Syntax :

t	$::=$		*terms :*
		x	*variable*
		$\lambda x : T.t$	*abstraction*
		$t\ t$	*application*
		$let\ x = t\ in\ t$	*let binding*
		$\{l_i = t_i^{i\in 1\cdots n}\}$	*record*
		$t.l$	*projection*
		$fix\ t$	*fixed point of t*
v	$::=$		*values :*
		$\lambda x : T.t$	*abstraction value*
		$\{l_i = v_i^{i\in 1\cdots n}\}$	*record value*
T	$::=$		*types :*
		$T \to T$	*type of functions*
		$\{l_i = T_i^{i\in 1\cdots n}\}$	*type of record*
Γ	$::=$		*contexts :*
		$\varnothing$	*empty context :*
		$\Gamma, x : T$	*term variable binding*

Typing : $\qquad \Gamma \vdash t : T$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \qquad \text{(TVar)}$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1.t_2 : T_1 \to T_2} \qquad \text{(TAbs)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1 t_2 : T_{12}} \qquad \text{(TApp)}$$

$$\frac{for\ each\ i \quad \Gamma \vdash t_i : T_i}{\Gamma \vdash \{l_i = t_i^{i\in 1\cdots n}\} : \{l_i : T_i^{i\in 1\cdots n}\}} \qquad \text{(TRcd)}$$

$$\frac{\Gamma \vdash t_1 : \{l_i : T_i^{i\in 1\cdots n}\}}{\Gamma \vdash t_1.l_j : T_j} \qquad \text{(TProj)}$$

$$\frac{\Gamma \vdash t_1 : T_1 \to T_1}{\Gamma \vdash fix\ t_1 : T_1} \qquad \text{(TFix)}$$

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash let\ x = t_1\ in\ t_2 : T_2} \qquad \text{(TLet)}$$

Evaluation : $\qquad t \longrightarrow t'$

$$\frac{t_1 \longrightarrow t_1'}{t_1 t_2 \longrightarrow t_1' t_2} \qquad \text{(EApp}_1\text{)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1 t_2 \longrightarrow v_1 t_2'} \qquad \text{(EApp}_2\text{)}$$

$$(\lambda x : T_{11}.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \qquad \text{(EAppAbs)}$$

$$\{l_i = v_i^{i\in 1\cdots n}\}.l_j \longrightarrow v_j \qquad \text{(EProjRcd)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1.l \longrightarrow t_1'.l} \qquad \text{(EProj)}$$

$$\frac{t_j \longrightarrow t_j'}{\begin{array}{l}\{l_i = v_i^{i\in 1\cdots j-1}, l_j = t_j, l_k = t_k^{k\in j+1\cdots n}\}\\ \longrightarrow \{l_i = v_i^{i\in 1\cdots j-1}, l_j = t_j', l_k = t_k^{k\in j+1\cdots n}\}\end{array}} \qquad \text{(ERcd)}$$

$$let\ x = v_1\ in\ t_2 \longrightarrow [x \mapsto v_1]t_2 \qquad \text{(ELetV)}$$

$$\frac{t_1 \longrightarrow t_1'}{let\ x = t_1\ in\ t_2 \longrightarrow let\ x\ \ = t_1'\ in\ t_2} \qquad \text{(ELet)}$$

$$fix(\lambda x : T_1.t_2) \longrightarrow [x \mapsto (fix(\lambda x : T_1.t_2))]t_2 \qquad \text{(EFixBeta)}$$

$$\frac{t_1 \longrightarrow t_1'}{fix\ t_1 \longrightarrow fix\ t_1'} \qquad \text{(EFix)}$$

Derived Forms :

$$letrec\ x : T_1 = t_1\ in\ t_2 \overset{\text{def}}{=} let\ x = fix(\lambda x : T_1.t1)\ in\ t_2$$

Cuadro 1: Simply typed lambda-calculus with Let binding, Records and General recursion.

