Due: 2020/4/5

Homework 5 - Extend

∗ If there is any problem, please contact TA.

Name:_____ Student ID:_____ Email: _____

Problem 1. (30 points) Consider the following program which is written in C syntax.

```
int x = 1;

void f1() {
int x = 2;
f2();
printf(x)
}

void f2() {
int x = 3;
printf(x)
}

int main() {
f1();
printf(x)
}
```

(a) What will be printed after running `main()` when it uses static scoping? dynamic scoping?

Solution.
static: 3 2 1
dynamic: 3 2 1 □

Problem 2. (40 points) Extend tuples to records, and write the (a) syntax and (b) semantic rules for records. Example usage:

- Elements are indexed by labels:
  - $\{y = 10\}$
  - $\{id = 1, salary = 50000, active = \textbf{true}\}$
- 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:
  - $a.id$
  - $b.salary$

Solution.

(a) Syntax: (x and x_i are names)

$$\begin{aligned} e &::= \ldots \mid \{x_1 = e_1, \ldots, x_n = e_n\} \mid e.x \\ v &::= \ldots \mid \{x_1 = v_1, \ldots, x_n = v_n\} \\ t &::= \ldots \mid \{(x_1, t_1), \ldots, (x_n, t_n)\} \end{aligned}$$

(b) Semantics:

$$\frac{e_i \to e_i'}{\{x_1 = v_1, \ldots, x_{i-1} = v_{i-1}, x_i = e_i, \ldots\} \to \{x_1 = v_1, \ldots, x_{i-1} = v_{i-1}, x_i = e_i', \ldots\}} \quad \text{(E-record1)}$$

$$\frac{e \to e'}{e.x \to e'.x} \quad \text{(E-label1)}$$

$$\frac{}{\{x_1 = v_1, \ldots, x_n = v_n\}.x_i \to v_i} \quad \text{(E-label2)}$$

$$\frac{\Gamma \vdash e_i : t_i}{\Gamma \vdash \{x_1 = e_1, \ldots, x_n = e_n\} : \{(x_1, t_1), \ldots, (x_n, t_n)\}} \quad \text{(T-record)}$$

$$\frac{\Gamma \vdash e : \{(x_1, t_1), \ldots, (x_n, t_n)\}}{\Gamma \vdash e.x_i : t_i} \quad \text{(T-label)}$$

□

Problem 3. (30 points) Another way of defining **let** as derived form might be to desugar it by "executing" it immediately-i.e., to regard **Let x=**t_1 **in** t_2 as an abbreviation for the substituted body $t_2[t_1/x]$. Is this a good idea ?

Solution. No. It changes the order of evaluation: the rules **E-LETV** and **E-LET** specify a call-by-value order, where t_1 in $let\ x = t_1\ in\ t_2$ must be reduced to a value before we are allowed to substitute it for x and begin working on t_2.
For another thing, although the validity of the typing rule is preserved by this translation—this follows directly from the substitution lemma—the property of ill-typedness of terms is not preserved. For example, the ill-typed term:

$$let\ x\ =\ unit(unit)\ in\ unit$$

is translated to the well-typed term *unit*: since x does not appear in the body unit, the ill-typed subterm unit(unit) simply disappears. □