1. (15 points) Give a well-typed term whose evaluation all the way to a value (beginning in the empty store) will produce the following store when evaluation terminates.

\[ \mu = (l_1 \mapsto 5, \quad l_2 \mapsto l_1) \]

(Hint: you can use “let x = ... in ... ”) . (5 points)

As `let` is a derived form (syntactic sugar), you can also represent the above term using lambda-calculus: (5 points)

What’s the store typing \( \Sigma \) (such that \( \emptyset \mid \Sigma \vdash \mu \))? (5 points)

2. (22 points) Given the above store \( \mu \) and store typing \( \Sigma \), now consider the following term \( t \):

\[ \text{ref (ref 0) := } !l_2 \]

(a) What is the type of this term (i.e., \( \emptyset \mid \Sigma \vdash t : T \))? (3 points)

(b) Give a typing derivation (i.e., draw the derivation tree). (7 points)
(c) Evaluate the above term step by step all the way to a value; in each step, fill in the store, store typing, and the computation rule used (note there is exactly one computation rule in each step).
(12 points)

<table>
<thead>
<tr>
<th>step</th>
<th>term</th>
<th>store</th>
<th>comp. rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ref (ref 0) := l₂</td>
<td>(l₁ ↦ 5, l₂ ↦ l₁)</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3. (10 points) Is there a well-typed term whose evaluation (beginning in the empty store) will produce the following store when evaluation terminates?

\[
\mu = (l₁ ↦ l₂, \\
l₂ ↦ l₃, \\
l₃ ↦ l₁)
\]

If so, give it. If not, explain briefly why no such term exists.
4. (26 points) Consider the following term:

\[
\text{case } <l_2=\text{ref } 0> \text{ as } T \text{ of } <l_1=x_1> \Rightarrow x_1 \\
<l_2=x_2> \Rightarrow !x_2
\]

(a) What T makes this term typecheck? Fill in the blanks: \( T = <l_1: \quad, \quad l_2: \quad >. \) (6 points)

(b) What is the type of this term (given \( \mu = \emptyset \) and \( \Sigma = \emptyset \))? (3 points)

(c) Give the typing derivation (i.e., draw the derivation tree). (7 points)

(d) Evaluate this term all the way to a value. Distinguish between labels (\( l_i \)) and locations (\( l_i \)). (10 points)

<table>
<thead>
<tr>
<th>step</th>
<th>term</th>
<th>store</th>
<th>comp. rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>case ( l_2=\text{ref } 0 ) as ( T ) of ( l_1=x_1 ) ⇒ ( x_1 ) ( &lt;l_2=x_2&gt; \Rightarrow !x_2 )</td>
<td>( \emptyset )</td>
<td>N/A</td>
</tr>
</tbody>
</table>
5. (15 points) Define the big-step semantics evaluation rule for case terms:

\[
\text{case } t \text{ of } \langle l_i = x_i \rangle \Rightarrow t_i \; i \in 1 \ldots n
\]

Do not forget the store (i.e., instead of \( t \Downarrow v \), use \( t|\mu \Downarrow v|\mu' \) to represent the evaluation sequence \( t|\mu \rightarrow t_1|\mu_1 \rightarrow \ldots \rightarrow v|\mu' \)).

6. (12 points) Forget about references and stores for now. We know that (preservation theorem says) \( t \rightarrow t' \) and \( \emptyset \vdash t : T \) implies \( \emptyset \vdash t' : T \). But is it also true that \( t \rightarrow t' \) and \( \emptyset \vdash t' : T \) implies \( \emptyset \vdash t : T \)? If so, explain; otherwise, give an example. You can only use the syntax defined in the companion sheet (excluding reference creation, dereference, assignment, and store location).
Simply-typed lambda calculus with variants, references, Unit and Nat

**Syntax**

\[
t ::= \\
\text{unit} \\
x \\
\lambda x : T . t \\
t t \\
\text{ref } t \\
!t \\
t := t \\
l \\
o \\
succ t \\
\langle l = t \rangle \text{ as } T \\
\text{case } t \text{ of } \langle l_i = x_i \rangle \Rightarrow t_i & i \in 1 \ldots n
\]

\[
v ::= \\
\text{unit} \\
\lambda x : T . t \\
l \\
v \\
\langle l = v \rangle \text{ as } T
\]

\[
T ::= \\
\text{Unit} \\
T \rightarrow T \\
\text{Ref } T \\
\text{Nat} \\
\langle l_i : T_i \rangle & i \in 1 \ldots n
\]

\[
\mu ::= \emptyset \mid \mu, l \mapsto v
\]

\[
\Gamma ::= \emptyset \mid \Gamma, x : T
\]

\[
\Sigma ::= \emptyset \mid \Sigma, l : T
\]

\[
v ::= 0 \mid \text{succ } nv
\]

\[
\text{Evaluation}
\]

