Linguagem L1 com construções imperativas

Sintaxe abstrata:

$$
\begin{array}{lcl}
e & \in & \mathsf{Expr} \\
e & ::= & n \mid b \mid e_1\ op\ e_2 \mid \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \\
 & \mid & x \mid e_1\ e_2 \mid \mathsf{fn}\ x:T \Rightarrow e \mid \mathsf{let}\ x:T = e_1\ \mathsf{in}\ e_2 \mid \mathsf{let\ rec}\ f:T_1 \to T_2 = (\mathsf{fn}\ x{:}T_1 \Rightarrow e_1)\ \mathsf{in}\ e_2 \\
 & \mid & e_1 := e_2 \mid\ !\,e \mid \mathsf{new}\ e \mid \mathsf{skip} \mid \mathsf{while}\ e_1\ \mathsf{do}\ e_2 \mid e_1;\ e_2 \mid \boxed{l}
\end{array}
$$

$$
\begin{array}{lcl}
T & \in & \mathsf{Types} \\
T & ::= & \mathsf{int} \mid \mathsf{bool} \mid T_1 \to T_2 \mid T\ \mathsf{ref} \mid \mathsf{unit}
\end{array}
$$

Semântica operacional *small-step*:

$$
\begin{array}{lcl}
v & \in & \mathsf{Values} \\
v & ::= & n \mid b \mid \mathsf{fn}\ :T \Rightarrow e \mid \boxed{l}
\end{array}
$$

$$\frac{[\![n]\!] = [\![n_1+n_2]\!]}{n_1+n_2,\ \sigma \longrightarrow n,\ \sigma}\ (\text{OP}+) \qquad \frac{}{(\mathsf{fn}\ x{:}T \Rightarrow e)\ v,\ \sigma \longrightarrow \{v/x\}e,\ \sigma}\ (\text{E-}\beta)$$

$$\frac{e_1,\ \sigma \longrightarrow e_1',\ \sigma'}{e_1\ op\ e_2,\ \sigma \longrightarrow e_1'\ op\ e_2,\ \sigma'}\ (\text{OP1}) \qquad \frac{e_1,\ \sigma \longrightarrow e_1',\ \sigma'}{e_1\ e_2,\ \sigma \longrightarrow e_1'\ e_2,\ \sigma'}\ (\text{E-APP1})$$

$$\frac{e_2,\ \sigma \longrightarrow e_2',\ \sigma'}{v\ op\ e_2,\ \sigma \longrightarrow v\ op\ e_2',\ \sigma'}\ (\text{OP2}) \qquad \frac{e_2,\ \sigma \longrightarrow e_2',\ \sigma'}{v\ e_2,\ \sigma \longrightarrow v\ e_2',\ \sigma'}\ (\text{E-APP2})$$

$$\mathsf{if\ true\ then}\ e_2\ \mathsf{else}\ e_3,\ \sigma \longrightarrow e_2,\ \sigma\ (\text{IF1}) \qquad \mathsf{if\ false\ then}\ e_2\ \mathsf{else}\ e_3,\ \sigma \longrightarrow e_3,\ \sigma\ (\text{IF2})$$

$$\frac{e_1,\ \sigma \longrightarrow e_1',\ \sigma'}{\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3,\ \sigma \longrightarrow \mathsf{if}\ e_1'\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3,\ \sigma'}\ (\text{IF3})$$

$$\frac{e_1,\ \sigma \longrightarrow e_1',\ \sigma'}{\mathsf{let}\ x:T = e_1\ \mathsf{in}\ e_2,\ \sigma \longrightarrow \mathsf{let}\ x:T = e_1'\ \mathsf{in}\ e_2,\ \sigma'}\ (\text{E-LET1})$$

$$\frac{}{\mathsf{let}\ x:T = v\ \mathsf{in}\ e_2,\ \sigma \longrightarrow \{v/x\}\ e_2,\ \sigma}\ (\text{E-LET2})$$

$$\frac{}{\mathsf{let\ rec}\ f:T_1 \to T_2 = \mathsf{fn}\ y:T_1 \Rightarrow e_1\ \mathsf{in}\ e_2,\ \sigma \longrightarrow \{\alpha/f\}\ e_2,\ \sigma}\ (\text{E-LETREC})$$

$$\alpha = \mathsf{fn}\ y:T_1 \Rightarrow \mathsf{let\ rec}\ f:T_1 \to T_2 = \mathsf{fn}\ y:T_1 \Rightarrow e_1\ \mathsf{in}\ e_1$$

$$\frac{l \in Dom(\sigma)}{l := v,\ \sigma \longrightarrow \mathsf{skip},\ \sigma[l \mapsto v]}\ (\text{ATR1}) \qquad \frac{e,\ \sigma \longrightarrow e',\ \sigma'}{l := e,\ \sigma \longrightarrow l := e',\ \sigma'}\ (\text{ATR2}) \qquad \frac{e_1,\ \sigma \longrightarrow e_1',\ \sigma'}{e_1 := e_2,\ \sigma \longrightarrow e_1' := e_2,\ \sigma'}\ (\text{ATR})$$

