanExpr ")"
		F.cleanExpr = E.cleanExpr
7)	F -> digit	F.precedence = 2
		F.expr = digit
		F.cleanExpr = digit