

1.

$\langle \text{assign} \rangle \rightarrow \langle \text{id} \rangle = \langle \text{expr} \rangle$

$\langle \text{id} \rangle \rightarrow A \mid B \mid C$

$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle * \langle \text{term} \rangle \mid \langle \text{term} \rangle$

$\langle \text{term} \rangle \rightarrow \langle \text{factor} \rangle + \langle \text{term} \rangle \mid \langle \text{factor} \rangle$

$\langle \text{factor} \rangle \rightarrow (\langle \text{expr} \rangle) \mid \langle \text{id} \rangle$

2.

(a)

$\langle \text{assign} \rangle \Rightarrow \langle \text{id} \rangle = \langle \text{expr} \rangle$

$\Rightarrow A = \langle \text{expr} \rangle$

$\Rightarrow A = \langle \text{id} \rangle * \langle \text{expr} \rangle$

$\Rightarrow A = A * \langle \text{expr} \rangle$

$\Rightarrow A = A * (\langle \text{expr} \rangle)$

$\Rightarrow A = A * (\langle \text{id} \rangle + \langle \text{expr} \rangle)$

$\Rightarrow A = A * (B + \langle \text{expr} \rangle)$

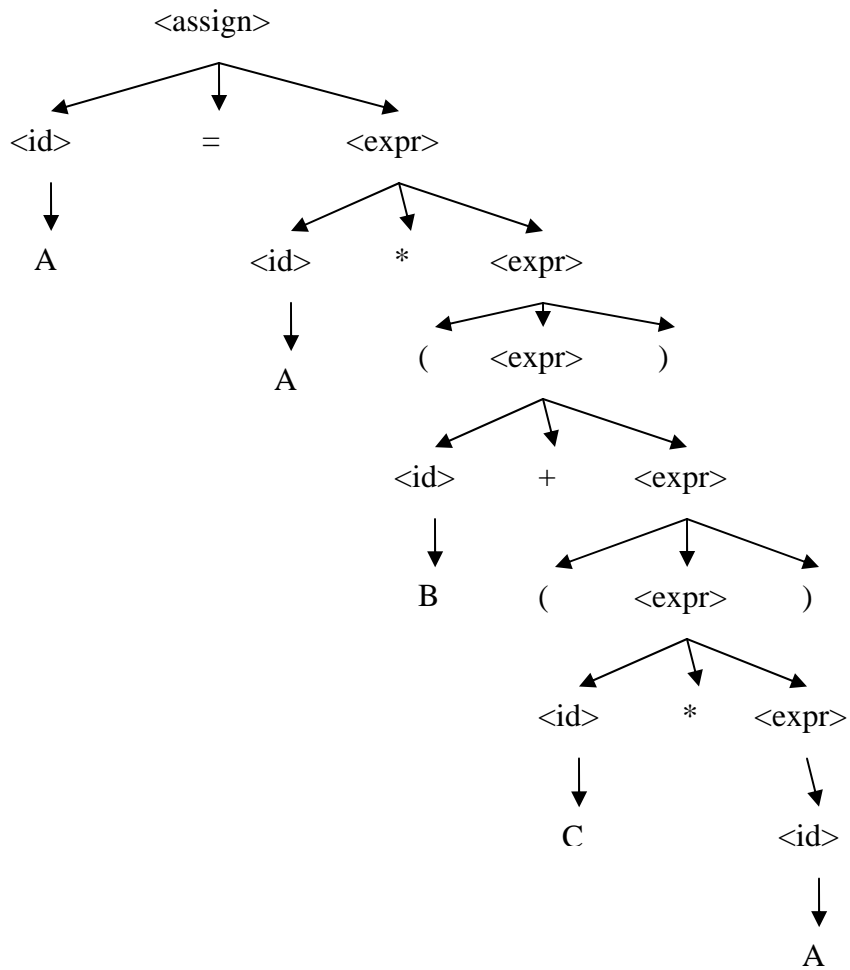
$\Rightarrow A = A * (B + (\langle \text{expr} \rangle))$

$\Rightarrow A = A * (B + (\langle \text{id} \rangle * \langle \text{expr} \rangle))$

$\Rightarrow A = A * (B + (C * \langle \text{expr} \rangle))$

$\Rightarrow A = A * (B + (C * \langle \text{id} \rangle))$

$\Rightarrow A = A * (B + (C * A))$



(b)

