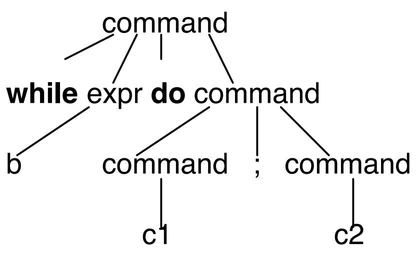
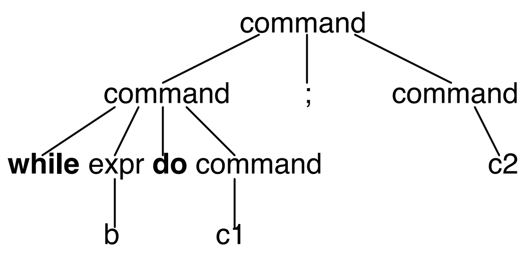
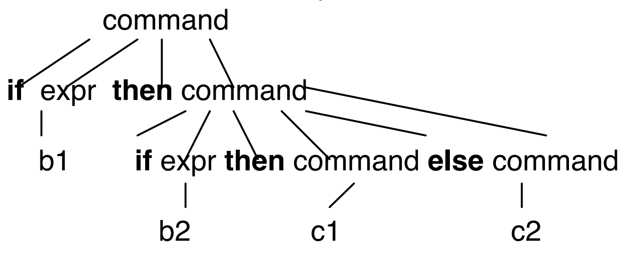
Midterm Exam
Sample Solutions

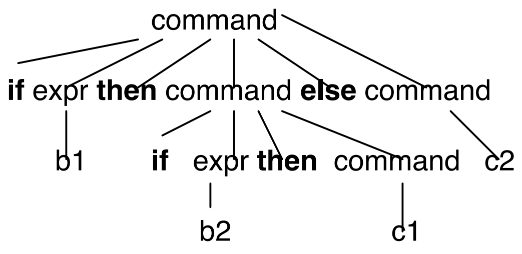
Problem 1.
Any of several instances of ambiguity could be identified as a response. For one, consider a command of the form **while** b **do** c1; c2. This has two structural descriptions.





Another example would be a command of the form **if** b1 **then if** b2 **then** c1 **else** c2 that has two structural descriptions.





Problem 2.

The first production in this attribute grammar is *replaced* by two new rules, namely

<binary numeral> ::= <binary digits>$_1$**.**<binary digits>$_2$ **e** <binary digits>$_3$, and

<binary numeral> ::= <binary digits>$_1$**.**<binary digits>$_2$ **e-** <binary digits>$_3$.

These productions provide for the addition of a binary exponent suffix. All the other productions and their attribute rules are unchanged. Note that in the examples in the problem statement, the exponent value is used as the multiplier (or dividend). This is in contrast with common usage where the multiplier has the exponent part as the *power* of the base -- this may have been distracting and was not enough simpler to justify it, sorry. Either of these choices was given full credit (and differ technically in a very small way). For consistency with the problem statement, the attribute evaluation rules provided here take the former approach (just put Val(<binary digits>$_3$) as the exponent of 2 for the latter).

Syntax rules	Attribute evaluation
<binary numeral> ::= <binary digits>$_1$**.**<binary digits>$_2$ **e**<binary digits>$_3$	Val(<binary numeral>) ← $\left(\text{Val(<binary digits>}_1\text{)} + \frac{\text{Val(<binary digits>}_2\text{)}}{2^{\text{Len(<binary digits>}_2\text{)}}}\right) * \text{Val(<binary digits>}_3\text{)}$
<binary numeral> ::= <binary digits>$_1$**.**<binary digits>$_2$ **e-**<binary digits>$_3$	Val(<binary numeral>) ← $\left(\text{Val(<binary digits>}_1\text{)} + \frac{\text{Val(<binary digits>}_2\text{)}}{2^{\text{Len(<binary digits>}_2\text{)}}}\right) / \text{Val(<binary digits>}_3\text{)}$

Since the Val(<binary digits>) attribute is correctly determined by the original attribute grammar rules, the only need is to adjust Val(<binary numeral>). Adding a positive or negative exponent requires the value of the numeral preceding the exponent to be multiplied/divided by the appropriate value. This is determined using the Val(<binary digits>) attribute, and then applied as indicated above.

Problem 3.
For the abstract syntax, we need only make the change that provides for an optional expression in declarations. In the abstract syntax we just use the pair notation $\lessdot$Identifier, Expression$\gtrdot$ for this case.

Declaration ::= **var** (Identifier | $\lessdot$Identifier, Expression$\gtrdot$)$^+$: Type;

For the semantics, since declarations now can have an effect on the store, we must change the definition of the meaning function to incorporate their effect.

meaning⟦**program** I **is** D **begin** C **end**⟧ = (execute⟦C⟧ ∘ execute⟦D⟧) emptySto

Then the definition of the execute function must be extended to apply to the declarations. Hence its signature must be changed to

execute : Command + Declaration* --> (Store --> Store).

Finally, the additional cases of the definition of the execute function must be given.

execute⟦I:T⟧ sto = sto

execute⟦$\lessdot$I, E:T$\gtrdot$⟧ sto = updateSto(sto, I, evaluate⟦E⟧ sto)

execute⟦V_1,V_2:T⟧ = execute⟦V_2:T⟧ ∘ execute⟦V_1:T⟧

execute⟦ ⟧ sto = sto

execute⟦D_1;D_2⟧ = execute⟦D_2⟧ ∘ execute⟦D_1⟧.