LEMMA [Inversion of the Typing Relation]:

1. If $\Gamma \vdash x : R$, then $x : R \in \Gamma$
2. If $\Gamma \vdash \lambda x : T_1.t_2 : R$, then $R = T_1 \to R_2$ for some R_2, with $\Gamma,\, x : T_1 \vdash t_2 : R_2$.
3. If $\Gamma \vdash t_1\ t_2 : R$, then there is some type T_{11} such that $\Gamma \vdash t_1 : T_{11} \to R$ and $\Gamma \vdash t_2 : T_{11}$.
4. If $\Gamma \vdash let\ x = t_1\ in\ t_2 : R$, then there is some type T_1 such that $\Gamma \vdash t_1 : T_1$, with $\Gamma,\, x : T_1 \vdash t_2 : R$.
5. If $\Gamma \vdash \{l_i = {t_i}^{i \in 1 \cdots n}\} : R$, then there are some types ${R_i}^{i \in 1 \cdots n}$ such that for each i is satisfied that $\Gamma \vdash t_i : R_i$ and $R = \{l_i = {R_i}^{i \in 1 \cdots n}\}$.
6. If $\Gamma \vdash t.l_j : R$, then there is some type $\{l_i = {R_i}^{i \in 1 \cdots n}\}$ such that $\Gamma \vdash t : \{l_i = {R_i}^{i \in 1 \cdots n}\}$ and $R = R_j$.
7. $\Gamma \vdash fix\ t_1 : R$ then $\Gamma \vdash t1 : R \to R$.

Proof : Immediate from the definition of the typing relation.

LEMMA [Canonical Forms]:

1. If v is a value of type $T_1 \to T_2$, then $v = \lambda x : T_1.t_2$.
2. If v is a value of type $\{l_i = {T_i}^{i \in 1 \cdots n}\}$ then $v = \{l_i = {v_i}^{i \in 1 \cdots n}\}$, and $\Gamma \vdash v_i : T_i\ i \in 1 \cdots n$.

Proof : Straightforward.

THEOREM [Progress]: Suppose t is a closed, well-typed term (that is, $\vdash\ t' : T$ for some T). Then either t is a value or else there is some t' with $t \longrightarrow t'$. Proof: Straightforward induction on typing derivations.

1. The variable case cannot occur (because t is closed).
2. The abstraction case is immediate, since abstractions are values.
3. For application, where $t = t_1\ t_2$, with $\vdash\ t_1 : T_{11} \longrightarrow T_{12}$ and $\vdash\ t_2 : T_{11}$, for the inversion lemma. Then, by the induction hypothesis, either t_1 is a value or else it can make a step of evaluation, and likewise t_2. If t_1 can take a step, then rule $EApp1$ applies to t. If t_1 is a value and t_2 can take a step, then rule $EApp2$ applies. Finally, if both t_1 and t_2 are values, then the canonical forms lemma tells us that t_1 has the form $\lambda x : T_{11}.t_{12}$, and so rule $EAppAbs$ applies to t.
4. If $t = (let\ x = t_1\ in\ t_2)$, then $\vdash\ t_1 : T_1$ and $\vdash\ t_2 : T_2$, for the inversion lemma. Then, by the induction hypothesis, either t_1 is a value or else it can make a step of evaluation. If t_1 can take a step, then rule $ELet$ applies to t. If t_1 is a value, then rule $ELetV$ applies.
5. If $t = \{l_i = {t_i}^{i \in 1 \cdots n}\}$, then there are some types ${T_i}^{i \in 1 \cdots n}$ such that for each i is satisfied that $\Gamma \vdash t_i : T_i$ and $T = \{l_i = {T_i}^{i \in 1 \cdots n}\}$. By the induction hypothesis, for each ${t_i}^{i \in 1 \cdots n}$, either it is a value or else it can make a step of evaluation. If all the ${t_i}^{i \in 1 \cdots n}$ are values, then t is a value. If all the ${t_i}^{i \in 1 \cdots n}$ are not values, then there is t_j such that it can take a step, then rule $ERcd$ applies to t.

6. If $t = t'.l_j$, then there is some type $\{l_i = T_i^{\,i\in 1\cdots n}\}$ such that $\Gamma \vdash t' : \{l_i = T_i^{\,i\in 1\cdots n}\}$ and $T = T_j$. By the induction hypothesis, either t' is a value or else it can make a step of evaluation. If t' can take a step, then rule $EProj$ applies to t. If t' is a value, then rule $EProjRcd$ applies.

7. If $t = fix\ t'$, then $\Gamma \vdash t' : T \rightarrow T$, for the inversion lemma. By the induction hypothesis, either t' is a value or else it can make a step of evaluation. If t' can take a step, then rule $EFix$ applies to t. If t' is a value, then the canonical forms lemma tells us that t' has the form $\lambda x : T.t_1$, and so rule $EFixBeta$ applies to t.

