Work both problems below.

**Equational specification of a queue**

For this problem I will use ML type notation. Suppose that you have a polymorphic queue with the following signature:

```ml
exception Empty
val empty : forall 'a . 'a queue
val put : forall 'a . 'a * 'a queue -> 'a queue
val get : forall 'a . 'a queue -> 'a * 'a queue (* raises Empty *)
```

And in addition, suppose that you are provided with the following operations on pairs:

```ml
val fst : forall 'a, 'b . 'a * 'b -> 'a
val snd : forall 'a, 'b . 'a * 'b -> 'b
```

To solve this problem, *give algebraic laws* that are sufficient to specify the results of all well-behaved combinations of `put`, `get`, and `empty`.

*Hint:* Although `get` returns a pair, which is a first-class value, you will find it easier to write equations involving `fst (get q)` and `snd (get q)`.

**Equational specification of a filesystem**

A source-code control system is essentially a filesystem with a history. For today, let's ignore the problem of history. For a filesystem without history, write the *signature* and *algebraic laws* that you think are important from the perspective of source-code control.