$\langle \text{assign} \rangle \Rightarrow \langle \text{id} \rangle = \langle \text{expr} \rangle$

$\Rightarrow B = \langle \text{expr} \rangle$

$\Rightarrow B = \langle \text{id} \rangle * \langle \text{expr} \rangle$

$\Rightarrow B = C * \langle \text{expr} \rangle$

$\Rightarrow B = C * (\langle \text{expr} \rangle)$

$\Rightarrow B = C * (\langle \text{id} \rangle * \langle \text{expr} \rangle)$

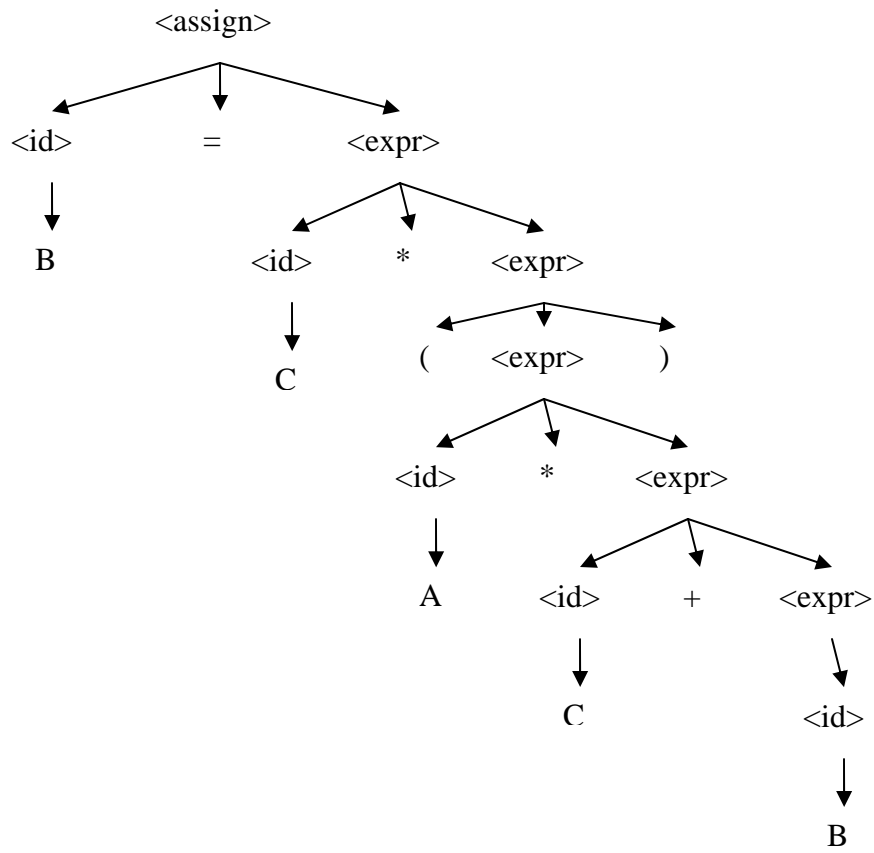
$\Rightarrow B = C * (A * \langle \text{expr} \rangle)$

$\Rightarrow B = C * (A * \langle \text{id} \rangle + \langle \text{expr} \rangle)$

