Functional Programming
Exercises Week 1
(for October 11, 2013)

Numbers in parentheses refer to the 6th edition of the course notes.

1. Read Chapter 1 and Appendix A of the lecture notes.

2. (Exercise A.1) Give at least one value of each basic type and the types `coord`, `direction`, and `number`.

3. (Exercise A.2) Give at least five examples of mathematical functions that would have the type `int -> int -> int` in OCaml.

4. (Exercise A.3) Write down the type of '`let f x = x - 1`' and justify your answer.

5. (Exercise A.4) Explain (in words) what the type `string -> char -> bool` means. Can you imagine any meaningful function having this type?

6. (Exercise A.7; list notation modified) Consider the user-defined type for lists

```
type 'a mylist = Nil | Cons of ('a * 'a mylist)
```

Which of the following items are patterns?

a) `x`

b) `_`

c) `3.14`

d) `Nil`

e) `Cons(x,xs)`

f) `Cons(x,Cons(y,xs))`

g) `n,Cons(x,xs) as ys`

h) `(n,Cons(x,xs)) as (_,ys)`

i) `(n,Cons(x,xs)) when n < 0`

j) `(n,Cons(x,xs)) | (_,Nil)`

k) `(n,Cons(x,xs)) when x < 0 | (_,Nil)`

Hint: Also try them in the interpreter by replacing the dots in:

```
let foo = function | ... -> ()
```

7. (Exercise A.10) Using pattern matching, define a function `square` that computes the square of a `number`.

8. (Exercise A.17) Implement a function `c : int -> int` which computes the following:

$$c(n) = \begin{cases} n & \text{if } n \leqslant 1 \\ c(3n+1) & \text{if } n > 1 \text{ and } n \text{ is odd} \\ c(\frac{n}{2}) & \text{if } n > 1 \text{ and } n \text{ is even} \end{cases}$$

At each recursive call print the value of `n`.

Trivia: This function is known as *Collatz'* or *Syracuse* function.

Bonus: Prove that for `n` greater 2 this function always ends with the values $4, 2, 1$.