We define a small functional language – an applied, typed lambda calculus with integers and booleans as primitive types. This is a very tiny approximation to ML, so we might call it MicroML. The abstract syntax of MicroML is given by the following pseudo grammar, defining type expressions $\tau$ and ordinary expressions $e$.

$$
\tau ::= \text{int} \mid \text{bool} \mid \tau_1 \to \tau_2
$$

$$
e ::= n \mid \text{true} \mid \text{false} \mid x \mid + (e_1, e_2) \mid < (e_1, e_2) \mid \ldots \quad (\text{other possible primitive ops})
\quad \mid \text{if } e \text{ then } e_1 \text{ else } e_2 \mid e_1 \ e_2 \mid \lambda x : \tau. \ e
$$

where $n$ ranges over integer constants and $x$ ranges over a set of variables (metavariable $b$ ranges over the boolean constants true and false). In this language a value is either a number, a boolean constant, or a function expression. The metavariable $v$ ranges over values. The dynamic semantics of evaluation is given by a transition relation defined by the rules

$$
\frac{(m = n_1 + n_2)}{+(n_1, n_2) \mapsto m} \quad (D1) \quad \frac{(b = (n_1 = n_2))}{=(n_1, n_2) \mapsto b} \quad (D2)
$$

$$
\frac{e_1 \mapsto e_1'}{+(e_1, e_2) \mapsto + (e_1', e_2)} \quad (D3) \quad \frac{e_2 \mapsto e_2'}{+(v_1, e_2) \mapsto + (v_1, e_2')} \quad (D4)
$$

$$
\frac{\text{if true then } e_1 \text{ else } e_2 \mapsto e_1}{(D5)} \quad \frac{\text{if false then } e_1 \text{ else } e_2 \mapsto e_2}{(D6)}
$$

$$
\frac{e \mapsto e'}{\text{if } e \text{ then } e_1 \text{ else } e_2 \mapsto \text{if } e' \text{ then } e_1 \text{ else } e_2} \quad (D7)
$$

$$
\frac{e_1 \mapsto e_1'}{e_1 \ e_2 \mapsto e_1' \ e_2} \quad (D8) \quad \frac{e_2 \mapsto e_2'}{v_1 \ e_2 \mapsto v_1 \ e_2'} \quad (D9)
$$

$$
\frac{e \mapsto e'}{\frac{(\lambda x : \tau. \ e)v \mapsto \{v/x\}e}{(D10)}}
$$
Here we presented representative rules for the + arithmetic operator and the = relational operator. Similar rules would cover other operators like * (multiplication) and <. These semantic rules specify a call by value regime for function applications, where the argument is evaluated before the function is applied.

An expression that is not a value and for which no transition is derivable by these rules is said to be stuck. Stuck expressions represent the dynamic detection of a type error.

The typing rules for MicroML are given below.

\[
\frac{(\Gamma(x) = \tau)}{\Gamma \vdash x : \tau} \quad (S1)
\]

\[
\Gamma \vdash n : \text{int} \quad (S2)
\]

\[
\Gamma \vdash b : \text{bool} \quad (S3)
\]

\[
\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash (e_1, e_2) : \text{int}} \quad (S4)
\]

\[
\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash (e_1, e_2) : \text{bool}} \quad (S5)
\]

\[
\frac{\Gamma \vdash e : \text{bool} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 : \tau} \quad (S6)
\]

\[
\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 \ e_2 : \tau} \quad (S7)
\]

\[
\frac{\Gamma \vdash x : \tau' \vdash e : \tau}{\Gamma \vdash \lambda x : \tau. e : \tau' \rightarrow \tau} \quad (S8)
\]

The following Theorems express the relation between between typing judgements and evaluation. Together these prove type soundness, i.e. that the evaluation of any well typed expression will not reach a stuck state.

**Theorem** [Preservation]: If \( \vdash e : \tau \) and \( e \mapsto e' \) then \( \vdash e' : \tau \).

**Theorem** [Progress]: If \( \vdash e : \tau \) then either \( e \) is a value, or there exists an \( e' \) such that \( e \mapsto e' \).

**Problem 1:** Complete the proofs of the Preservation and Progress theorems that were started in class. Recall that the proof of Preservation is by induction on the rules deriving \( e \mapsto e' \), and we considered the cases for rules \( D10 \). The proof of Progress is by induction on the rules deriving \( \vdash e : \tau \), and we looked at the cases for rules \( S2 \) and \( S4 \) in class.

**Problem 2:** One of the simpler extensions of MicroML is to add expressions for pairs and corresponding binary product types:

\[
\tau ::= \ldots \tau_1 \times \tau_2 \n\]

\[
e ::= \ldots \mid (e_1, e_2) \mid \text{fst}(e) \mid \text{snd}(e)
\]
with new typing rules

\[
\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2} \quad (S9)
\]

\[
\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \text{fst}(e) : \tau_1} \quad (S10) \quad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \text{snd}(e) : \tau_2} \quad (S11)
\]

Specify what new values terms are introduced for pairs, and give evaluation rules for the pair constructs (i.e. pairing, and the projections \text{fst} and \text{snd}).

For a bonus, give the proof cases for Preservation and Progress that deal with the pairing constructs.