Expression

An expression could be a number,	e	$::=$	`1|2|3|...`
or a boolean,		$\mid$	`#t|#f`
or addition of two expressions,		$\mid$	`(* e e)`
or a variable,		$\mid$	`x`
or an abstraction,		$\mid$	`(λ (x) e)`
or an application of two expressions.		$\mid$	`(e e)`
or an if branch.		$\mid$	`(if e e e)`

Type

A type could describe a number,
or a boolean,
or a function from one type to another.

$$\tau \; ::= \; \texttt{Nat} \mid \texttt{Bool} \mid (\rightarrow \tau \; \tau)$$

Context

A context could be an empty list, or a pair consed on another context. (The pair is a variable and its type.)	$\begin{array}{lcl} \Gamma & ::= & \epsilon \\ & \mid & x, \tau; \Gamma \end{array}$

Judgment

A judgment is a formalized sentence.
(The sentence may or may not make sense.)

With context Γ, expression e has type τ.	$\Gamma \vdash e : \tau$
In context Γ, variable x has type τ.	$\Gamma\texttt{(x)=}\tau$

Inference Rules

An inference rule has
0^+ *premise judgments*,
a horizontal line,
and 1 *conclusion judgment*.

$$\frac{J_0 \; J_1 \; \ldots}{J}$$

Inference Rule for Multiplication

If	
with Γ, e_1 has type `Nat`	$\Gamma \vdash e_1 : \texttt{Nat}$
and with Γ, e_2 has type `Nat`,	$\Gamma \vdash e_2 : \texttt{Nat}$
then	
with Γ, $(* \; e_1 \; e_2)$ has type `Nat`.	$\Gamma \vdash (* \; e_1 \; e_2) : \texttt{Nat}$

$$\frac{\Gamma \vdash e_1 : \texttt{Nat} \qquad \Gamma \vdash e_2 : \texttt{Nat}}{\Gamma \vdash (* \; e_1 \; e_2) : \texttt{Nat}}$$

Inference Rule for Numbers and Booleans

If (nothing is required) then with Γ, any number n has type `Nat`.	$$\frac{}{\Gamma \vdash n : \texttt{Nat}}$$
If (nothing is required) then with Γ, any boolean b has type `Bool`.	$$\frac{}{\Gamma \vdash b : \texttt{Bool}}$$

Inference Rule for Sub1

If with Γ, e has type `Nat`, then with Γ, $(sub1\ e)$ has type `Nat`.	$$\frac{\Gamma \vdash e : \texttt{Nat}}{\Gamma \vdash (sub1\ e) : \texttt{Nat}}$$

Inference Rule for Zero?

If with Γ, e has type `Nat`, then with Γ, $(\mathit{zero?}\ e)$ has type `Bool`.	$$\frac{\Gamma \vdash e : \texttt{Nat}}{\Gamma \vdash (\mathit{zero?}\ e) : \texttt{Bool}}$$

Inference Rule for If

If	
with Γ, e_1 has type `Bool`,	$\Gamma \vdash e_1 : \texttt{Bool}$
with Γ, e_2 has type τ,	$\Gamma \vdash e_2 : \tau$
with Γ, e_3 has type τ,	$\Gamma \vdash e_3 : \tau$
then	―――――――――
with Γ, $(\mathit{if}\ e_1\ e_2\ e_3)$ has type τ.	$\Gamma \vdash (\mathit{if}\ e_1\ e_2\ e_3) : \tau$

Inference Rule for Variables

If in context Γ, variable x has type τ, then with context Γ, variable x has type τ.	$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

Inference Rule for Applications

If	
in context Γ, e_1 has type $(\rightarrow \tau_{in}\ \tau_{out})$,	$\Gamma \vdash e_1 : (\rightarrow \tau_{in}\ \tau_{out})$
and in context Γ, e_2 has type τ_{in} ,	$\Gamma \vdash e_2 : \tau_{in}$
then	———————————
with context Γ, $(e_1\ e_2)$ has type τ_{out}.	$\Gamma \vdash (e_1\ e_2) : \tau_{out}$

Inference Rule for Abstractions

If with the extended Γ, e has type τ_{out}, then with Γ, `(λ (x) e)` has type $(\rightarrow \tau_{in}\, \tau_{out})$	$$\frac{x : \tau_{in}; \Gamma \vdash e : \tau_{out}}{\Gamma \vdash (\lambda\ (\mathtt{x})\ \mathtt{e}) : (\rightarrow \tau_{in}\, \tau_{out})}$$

Inference Rule for Fix

We need this rule for recursion because `(x x)` does not type check.

If with the extended Γ, e has type τ, then with Γ, `(fix (λ (x) e))` has type τ	$$\frac{x : \tau; \Gamma \vdash e : \tau}{\Gamma \vdash \texttt{(fix (}\lambda\texttt{ (x) e))} : \tau}$$