$\Rightarrow B = C * (A * C + \langle \text{expr} \rangle)$

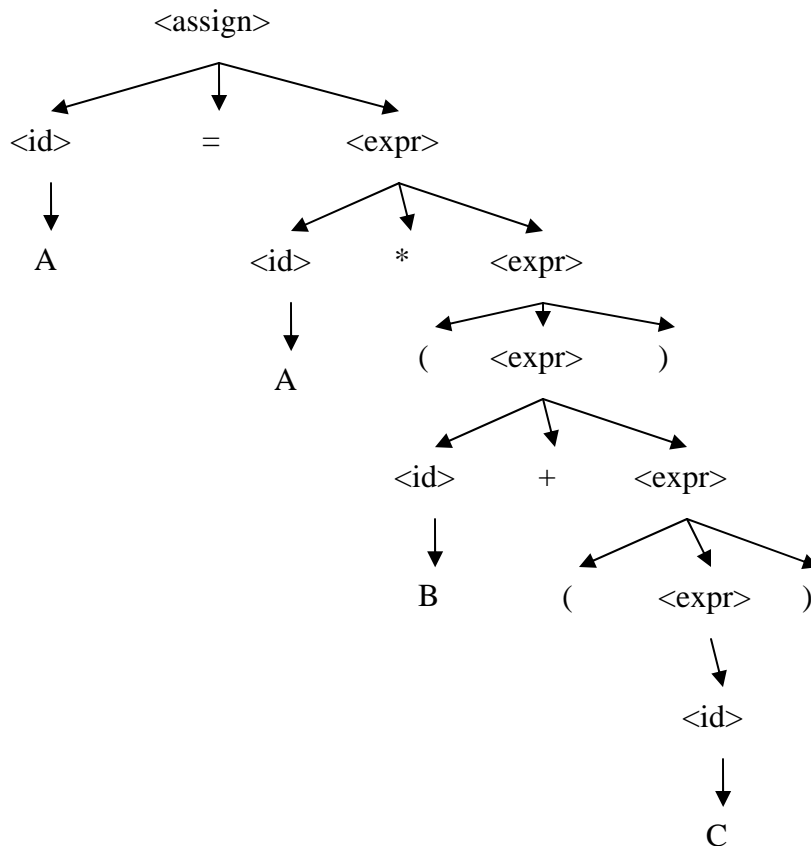
$\Rightarrow B = C * (A * C + \langle \text{id} \rangle)$

$\Rightarrow B = C * (A * C + B)$



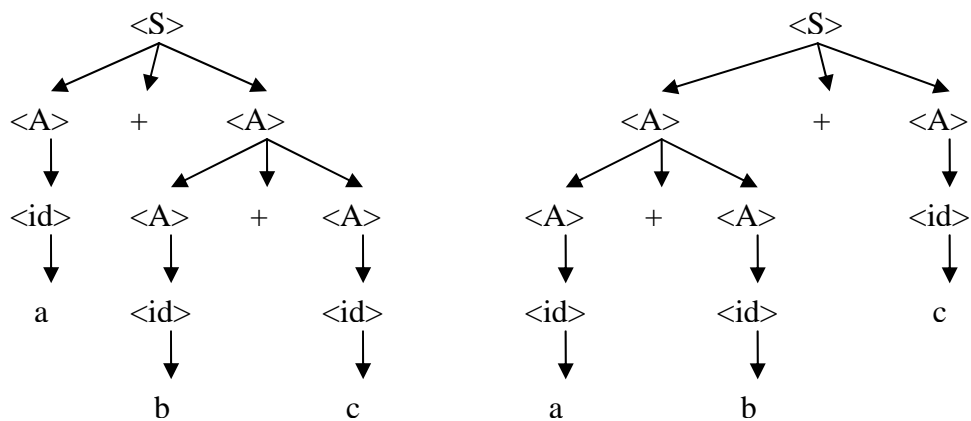
(c)

$\langle \text{assign} \rangle \Rightarrow \langle \text{id} \rangle = \langle \text{expr} \rangle$
 $\Rightarrow A = \langle \text{id} \rangle * \langle \text{expr} \rangle$
 $\Rightarrow A = A * \langle \text{expr} \rangle$
 $\Rightarrow A = A * (\langle \text{expr} \rangle)$
 $\Rightarrow A = A * (\langle \text{id} \rangle + \langle \text{expr} \rangle)$
 $\Rightarrow A = A * (B + \langle \text{expr} \rangle)$
 $\Rightarrow A = A * (B + (\langle \text{expr} \rangle))$
 $\Rightarrow A = A * (B + (\langle \text{id} \rangle))$
 $\Rightarrow A = A * (B + (C))$



3.

Given “abc”, two parsing trees are legal



or

So, the grammar is ambiguous.

4.

$\langle \text{assign} \rangle \rightarrow \langle \text{id} \rangle = \langle \text{expr} \rangle$

$\langle \text{id} \rangle \rightarrow A \mid B \mid C$

$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle + \langle \text{term} \rangle \mid \langle \text{term} \rangle$

$\langle \text{term} \rangle \rightarrow \langle \text{term} \rangle * \langle \text{factor} \rangle \mid \langle \text{factor} \rangle$

$\langle \text{factor} \rangle \rightarrow (\langle \text{expr} \rangle) \mid \langle \text{minus} \rangle$

$\langle \text{minus} \rangle \rightarrow - \langle \text{id} \rangle \mid \langle \text{id} \rangle$

5.

