You should try to solve the exercises 9.1 and 9.2 from H&R Chapter 9, and exercise 10.1 from Chapter 10. The exercises 10.2 and 10.3 are also good, though rather formal. From Chapter 11 you should try to solve the exercises 11.1 and 11.4.

Handing in exercise E6.1 will make you a happier person.

Exercise E6.1 (Representing sets as binary search trees)

In this exercise you'll make two set libraries. One library using just a signature and a structure. The other using a functor. In both of the libraries you should represent sets as binary search trees (see H&R Section 8.5 and the slides for lecture 5).

Your set libraries should at least have the following values:

- `empty` The empty set (can be a constant or a function).
- `member` Check that an element is in the set.
- `insert` Insert an element into the set.
- `fold` Accumulate a function over the elements of the set.
- `card` Compute the size of the set.
- `toList` Generate a list of the elements in the set.

Feel free to extend your library with other functions from H&R Chapter 10 (start with `union`).

(a) Make a set library where a set is represented as a binary search tree containing the elements in the set and a comparison function (like `Int.compare`) to compare the elements.
(b) Make a functor that takes a structure matching the signature ORD as argument and gives a set structure as result.

```ml
signature ORD =
  sig
    type t
    val compare : t * t -> order
  end
```