LEMMA[Permutation]: If $\Gamma \vdash t : T$ and Δ is a permutation of Γ, then $\Delta \vdash\ t : T$.
Proof: Straightforward induction on typing derivations.

LEMMA[Weakening]: If $\Gamma \vdash t : T$ and $x \notin dom(\Gamma)$, then $\Gamma, x : S \vdash\ t : T$.
Proof: Straightforward induction on typing derivations.

LEMMA [Preservation of types under substitution]: If $\Gamma, x : S\ \vdash\ t : T$ and $\Gamma \vdash s : S$, then $\Gamma \vdash [x \mapsto s]t : T$.
Proof: By induction on a derivation of the statement $\Gamma, x : S\ \vdash\ t : T$. For a given derivation, we proceed by cases on the final typing rule used in the proof.

1. *Case* $TVar : t = z$ with $z : T \in (\Gamma, x : S)$. There are two sub-cases to consider, depending on whether z is x or another variable. If $z = x$, then $[x \mapsto s]z = s$. The required result is then $\Gamma \vdash s : S$, which is among the assumptions of the lemma. Otherwise, $[x \mapsto s]z = z$, and the desired result is immediate.

2. *Case* $TAbs$: $t = \lambda y : T_2.t_1, with\ T = T_2 \rightarrow T_1$ and $\Gamma, x : S, y : T_2\ \vdash\ t_1 : T_1$. By convention, we may assume $x \neq y$ and $y \notin FV(s)$. Using permutation on the given subderivation, we obtain $\Gamma, y : T_2, x : S\ \vdash\ t_1 : T_1$. Using weakening on the other given derivation ($\Gamma \vdash s : S$), we obtain $\Gamma, y : T_2\ \vdash\ s : S$. Now, by the induction hypothesis, $\Gamma, y : T_2\ \vdash\ [x \mapsto s]t_1 : T_1$. By $TAbs, \Gamma \vdash \lambda y : T_2.[x \mapsto s]t_1 : T_2 \rightarrow T_1$. But this is precisely the needed result,since, by the definition of substitution, $[x \mapsto s]t = \lambda y : T_1.[x \mapsto s]t_1$.

3. *Case* $TApp$: $t = t_1\ t_2, \Gamma, x : S\ \vdash\ t_1 : T_2 \rightarrow T_1, \Gamma, x : S\ \vdash\ t_2 : T_2, T = T_1$. By the induction hypothesis, $\Gamma \vdash [x \mapsto s]t_1 : T_2 \rightarrow T_1$ and $\Gamma \vdash [x \mapsto s]t_2 : T_2$. By $TApp$, $\Gamma \vdash [x \mapsto s]t_1\ [x \mapsto s]t_2 : T$, then $\Gamma \vdash [x \mapsto s](t_1\ t_2) : T$.

4. *Case* $TRcd$: $t = \{l_i = t_i^{\,i\in 1\cdots n}\}$, for each $i, \Gamma, x : S\ \vdash\ t_i : T_i$, $T = \{l_i = T_i^{\,i\in 1\cdots n}\}$. By the induction hypothesis, for each $i, \Gamma \vdash [x \mapsto s]t_i : T_i$. By the $TRcd$, $\Gamma \vdash \{l_i = [x \mapsto s]t_i^{\,i\in 1\cdots n}\} : T$, then $\Gamma \vdash [x \mapsto s]\{l_i = t_i^{\,i\in 1\cdots n}\} : T$.

5. *Case* $TLet$: $t = (let\ y = t_1\ in\ t_2), \Gamma, x : S\ \vdash\ t_1 : T_1, \Gamma, x : S,\ y : T_1\ \vdash\ t_2 : T_2,\ T = T_2$. By convention, we may assume $x \neq y$ and $y \notin FV(s)$. Using permutation on the given subderivation, we obtain $\Gamma, y : T_1, x : S\ \vdash\ t_2 : T_2$. Using weakening on the other given derivation ($\Gamma \vdash s : S$), we obtain $\Gamma, y : T_1\ \vdash\ s : S$. Now, by the induction hypothesis, $\Gamma, y : T_1\ \vdash\ [x \mapsto s]t_2 : T_2$ and $\Gamma\ \vdash\ [x \mapsto s]t_1 : T_1$. By $TLet, \Gamma \vdash let\ y : [x \mapsto s]t_1\ in\ [x \mapsto s]t_2 : T_2$. But this is precisely the needed result,since, $[x \mapsto s]t = (let\ y = [x \mapsto s]t_1\ in\ [x \mapsto s]t_2)$.

