Functional Programming

Exam June 8th 2012

Exercise 1. Give types of the following expressions, either by guessing or inferring them by hand:

1. `let double f y = f (f y) in fun g x -> double (g x)`
2. `let rec tails l = match l with [] -> [] | x::xs -> xs::tails xs in`
   `fun l -> List.combine l (tails l)`

Exercise 2. Assume that the corresponding expression from previous exercise is bound to name `foo`. What are the values computed for the expressions (compute in your head or derive on paper):

1. `foo (+) 2 3, foo ( * ) 2 3, foo ( * ) 3 2`
2. `foo [1; 2; 3]`

Exercise 3. Give example expressions that have the following types (without using type constraints):

1. `(int -> int) -> bool`
2. `'a option -> 'a list`

Exercise 4. Write function that returns the list of all lists containing elements from the input list, preserving order from the input list, but without two elements.

Exercise 5. Write a breadth-first-search function that returns an element from a binary tree for which a predicate holds, or `None` if no such element exists. The function should have signature:

```
val bfs : ('a -> bool) -> 'a btree -> 'a option
```

Exercise 6. Solve the n-queens problem using backtracking based on lists.

Available functions: `from_to`, `concat_map`, `concat_foldl`, `unique`.

Hint functions (asking for hint each loses one point): `valid_queens`, `add_queen`, `find_queen`, `find_queens`. Final function `solve` takes n as an argument. Each function, other than `valid_queens` that takes 3 lines, fits on one line.

Exercise 7. Provide an algebraic specification and an implementation for first-in-first-out queues (lecture 5 exercise 9).