

Functional Programming

Exercises Week 12

(for January 16, 2009)

1. Read Chapter 10 of the lecture notes.
2. Consider following type for ‘lazy’ lists

```
type 'a cell = Nil
           | Cons of ('a * 'a llist)
and 'a llist = (unit -> 'a cell)
```

Implement the functions

- `append : 'a llist -> 'a llist -> 'a llist` that concatenates two lazy lists
 - and `map : ('a -> 'b) -> 'a llist -> 'b llist` that maps a function over a lazy list.
3. Implement the *Sieve of Eratosthenes* using the type of Exercise 2.
 4. Consider this alternative type for lazy lists

```
type 'a cell = Nil
           | Cons of ('a * 'a llist)
and 'a llist = 'a cell Lazy.t
```

Implement `fibs : int llist`, computing the infinite list of Fibonacci numbers.

5. Use the type of Exercise 4 to implement a function `merge` that combines two `llists` lazily—assuming they were sorted—thereby removing duplicates, i.e.,

$$\text{merge } (1, 4, 6, 7, 8, \dots) (1, 2, 3, 4, 10, \dots) = (1, 2, 3, 4, 6, 7, 8, 10, \dots)$$

6. Define the infinite sequence of *Hamming numbers* (i.e., all natural numbers whose only prime factors are 2, 3, and 5).

Hint: You will need at least `map` and `merge`.