6. *Case TProj*: $t = t_1.l_j,\ \Gamma, x : S \vdash\ \ t_1 : \{l_i = T_i^{\,i\in 1\cdots n}\}$ and $T = T_j$. By the induction hypothesis, $\Gamma \vdash [x \mapsto s]t_1 : \{l_i = T_i^{\,i\in 1\cdots n}\}$. By $TProj$, $\Gamma \vdash [x \mapsto s]t_1.l_j : T$. But this is precisely the needed result,since, $[x \mapsto s]t = [x \mapsto s]t_1.l_j$.

7. *Case TFix*: $t = fix\ t_1$ and $\Gamma, x : S \vdash t_1 : T \rightarrow T$. By the induction hypothesis, $\Gamma \vdash [x \mapsto s]t_1 : T \rightarrow T$. By $TFix$, $\Gamma \vdash fix\ [x \mapsto s]t_1 : T$. But this is precisely the needed result,since, $[x \mapsto s]t = fix\ [x \mapsto s]t_1$.

LEMMA[Preservation]: If $\Gamma \vdash t : T$ and $t \longrightarrow t'$, then $\Gamma \vdash t' : T$.
Proof: By induction on a derivation of $t : T$. At each step of the induction, we assume that the desired property holds for all subderivations (i.e., that if $s : S$ and $s \longrightarrow s'$, then $s' : S$, whenever s : S is proved by a subderivation of the present one) and proceed by case analysis on the final rule in the derivation.

1. *Case TVar*: It cannot be the case that $t \longrightarrow t'$ for any t', and the requirements of the theorem are vacuously satisfied.

2. *Case TAbs*: $t = \lambda x : T_2.t_1, with\ T = T_2 \rightarrow T_1$ and $\Gamma, x : T_2\ \ \vdash\ \ t_1 : T_1$. If the last rule in the derivation is $TAbs$, then we know from the form of this rule that t must be a function $\lambda x : T_2.t_1$ and T must be $T_2 \rightarrow T_1$, with $\Gamma, x : T_2\ \ \vdash\ \ t_1 : T_1$. But then t is a value, so it cannot be the case that $t \longrightarrow t'$ for any t', and the requirements of the theorem are vacuously satisfied.

3. *Case TApp*: $t = t_1\ t_2,\ \vdash\ \ t_1 : T_2 \rightarrow T$ and $\vdash\ \ t_2 : T_2$. If the last rule in the derivation is $TApp$, then we know from the form of this rule that t must have the form $t_1\ t_2$, for some t_1 and t_2. We must also have a subderivation with conclusions $\vdash t_1 : T_2 \rightarrow T$ and $\vdash t_2 : T_2$. Now, looking at the evaluation rules with this form on the left-hand side, we find that there are three rules by which $t \longrightarrow t'$ can be derived: $EApp1$, $EApp2$ and $EAppAbs$. We consider each case separately.

   - *Subcase EApp1*: $t_1 \longrightarrow t'_1,\ t' = t'_1\ t_2$. By the induction hypothesis, $\vdash t'_1 : T_2 \rightarrow T$, then we can apply rule $TApp$, to conclude that $\vdash t'_1\ t_2 : T$, that is $\vdash t' : T$.
   - *Subcase EApp2*: $t_2 \longrightarrow t'_2,\ t' = t_1\ t'_2$. By the induction hypothesis, $\vdash t'_2 : T_2$, then we can apply rule $TApp$, to conclude that $\vdash t_1\ t'_2 : T$, that is $\vdash t' : T$.
   - *Subcase EAppAbs*: $t_1 = \lambda x : T_1.t_{12},\ t_2 = v_2,\ t' = [x \mapsto v_2]t_{12}$ and $\vdash t_{12} : T$ for the inversion lemma. The resulting term $\vdash [x \mapsto v_2]t_{12} : T$, for the Substitution lemma, that is $\vdash t' : T$.

4. *Case TRcd*: $t = \{l_i = t_i^{\,i\in 1\cdots n}\}$, for each $i, \vdash\ \ t_i : T_i$, $T = \{l_i = T_i^{\,i\in 1\cdots n}\}$.