$$\frac{l \in Dom(\sigma) \quad \sigma(l) = v}{!\,l,\ \sigma \longrightarrow v,\ \sigma}\ (\text{DEREF}) \qquad \frac{e,\ \sigma \longrightarrow e',\ \sigma'}{\mathsf{new}\ e,\ \sigma \longrightarrow \mathsf{new}\ e',\ \sigma'}\ (\text{NEW})$$

$$\frac{l \notin Dom(\sigma)}{\mathsf{new}\ v,\ \sigma \longrightarrow l,\ \sigma[l \mapsto v]}\ (\text{NEW1}) \qquad \frac{}{\mathsf{skip};e_2,\ \sigma \longrightarrow e_2,\ \sigma}$$

$$\frac{e,\ \sigma \longrightarrow e',\ \sigma'}{!\,e,\ \sigma \longrightarrow\ !\,e',\ \sigma'}\ (\text{DEREF1}) \qquad \frac{e_1,\ \sigma \longrightarrow e_1',\ \sigma'}{e_1;e_2,\ \sigma \longrightarrow e_1';e_2,\ \sigma'}\ (\text{SEQ2})$$

$$\mathsf{while}\ e_1\ \mathsf{do}\ e_2,\ \sigma \longrightarrow \mathsf{if}\ e_1\ \mathsf{then}\ (e_2;\mathsf{while}\ e_1\ \mathsf{do}\ e_2)\ \mathsf{else\ skip},\ \sigma\ (\text{E-WHILE})$$

Sistema de Tipos:

$$\frac{}{\Gamma \vdash n : \mathsf{int}} \quad (\text{T-INT})$$

$$\frac{\Gamma(x) = T}{\Gamma \vdash x : T} \quad (\text{T-VAR})$$

$$\frac{}{\Gamma \vdash b : \mathsf{bool}} \quad (\text{T-BOOL})$$

$$\frac{\Gamma, x \mapsto T \vdash e : T'}{\Gamma \vdash (\mathsf{fn}\ x{:}T \Rightarrow e) : T \to T'} \quad (\text{T-FUN})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{int} \qquad \Gamma \vdash e_2 : \mathsf{int}}{\Gamma \vdash e_1 + e_2 : \mathsf{int}} \quad (\text{T-OP+})$$

$$\frac{\Gamma \vdash e_1 : T \to T' \qquad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1\ e_2 : T'} \quad (\text{T-APP})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{bool} \qquad \Gamma \vdash e_2 : T \qquad \Gamma \vdash e_3 : T}{\Gamma \vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 : T} \quad (\text{T-IF})$$

$$\frac{\Gamma \vdash e_1 : T \qquad \Gamma, x \mapsto T \vdash e_2 : T'}{\Gamma \vdash \mathsf{let}\ x : T = e_1\ \mathsf{in}\ e_2 : T'} \quad (\text{T-LET})$$

$$\frac{\Gamma, x \mapsto T_1, f \mapsto (T_1 \to T_2) \vdash e_1 : T_2 \qquad \Gamma, f \mapsto (T_1 \to T_2) \vdash e_2 : T}{\Gamma \vdash \mathsf{let\ rec}\ f : T_1 \to T_2 = (\mathsf{fn}\ x{:}T_1 \Rightarrow e_1)\ \mathsf{in}\ e_2 : T} \quad (\text{T-LETREC})$$

$$\frac{\Gamma \vdash e_1 : T\ \mathsf{ref} \qquad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 := e_2 : \mathsf{unit}} \quad (\text{T-ATR})$$

$$\frac{}{\Gamma \vdash \mathsf{skip} : \mathsf{unit}} \quad (\text{T-SKIP})$$

$$\frac{\Gamma \vdash e : T\ \mathsf{ref}}{\Gamma \vdash\ !\, e : T} \quad (\text{T-DEREF})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{bool} \qquad \Gamma \vdash e_2 : \mathsf{unit}}{\Gamma \vdash \mathsf{while}\ e_1\ \mathsf{do}\ e_2 : \mathsf{unit}} \quad (\text{T-WHILE})$$

$$\frac{\Gamma \vdash e : T}{\Gamma \vdash \mathsf{new}\ e : T\ \mathsf{ref}} \quad (\text{T-NEW})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{unit} \qquad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1;\ e_2 : T} \quad (\text{T-SEQ})$$

Trabalho

O trabalho consiste em implementar em OCaml um interpretador para a linguagem L1 com as construções imperativas definidas acima. O avaliador do interpretador deve ser implementado seguindo o estilo *big step* com ambientes. Os *data types* `expr` e `tipo` devem se estendidos da seguinte forma (o *data types* `valor` também deve ser modificado para representação dos novos valores da linguagem):

```
type expr =
   ...
|  Asg  of expr * expr
|  Dref of expr
|  New of expr
|  Seq of expr * expr
|  Whl of expr * expr
|  Skip

type tipo =
   ...
|  TyRef of tipo
|  TyUnit
```

O trabalho deve ser realizado em **grupos com 3 componentes.**

Observação: note que endereços representados por l, l_1, ... **não podem aparecer em programas**. Endereços podem aparecer apenas em passos intermediários da avaliação de programas no estilo *small step*.