Exercise Sheet 1
Symbolic Logic

You do not have to answer all questions, but try to answer as many as you can.

1. Consider the following language for arithmetic expressions that contains a nullary (arity 0) operator `zero`, a unary (arity 1) operator `succ` (successor), a unary operator `pred` (predecessor), a unary operator `iszero` (is-zero check), a ternary (arity 3) infix operator `if-then-else`, a nullary operator `true`, and a nullary operator `false`.
   - Define a BNF for this language. Your BNF should contain a single rule (of the form $lhs ::= rhs_1 \mid \cdots \mid rhs_n$).
   - Indicate whether some expressions can be ambiguous.
   - Let e be an arithmetic expression. Using this grammar, write down another expression that returns zero if e is zero, and otherwise returns e's predecessor. In addition, write down the parse tree corresponding to this expression.
2. Some language also support "if-then" expressions where the infix "if-then" operator takes two arguments: a condition and a "then" branch.
   - Add an infix binary (arity 2) "if-then" operator to your language.
   - Indicate whether some expressions can be ambiguous.
   - In case some expressions are ambiguous, write down the parse trees corresponding to two different ways an ambiguous expression can be derived.
3. To use this language as part of a logical system, we can for example add an equality operator.
   - Add a new rule to your BNF for stating equalities between arithmetic expressions.
   - Define an axiom schema that states that "the expression that given an expression e, checks whether e is zero, and if it is returns zero, else returns e's predecessor" is equal to "e's predecessor", and indicate which variables are metavariables in your axiom, if any.
   - Provide 2 different instances of this axiom.