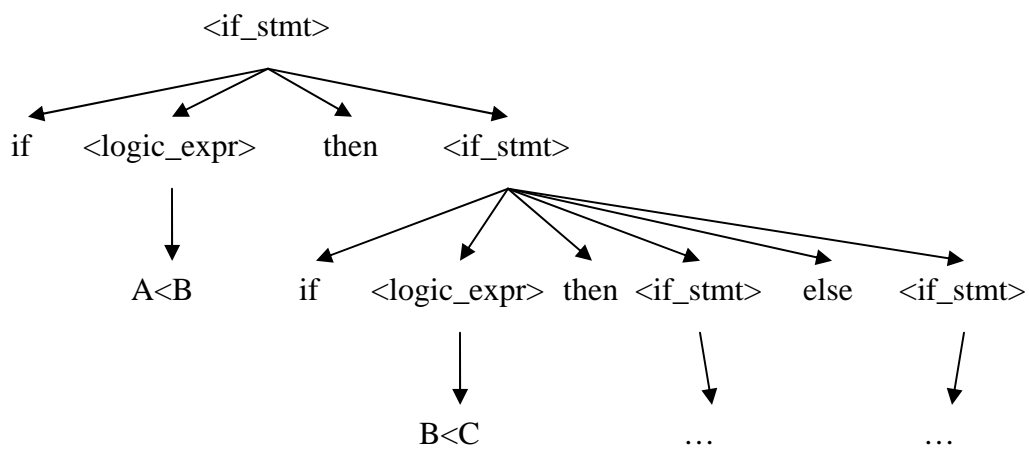
Consider following snippet

If A<B then *dosth1*

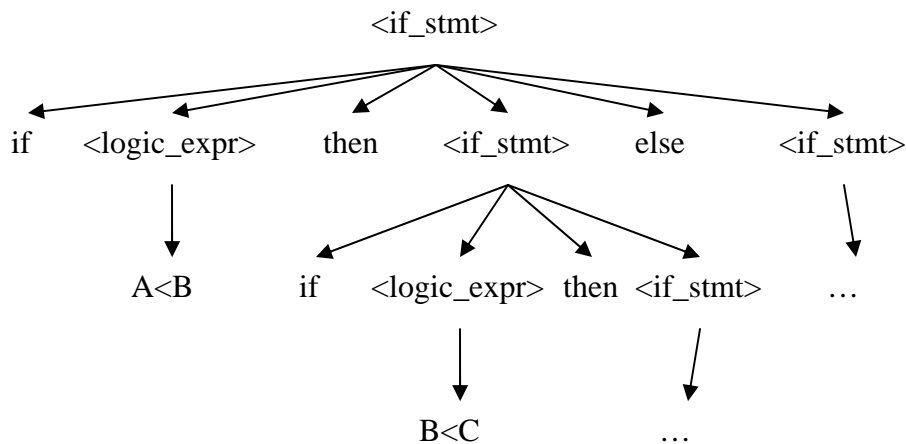
If B<C then *dosth2*

Else

Two parsing trees are legal



Or



So, the grammar is ambiguous.

Java solves the problem by separating if-then clause into two nonterminals, “if-then” and “if-then-else” (named with a postfix “short if” in Java BNF, <http://www.daimi.au.dk/dRegAut/JavaBNF.html>) while strictly applying the leftmost derivation. That ensures that the “else” terminal always matches the nearest “if-then” nonterminal. Or, in other word, arbitrarily decree that an else clause belongs to the innermost if to which it might possibly belong.

The corresponding modification is

```

<stmt> → <if_stmt> | <if_else_stmt> | <other_stmt>
<stmt_no_short_if> → <if_else_stmt_no_short_if> | <other_stmt>
<if_stmt> → if <logic_expr> then <stmt>
<if_else_stmt> → if <logic_expr> then <stmt_no_short_if> else <stmt>
<if_else_stmt_no_short_if> → if <logic_expr> then <stmt_no_short_if> else
<stmt_no_short_if>
  
```

Corresponding leftmost derivation of previous snippet

```

snippet => <stmt>
=> <if_stmt>
=> if <logic_expr> then <stmt>
=> if A<B then <stmt>
=> if A<B then if <logic_expr> then <stmt_no_short_if> else <stmt>
=> if A<B then if B<C then <stmt_no_short_if> else <stmt>
=> if A<B then if B<C then <other_stmt> else <other_stmt>
=> ...
  
```