A Simple Language and its Type System

Syntax for types:

$\tau \rightarrow$	Int | Char | Bool | ...
$\tau \rightarrow$	Pointer(τ) | Tuple(τ, τ) | List(τ) | Record(label τ, label $\tau, \ldots$)

Syntax for expressions:

$e \rightarrow$	< intLit > | < charLit> | <boolLit> | <listLit> | ...
$e \rightarrow$	varId
$e \rightarrow$	(e)
$e \rightarrow$	e mod e | $e + e$ |...
$e \rightarrow$	e eq e
$e \rightarrow$	deref(e)
$e \rightarrow$	fst(e) | snd(e) | pair (e, e)
$e \rightarrow$	hd(e) | tail(e) | cons (e, e)
etc.	
<intLit>$\rightarrow$	1|2|3|4|5|6|7|8|9|0
<listLit>$\rightarrow$	nil
etc.	

Assigning types to expressions by deduction:

1. Assumptions / Environment

1 : Int , 2 : Int ... , true: Bool,..., nil : List(??) , ...
var : τ /* need to be told this (declarations!)or figure it out somehow */

2. Deduction rules

[Const] ... c:τ yields c:τ

[Var] ... y:τ yields y:τ

[Arithm]
$$\frac{\vdash e_1\text{:Int}, \vdash e_2\text{:Int}}{\vdash (e_1 \text{ mod } e_2)\text{: Int}}$$

[Eq]
$$\frac{\vdash e_1 : \tau, \vdash e_2 : \tau}{\vdash (e_1 \text{ eq } e_2) : \text{Bool}}$$

[Deref] $\dfrac{\vdash e\text{: Pointer } (\tau)}{\text{deref}(e) : \tau}$

[fst] $\dfrac{\vdash e \text{ Tuple}(\tau_1, \tau_2)}{\vdash \text{fst}(e) : \tau_1}$

[pair] $\dfrac{\vdash e_1 : \tau_1, \vdash e_2 : \tau_2}{\vdash \text{pair } (e_1, e_2) : \text{Tuple } (\tau_1, \tau_2)}$