Homework 3

Student Number:
Name:

Problem 1. Consider the following program which is written in C syntax.

```c
int x = 1;
int f() { return x; }
int main() {
    int x = 2;
    return f();
}
```

(a) What is the return value of `main()` when it uses static scoping? dynamic scoping?

(b) Try to write another program whose execution result will be affected by static/dynamic scoping in a way that is essentially different from the one illustrated above, or explain why you cannot write such a program.

Problem 2. Extend tuples to records, and write the (a) syntax and (b) semantic rules for records. Example usage:

- Elements are indexed by labels:
  - `{y = 10}`
  - `{id = 1, salary = 50000, active = true}`

- The order of the record fields is insignificant:
  - `{y = 10, x = 5}` is the same as `{x = 5, y = 10}`

- To access fields of a record:
  - `a.id`
  - `b.salary`

Problem 3. Consider adding trees with leaf node elements of type `t` (a “t tree”) to the extended simply-typed language we have defined up to Lecture 6. The additional types, expressions, and values are as follows:

```
t ::= ... (as already defined) ... | t tree
e ::= ... (as already defined) ... | Lf(e) | Br(e_1, e_2) | case e of (Lf(x) ⇒ e_1 | Br(x, y) ⇒ e_2)
v ::= ... (as already defined) ... | Lf(v) | Br(v_1, v_2)
```
Here are the operational rules:

\[
\begin{align*}
  &e \to e' \\
  &Lf(e) \to Lf(e') \quad (L) \\
  &e_1 \to e_1' \\
  &Br(e_1, e_2) \to Br(e_1', e_2) \quad (B1) \\
  &e_2 \to e_2' \\
  &Br(v_1, e_2) \to Br(v_1, e_2') \quad (B2) \\
  &\text{case } e \text{ of } (Lf(x) \Rightarrow e_1 \mid Br(x, y) \Rightarrow e_2) \to \text{case } e' \text{ of } (Lf(x) \Rightarrow e_1 \mid Br(x, y) \Rightarrow e_2) \quad (C1) \\
  &\text{case } Lf(v) \text{ of } (Lf(x) \Rightarrow e_1 \mid Br(x, y) \Rightarrow e_2) \to e_1[v/x] \quad (C2) \\
  &\text{case } Br(v_1, v_2) \text{ of } (Lf(x) \Rightarrow e_1 \mid Br(x, y) \Rightarrow e_2) \to e_2[v_1/x][v_2/y] \quad (C3)
\end{align*}
\]

(a) Give the typing rules for Lf, Br and case expressions.
(b) Define a function \textit{height} : t \ tree \to \textit{int}, which returns the height of a given tree, using the fix-point combinator.

**Problem 4.** Using the environment model for lambda calculus with let, write the detailed multi-step evaluation steps of the following \( \lambda \) expression. (Please define closures carefully just like in the lecture slides.)

\[
\begin{align*}
  &\text{let } x = 1 \text{ in} \\
  &\quad \text{let } y = 3 \text{ in} \\
  &\quad \quad \text{let add = } \lambda x. \lambda y. x+y \text{ in} \\
  &\quad \quad \text{let sub = } \lambda x. \lambda y. x-y \text{ in} \\
  &\quad \quad \text{sub (add x y) (sub y x)}
\end{align*}
\]