Notes 1.30-1.33

- Turning a recursive process to an iterative process
  - Defining an invariant quantity
    - One function, extra parameter
  - Defining a nested function
    - Inside the enclosing function (2)
- Turning an iterative process to a recursive process
  - Remove invariant quantity if applicable
  - Add an operation to the left of the recursive call
- Higher order procedures: procedures that manipulate other procedures
  - Take a function as an input parameter
  - Can serve as a powerful abstraction (only the important data is displayed to the user) mechanism
  - Taking common patterns and simplifying it to apply to a wider range of functions
- Nested function
  - Invisible outside of its immediately enclosing function
  - Nested functions can use any parameters/variables that are defined in the enclosing functions

Basic pattern for recursive high-order functions

```
(define (<name> a b)
  (if (> a b)
      0
      (+ (<term> a)
        (<name> (<next> a) b))))
```

Extra

- Add/subtract → end condition with zero
- Multiply/divide → end condition with one