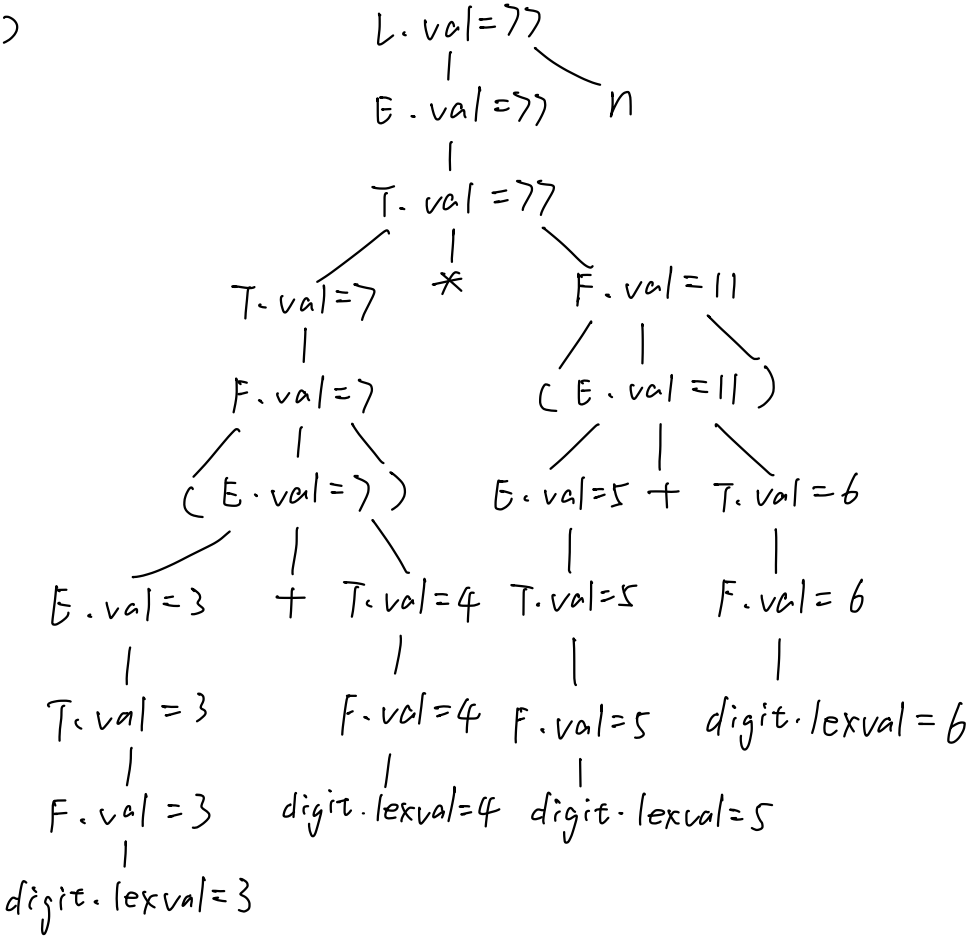
编译原理 - 作业(4)：语义分析

Q1: (P309, Exercise 5.1.1) For the SDD below, give annotated parse trees for the following expressions:

