Functional Programming

# Mapping and folding
# List-based backtracking

**Exercise 1.** Recall how we generated all subsequences of a list. Find (i.e. generate) all:

1. permutations of a list;
2. ways of choosing without repetition from a list;
3. combinations of K distinct objects chosen from the N elements of a list.

**Exercise 2.** Using folding for the `expression` data type, compute the degree of the corresponding polynomial. See http://en.wikipedia.org/wiki/Degree_of_a_polynomial.

**Exercise 3.** Implement simplification of expressions using mapping for the `expression` data type.

**Exercise 4.** Express in terms of `fold_left` or `fold_right`:

1. `indexed : 'a list -> (int * 'a) list`, which pairs elements with their indices in the list;
2. * `concat_fold`, as used in the solution of *Honey Islands* puzzle:
   - ```
     let rec concat_fold f a = function
       | [] -> [a]
       | x::xs ->
         f x a |-> (fun a' -> concat_fold f a' xs)
     ```
   - Hint – consider the function:
     ```
     let rec concat_foldl f a = function
       | [] -> a
       | x::xs -> concat_foldl f (concat_map (f x) a) xs
     ```
3. run-length encoding of a list (exercise 10 from *99 Problems*).
   - `encode [‘a;‘a;‘a;‘b;‘c;‘c;‘a;‘a;‘d] = [4,‘a; 1,‘b; 2,‘c; 2,‘a; 1,‘d]`

**Exercise 5.**

1. Write a more efficient variant of `list_diff` that computes the difference of sets represented as sorted lists.
2. `is_unique` in the provided code takes quadratic time – optimize it.

**Exercise 6.** Write functions `compose` and `perform` that take a list of functions and return their composition, i.e. a function `compose [f1; ...; fn] = x ↦ f1 (... (fn x)...)` and `perform [f1; ...; fn] = x ↦ fn (... (f1 x)...)`.

**Exercise 7.** Write a solver for the *Tents Puzzle* http://www.mathsisfun.com/games/tents-puzzle.html.

**Exercise 8.** * **Robot Squad**. We are given a map of terrain with empty spaces and walls, and lidar readings for multiple robots, 8 readings of the distance to wall or another robot, for each robot. Robots are equipped with compasses, the lidar readings are in directions E, NE, N, NW, W, SW, S, SE. Determine the possible positions of robots.

**Exercise 9.** * Write a solver for the *Plinx Puzzle* http://www.mathsisfun.com/games/plinx-puzzle.html. It does not need to always return correct solutions but it should correctly solve the initial levels from the game.
     ```