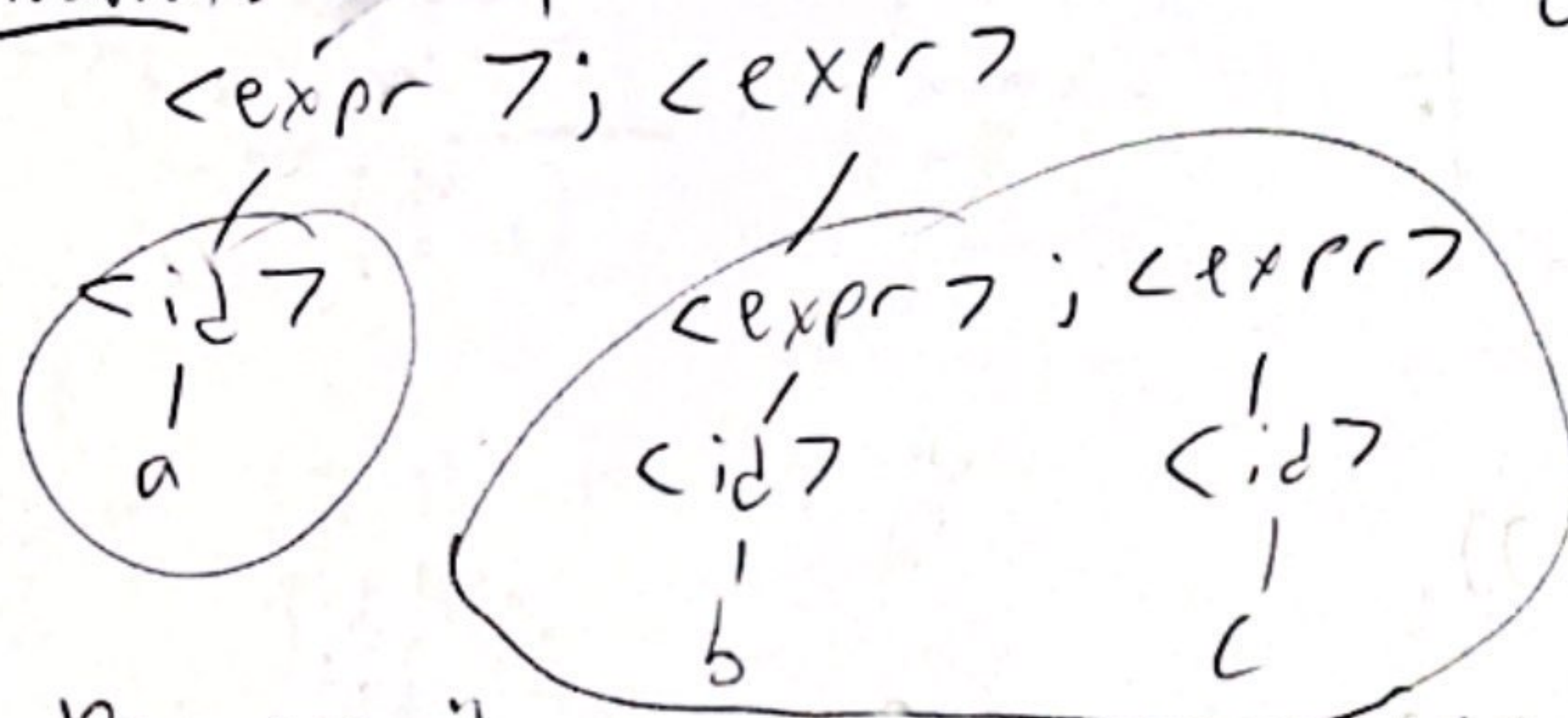


# CS320 Assign b Written

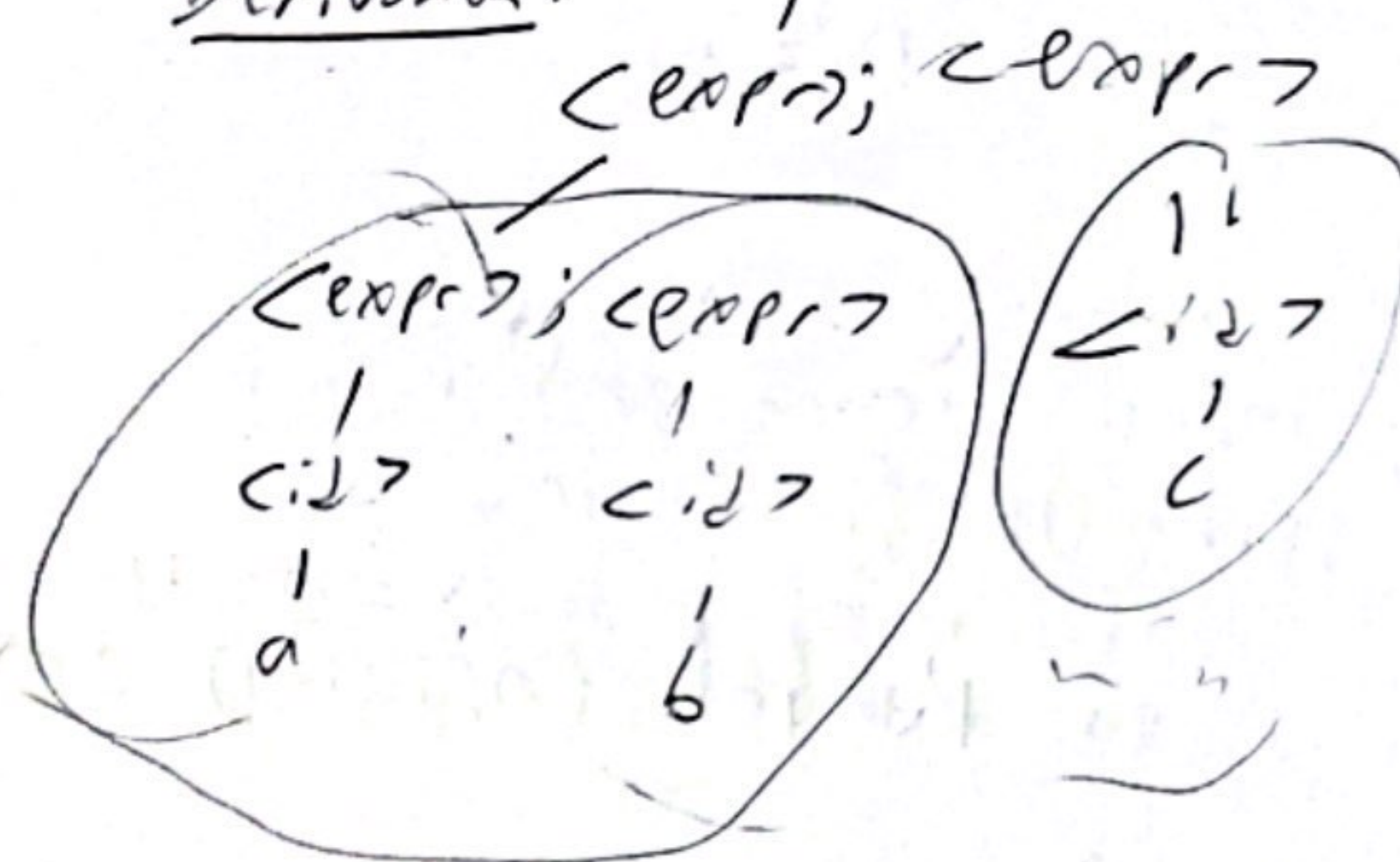
1. So we try to parse a;b;c

Leftmost  $\langle \text{expr} \rangle$   
Derivation:



Or

Rightmost  $\langle \text{expr} \rangle$   
Derivation:



Because "a;b;c" has two distinct parse trees, the grammar is ambiguous

2. Modified grammar

$\langle \text{id} \rangle ::= a | b | c | \dots | z$

$\langle \text{dig} \rangle ::= 0 | 1 | 2 | \dots | 9$

$\langle \text{symb} \rangle ::= \langle \text{id} \rangle | \langle \text{dig} \rangle | ( )$

$\langle \text{term} \rangle ::= \text{let } \langle \text{id} \rangle = \langle \text{expr} \rangle \text{ in } \langle \text{expr} \rangle$

$\langle \text{term-in} \rangle ::= \text{let } \langle \text{id} \rangle = \langle \text{expr} \rangle \text{ in } \langle \text{expr} \rangle$

$\langle \text{beg-end} \rangle ::= \text{begin } \langle \text{expr} \rangle \text{ end}$

$\langle \text{exprs} \rangle ::= \langle \text{symb} \rangle ; \langle \text{exprs} \rangle$   
 $\quad \quad \quad | \langle \text{term-in} \rangle ; \langle \text{expr} \rangle$   
 $\quad \quad \quad | \langle \text{beg-end} \rangle ; \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= \langle \text{symb} \rangle$   
 $\quad \quad \quad | \langle \text{term-in} \rangle$   
 $\quad \quad \quad | \langle \text{beg-end} \rangle$   
 $\quad \quad \quad | \langle \text{exprs} \rangle$

3. The previously-demonstrated ambiguity arose

because the  $\langle \text{expr} \rangle ; \langle \text{expr} \rangle$  grammar was associative. Note that other grammars (like  $\langle \text{symb} \rangle$ ,  $\langle \text{term} \rangle$ , and  $\langle \text{beg-end} \rangle$ ) are just sugar in this grammar. These are not associative. By switching the  $\langle \text{expr} \rangle ; \langle \text{expr} \rangle$  grammar to the  $\langle \text{exprs} \rangle$  grammar, the derivation tree becomes right-associative