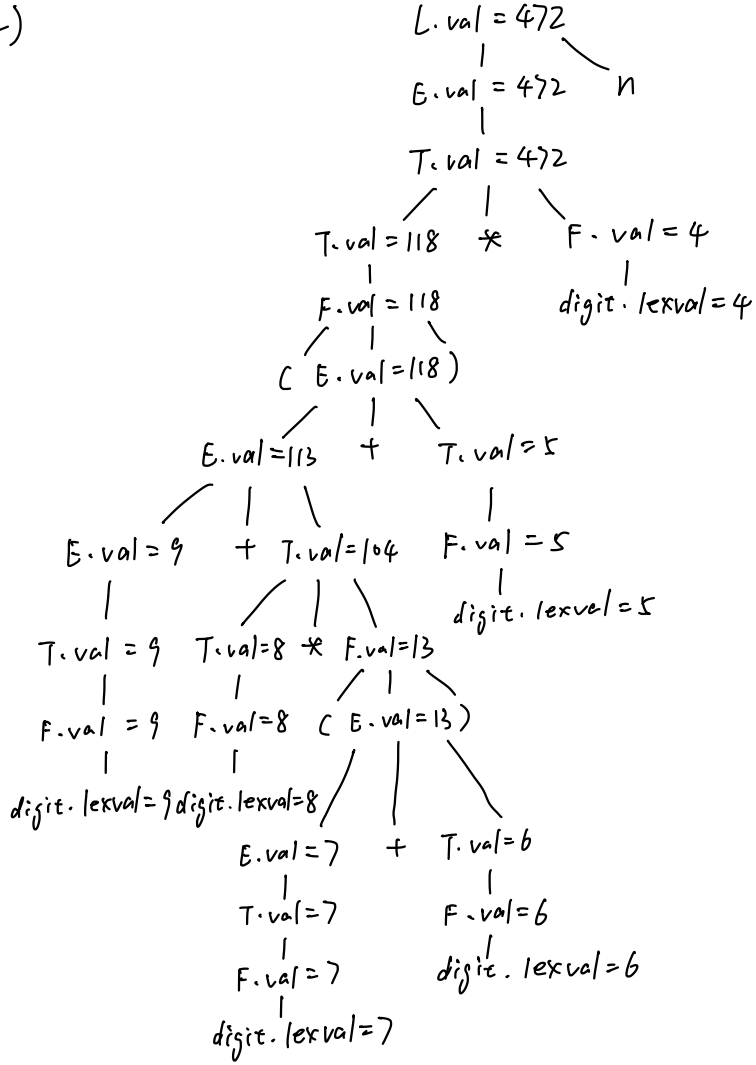
PRODUCTIONS	SEMANTIC RULES
1) $L \rightarrow E$ **n**	L.val = E.*val*
2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
3) $E \rightarrow T$	$E.val = T.val$
4) $T \rightarrow T_1 * F$	$T.val = T_1.val$ x $F.val$
5) $T \rightarrow F$	$T.val = F.val$
6) $F \rightarrow (E)$	$F.val = E.val$
7) $F \rightarrow$ digit	$F.val =$ digit.*lexval*

(1) (3 + 4) * (5 + 6) **n**
(2) (9 + 8 * (7 + 6) + 5) * 4 **n**

(1)



(2)



Q2: (p323, Exercises 5.3.1) Below is a grammar for expressions involving operator + and integer of floating-point operands. Floating-point numbers are distinguished by having a decimal point:

$E \rightarrow E + T \mid T$

$L \rightarrow$ num . num | num

Give an SDD to determine the type of each term T and expression E.

Production Rules	Semantic Rules
(1) $E \rightarrow E_1 + T$	E.type = E1.type == float || T.type == float ? float : int
(2) $E \rightarrow T$	E.type = T.type
(3) $T \rightarrow$ num.num	T.type = float
(4) $T \rightarrow$ num	T.type = int

Q3: (p317, Exercises 5.2.4) This grammar generates binary numbers with a "decimal" point:

$S \rightarrow L \,.\, L \mid L$

$L \rightarrow L\, B \mid B$

$B \rightarrow 0 \mid 1$

(1) Design an L-attributed SDD to compute *S.val*, the decimal number value of an input string. For example, the translation of string 101.101 should be the decimal number 5.625. Hint: use an inherited attribute *L.side* that tells which side of the decimal point a bit is on.

(2) Draw the annotated parse tree of 101.101.

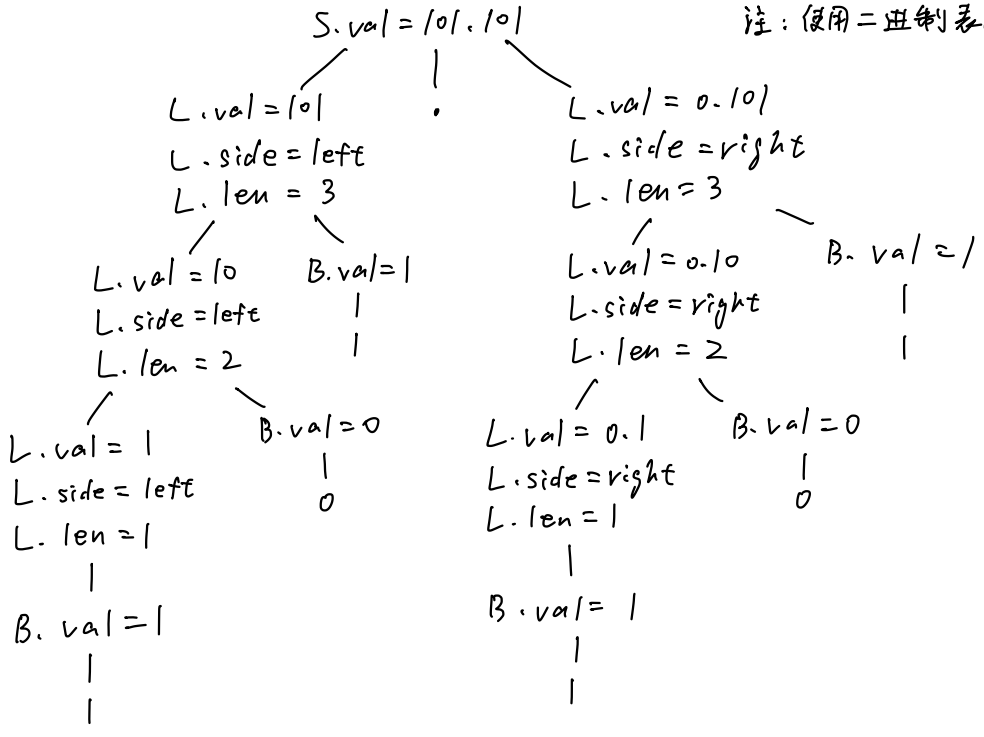
(1)

