Chapter 1

Library ex_ty

```
From Coq Require Import Arith.Arith.
From PLF Require Import SmallstepAM.
From PLF Require Import TypesAM.
From Hammer Require Import Tactics.
Import TM.
```

In this exercise, we explore different ways to talk about the arithmetic language seen in Types. You can use *sauto*, but you *must* pass it the right arguments (ctrs/inv/use)

Part 1.1: define a relational big step semantics for evaluating *tm*.

```
Inductive eval : tm → tm → Prop :=.
```

Part 1.2. now prove the value soundness theorem:

```
Theorem vs: ∀ t v, eval t v → value v.
```

If you have problems, perhaps you need a different notion of *value*

1.3. Show that `eval` is a *partial* function

```
Theorem eval_det: deterministic eval.
```

1.4. Prove preservation:

```
Theorem preservationB : ∀ (t : tm) T,
  has_type t T → ∀ t',
  eval t t' →
  has_type t T.
```

Part 2: write a (functional) type checker. You will need to define a *boolean* test for *ty* equality. To make the code more concise, we suggest to use the let monadic notation used in the ImpCevalFun chapter.

```
Fixpoint typeof
    (t : tm) : option ty.
Admitted.
```

2.1 Prove that the relational version entails the functional one.

```
Theorem rel2f: ∀ t T, has_type t T → typeof t = Some T.
```

Extra credit: prove the other direction