   - If for each i, $t_i = v_i$, then t is a value, so it cannot be the case that $t \longrightarrow t'$ for any t', and the requirements of the theorem are vacuously satisfied.
   - *Subcase ERcd*: $t = \{l_i = v_i^{\,i\in 1\cdots j-1}, l_j = t_j, l_k = t_k^{\,k\in j+1\cdots n}\}$, $t_j \longrightarrow t'_j$, $t' = \{l_i = v_i^{\,i\in 1\cdots j-1}, l_j = t'_j, l_k = t_k^{\,k\in j+1\cdots n}\}$ and $\vdash t_j : T_j$. By the induction hypothesis, $\vdash t'_j : T_j$, then we can apply rule $TRcd$, to conclude that $\vdash t' : T$.

5. *Case TLet*: $t = (let\ x = t_1\ in\ t_2), \Gamma \vdash\ \ t_1 : T_1$ and $\Gamma, x : T_1\ \ \vdash\ \ t_2 : T$. If the last rule in the derivation is $TLet$, then we know from the form of this rule that t must have the form

$(let\ x = t_1\ in\ t_2)$, for some t_1 and t_2. We must also have a subderivation with conclusions $\vdash t_1 : T_1$ and $\vdash t_2 : T_2$. Now, looking at the evaluation rules with *Let* form on the left-hand side, we find that there are two rules by which $t \longrightarrow t'$ can be derived: *ELetV* and *ELet*. We consider each case separately.

- *Subcase ELetV*: $t_1 = v_1$, $t' = [x \mapsto v_1]t_2$, and $\vdash t_2 : T$. Then for the substitution lemma $t' : T$.
- *Subcase ELet*: $t_1 \longrightarrow t'_1$ and $t' = (let\ x = t'_1\ in\ t_2)$. By the induction hypothesis, $\vdash t'_1 : T_1$, then we can apply rule *TLet*, to conclude that $\vdash (let\ x = t'_1\ in\ t_2) : T$, that is $\vdash t' : T$.

6. *Case TProj*: $t = t_1.l_j$, $\vdash\ \ t_1 : \{l_i = T_i{}^{i\in 1\cdots n}\}$ and $T = T_j$. If the last rule in the derivation is *TProj*, then we know from the form of this rule that t must have the form $t_1.l_j$. We must also have a subderivation with conclusions $\vdash t_1 : \{l_i = T_i{}^{i\in 1\cdots n}\}$ and $T = T_j$. Now, looking at the evaluation rules with this form on the left-hand side, we find that there are two rules by which $t \longrightarrow t'$ can be derived: *EProjRcd* and *EProj*. We consider each case separately.

   - *Subcase EProjRcd*: $t_1 = \{l_i = v_i{}^{i\in 1\cdots n}\}$ and $t' = v_j$.This means we are finished, since we know $\vdash v_j : T_j$ and $T = T_j$.
   - *Subcase EProj*: $t_1 \longrightarrow t'_1$ and $t' = t'_1.t_j$. By the induction hypothesis, $\vdash t'_1 : \{l_i = T_i{}^{i\in 1\cdots n}\}$, then we can apply rule *TLet*, to conclude that $\vdash t'_1.t_j : T_j$, that is $\vdash t' : T$.

7. *Case TFix*: $t = fix\ t_1$ and $\vdash t_1 : T \to T$. If the last rule in the derivation is *TFix*, then we know from the form of this rule that t must have the form $fix\ t_1$, for some t_1. We must also have a subderivation with conclusions $\vdash t_1 : T \to T$. Now, looking at the evaluation rules with fix on the left-hand side, we find that there are two rules by which $t \longrightarrow t'$ can be derived: *EFixBeta* and *EFix*. We consider each case separately.

   - *Subcase EFixBeta*: $t_1 = \lambda x : T_1.t_2,\ t' = [x \mapsto (fix\ (\lambda x : T_1.t_2))]t_2$, $\vdash t_1 : T \to T$ and $\vdash t_2 : T$, for the inversion lemma. Then, by the substitution lemma, $\vdash [x \mapsto (fix\ (\lambda x : T_1.t_2))]t_2 : T$, which is what we need.
   - *Subcase EFix*: $t_1 \longrightarrow t'_1$ and $t' = fix\ t'_1$. By the induction hypothesis, $\vdash t'_1 : T \to T$, then we can apply rule *TFix*, to conclude that $\vdash fix\ t'_1 : T$, that is $\vdash t' : T$.