Production Rules	Semantic Rules
(1) $S \rightarrow L_1 . L_2$	$S.val = L_1.val + L_2.val$ $L_1.side = left$ $L_2.side = right$
(2) $S \rightarrow L$	$S.val = L.val$ $L.side = left$
(3) $L \rightarrow L_1 B$	$L_1.side = L.side$ $L.len = L_1.len + 1$ $L.val = L.side == left \,?\, L_1.val * 2 + B.val : L_1.val + B.val * 2^{-L.len}$
(4) $L \rightarrow B$	$L.len = 1$ $L.val = L.side == left \,?\, B.val : B.val / 2$
(5) $B \rightarrow 0$	$B.val = 0$
(6) $B \rightarrow 1$	$B.val = 1$

繼承屬性：side

綜合屬性：val , len

(2)



注：使用二进制表示

Q4: For the code snippet below：

```
1:  int x = 0;
2:  float y = 0.0;
3:  while (x < 10) {
4:    int y, z;
5:    y = x;
6:    z = 0;
7:    while (y < 10) {
8:      z = z + y;
9:      y = y + 1;
10:   }
11: }
12: y = x * 1.0;
```

Regarding the semantic analysis of variable type, we consider the following simplified grammar and syntax-directed translation (SDT):

1) $D \rightarrow T\ \{\ L.type = T.type\ \}\ L$
2) $T \rightarrow \text{int}\ \{\ T.type = \text{int}\ \}$
3) $T \rightarrow \text{float}\ \{\ T.type = \text{float}\ \}$
4) $L \rightarrow \{\ L_1.type = L.type\ \}$
   L_1, id $\{$ *addtype*(id.*entry*, *L.type*)
5) $L \rightarrow$ id $\{$ *addtype*(id.*entry*, *L.type*)

(1) In the above SDT, both *T* and *L* have attribute 'type'. The type attribute is synthesized or inherited? Please explain.

(1) T.type 是综合属性，L.type 是继承属性

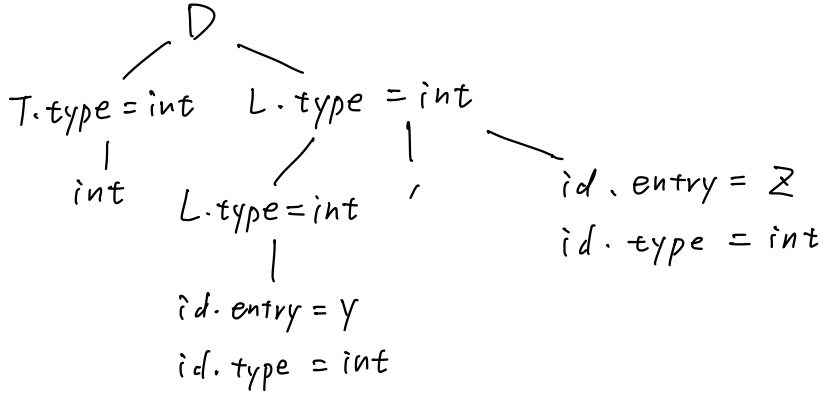
比如，T → int { T.type = int } 中 T.type 从子节点获得

L → { L1.type = L.type } L1 , id { addtype(id.entry , L.type) } 中

L.type 从父节点获得

(2) For Line 4 of the code snippet：int y, z; Construct the annotated parse tree based on the above SDT.

(2)



(3) For Lines 3, 7 and 11 of the code snippet, list the valid variables (name and type) in symbol table.

name	type	
x	int	3
y	int	7
x	int	12
y	float	