The Specification Game

When evaluating specifications, it can be useful to imagine that a game is being played between two people: a *specifier* and a *devious programmer*.

Suppose that the specifier writes the following specification:

```
(** returns a list *)
val reverse : 'a list -> 'a list
```

This spec is clearly incomplete. For example, a devious programmer could meet the spec with an implementation that gives the following output:

```
# reverse [1;2;3];;
- : int list = []
```

The specifier, upon realizing this, refines the spec as follows:

```
(** [reverse lst] returns a list that is the same length as [lst] *)
val reverse : 'a list -> 'a list
```

But the devious specifier discovers that the spec still allows broken implementations:

```
# reverse [1;2;3];;
- : int list = [0;0;0]
```

Finally, the specifier settles on a third spec:

```
(** [reverse lst] returns a list [m] satisfying the following conditions:
    - [length l = length m]
    - for all [i], [nth m i = nth l (n - i - 1)],
      where [n] is the length of [lst] *)
val reverse : 'a list -> 'a list
```

With this spec, the devious programmer is forced to provide a working implementation to meet the spec, so the specifier has successfully written her spec.