Functional Programming Exercises Week 2

(for October 18, 2013)

Numbers in parentheses refer to the 6th edition of the course notes.
Exercises marked with ⋆ are optional and can be ignored.
Please have your `.ml` files accessible (e.g. on `zid-gpl.uibk.ac.at`).

1. Read Chapter 2 and Appendix B of the lecture notes.

2. (Exercise 2.2) Evaluate the function calls `range 0 3` and `range 3 0` by giving all computation steps on paper.

3. (Exercise 2.6) Write a function `last : 'a list -> 'a` to return the last element of a list.

4. (Exercise 2.8) Write a function `reverse : 'a list -> 'a list` that reverses a list by (recursively) appending the head element at the tail. Then `reverse` $[x_1; \ldots ; x_n]$ evaluates to $[x_n; \ldots ; x_1]$.

5. (Exercise 2.12) Write a function `zip` which takes two lists and returns a list of pairs with the same length as the shorter of both as in the following example:

```
zip [1;2;3] [4;5;6;7;8] = [(1,4);(2,5);(3,6)]
```

6. (Exercise 2.21) Write a recursive function `euler : int -> float` that computes an approximation of Euler's number:

$$\texttt{euler}(n) = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{(n-1)!}$$

⋆. (Exercise 2.25) You have 100 doors (#1 to #100) in a row that are all initially closed. You make 100 passes by the doors. The first time through, you visit every door and toggle the door (if the door is closed, you open it; if it is open, you close it). The second time you only visit every 2nd door (door #2, #4, #6, ...). The third time, every 3rd door (door #3, #6, #9, ...), etc, until you only visit the 100th door.

a) Define a type `t` for doors (open/closed).

b) Write a function `toggle : t -> t` that toggles the door.

c) Generate a list of 100 closed doors.

d) Write a function `pass : int -> t Lst.t -> t Lst.t` that executes the i-th pass over the doors.

e) Write a function `passes : int -> t Lst.t -> t Lst.t` that executes n passes over the doors.