Problem 1

<C> ::= <A> = <C> | <A> // Assignment =, right associative

<A> ::= <B> ? <A> : <B> | <B> // Conditional ?:, right associative

<B> ::= <B> || <D> | <D> // Or ||, left associative

<D> ::= <D> && <E> | <E> // And &&, left associative

<E> ::= ! <F> | <F> // Not !, right associative

<F> ::= <V> | (<C>) | true | false // Variables, T, F, parentheses

<V> ::= x | y | z // Variables

Problem 2

<start1>.typetable := <stmt3>.typetable  
<start1>.inittable := <stmt3>.inittable  
<start2>.typetable := <stmt4>.typetable  
<start2>.inittable := <stmt4>.inittable

<stmt1>.typetable := <declare2>.typetable  
<stmt1>.inittable := <declare2>.inittable  
<stmt1>.initialized := <declare2>.initialized  
<stmt2>.typetable := <assign2>.typetable  
<stmt2>.inittable := <assign2>.inittable  
<stmt2>.initialized := <assign2>.initialized

<declare1>.typebinding := (<var>.name, <type3>.type)  
<declare1>.typetable := {<var>.name: <type3>.type}  
<declare1>.initialized := (<var>.name, false)

<type1>.type := integer  
<type2>.type := double

<assign1>.initialized := (<var>.name, true)  
<assign1>.inittable := {<var>.name: true}

<expression1>.type := if <expression4>.type = <expression5>.type then <expression4>.type else error  
<expression2>.type := <value4>.type

<value1>.type := typetable(<var>.name)  
<value1>.initialized := initialized(<var>.name)  
<value2>.type := integer  
<value3>.type := double

Problem 3A

This rule should be checked at the <assign1> non-terminal, which represents assignment statements.

* At <assign1>, check if the type of the variable matches the type of the expression.
* Test: typetable[<var>.name] = <expression3>.type
* If the types do not match, it indicates a type mismatch error in the assignment.

Problem 3B

This rule applies to any non-terminal where a variable (<var>) is used. This includes <assign1> (for the variable being assigned) and <value1> (where a variable is used in an expression).

* At <assign1> and <value1>, check if the variable has been declared.
* Test: typetable[<var>.name] ≠ error
* If the result is error, it indicates the variable is used without being declared.

Problem 3C

This rule should also be checked at <assign1> and <value1>.

* At <assign1>, when a variable is being assigned a value, update the inittable to mark it as initialized.
* At <value1>, when a variable is used in an expression, check if it has already been assigned a value.
* Test for <value1>: inittable[<var>.name] = true
* If the result is false or error, it indicates the variable is used before being assigned a value.