\[
\frac{t_1 \mu \rightarrow t_1' \mu'}{t_1 t_2 \mu \rightarrow t_1' t_2' \mu'} \quad (E-\text{App1})
\]

\[
\frac{t_2 \mu \rightarrow t_2' \mu'}{v_1 t_2 \mu \rightarrow v_1 t_2' \mu'} \quad (E-\text{App2})
\]

\[
(\lambda x : T_{t_1} \cdot t_{t_2}) v_2 \mu \rightarrow [x \mapsto v_2] t_{t_1} \mu \quad (E-\text{AppAbs})
\]

\[
\frac{\text{l } \not\in \text{ dom}(\mu)}{\text{ref } v_1 \mu \rightarrow l[(\mu, l \mapsto v_1)]} \quad (E-\text{RefV})
\]

\[
\frac{t_1 \mu \rightarrow t_1' \mu'}{\text{ref } t_1 \mu \rightarrow \text{ref } t_1' \mu'} \quad (E-\text{Ref})
\]

\[
\frac{\mu(l) = v}{!l \mu \rightarrow v \mu} \quad (E-\text{DerefLoc})
\]
\[ t_1 | \mu \rightarrow t_1' | \mu' \]  (E-Deref)

\[ l := v_2 | \mu \rightarrow \text{unit} | [l \mapsto v_2] | \mu \]  (E-Assign)

\[ t_1 | \mu \rightarrow t_1' | \mu' \]  (E-Assign1)

\[ t_2 | \mu \rightarrow t_2' | \mu' \]  (E-Assign2)

\[ \text{succ} \ t_1 | \mu \rightarrow \text{succ} \ t_1' | \mu' \]  (E-Succ)

\[ \text{case} \ <l_j := v_j > \text{ as } T \rightarrow \text{case } \ t_i \ i \in 1..n \rightarrow [x_j \mapsto v_j] \ t_j \]  (E-CaseVariant)

\[ t_0 \rightarrow t_0' \]  (E-Case)

\[ \text{case } t_0 \text{ of } <l_i := x_i > \rightarrow \text{case } t_0' \text{ of } <l_i := x_i > \rightarrow \text{case } t_i \ i \in 1..n \rightarrow \text{case } t_i' \ i \in 1..n \]  (E-Case)

\[ \text{case } <l_i := t_i > \text{ as } T \rightarrow \text{case } <l_i := t_i' > \text{ as } T \]  (E-Assign)

\[ \Gamma | \Sigma \vdash \text{unit} : \text{Unit} \]  (T-Unit)

\[ x : T \in \Gamma \]
\[ \Gamma | \Sigma \vdash x : T \]  (T-Var)

\[ \Gamma, x : T_1 | \Sigma \vdash t_2 : T_2 \]
\[ \Gamma | \Sigma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2 \]  (T-Abs)

\[ \Gamma | \Sigma \vdash t_1 : T_{11} \rightarrow T_{12} \]
\[ \Gamma | \Sigma \vdash t_2 : T_{11} \]
\[ \Gamma | \Sigma \vdash t_1 \ t_2 : T_{12} \]  (T-App)

\[ \Sigma (l) = T_1 \]
\[ \Gamma | \Sigma \vdash l : \text{Ref } T_1 \]  (T-Loc)

\[ \Gamma | \Sigma \vdash t_1 : T_1 \]
\[ \Gamma | \Sigma \vdash \text{ref } t_1 : \text{Ref } T_1 \]  (T-Ref)

\[ \Gamma | \Sigma \vdash t_1 : \text{Ref } T_{11} \]
\[ \Gamma | \Sigma \vdash \text{!t}_1 : T_{11} \]  (T-Deref)

\[ \Gamma | \Sigma \vdash t_1 : \text{Ref } T_{11} \]
\[ \Gamma | \Sigma \vdash t_2 : T_{11} \]
\[ \Gamma | \Sigma \vdash \text{t}_1 := \text{t}_2 : \text{Unit} \]  (T-Assign)

\[ \Gamma | \Sigma \vdash t_1 : \text{Nat} \]
\[ \Gamma | \Sigma \vdash \text{succ } \text{t}_1 : \text{Nat} \]  (T-Succ)

\[ \Gamma \vdash t_j : T_j \]
\[ \Gamma | \Sigma \vdash <l_j := t_j > \text{ as } <l_j : T_j \ i \in 1..n > \rightarrow <l_j : T_j \ i \in 1..n > \]  (T-Variant)

\[ \Gamma \vdash t_0 : <l_i : T_i \ i \in 1..n > \]
\[ \text{for each } i \]
\[ \Gamma, x_i : T_i \vdash t_i : T \]
\[ \Gamma \vdash \text{case } t_0 \text{ of } <l_i := x_i > \rightarrow t_i \ i \in 1..n : T \]  (T-Case)