Exam set 2

Exercise 1. (Blue.) What is the type of the subexpression `f` as part of the expression below assuming that the whole expression has the type given?

```
(fun double g x -> double (g x)) (fun f y -> [f] (f y))
 : ('a -> 'b -> 'b) -> 'a -> 'b -> 'b
```

Exercise 2. (Blue.) Write an example function with type:

```
(int -> int list) -> bool
```

Tell "in your words" what it does.

Exercise 3. (Green.) Find the number of elements of a list.

Exercise 4. (Green.) Split a list into two parts; the length of the first part is given.

Exercise 5. (Yellow.) Rotate a list N places to the left.

Exercise 6. (Yellow.) Let us call a binary tree symmetric if you can draw a vertical line through the root node and then the right subtree is the mirror image of the left subtree. Write a function `is_symmetric` to check whether a given binary tree is symmetric.

Exercise 7. (White.) By "traverse a tree" we mean: write a function that takes a tree and returns a list of values in the nodes of the tree. Traverse a tree in breadth-first order – first values in more shallow nodes.

Exercise 8. (White.) Generate all combinations of K distinct elements chosen from the N elements of a list.

Exercise 9. (Orange.) Implement a topological sort of a graph: write a function that either returns a list of graph nodes in topological order or informs (via exception or option type) that the graph has a cycle.

Exercise 10. (Orange.) Express `fold_left` in terms of `fold_right`. Hint: continuation passing style.

Exercise 11. (Purple.) Show why for a monomorphic specification, if datastructures d_1 and d_2 have the same behavior under all operations, then they have the same representation $d_1 = d_2$ in all implementations.

Exercise 12. (Purple.) `append` for lazy lists returns in constant time. Where has its linear-time complexity gone? Explain how you would account for this in a time complexity analysis.

Exercise 13. (Red.) Write a function `ms_tree graph` to construct the *minimal spanning tree* of a given weighted graph. A weighted graph will be represented as follows:

```
type 'a weighted_graph = {nodes : 'a list; edges : ('a * 'a * int) list}
```

The labels identify the nodes `'a` uniquely and there is at most one edge between a pair of nodes. A triple `(a,b,w)` inside `edges` corresponds to edge between `a` and `b` with weight `w`. The minimal spanning tree is a subset of `edges` that forms an undirected tree, covers all nodes of the graph, and has the minimal sum of weights.

Exercise 14. (Crimson.) Von Koch's conjecture. Given a tree with N nodes (and hence N-1 edges). Find a way to enumerate the nodes from 1 to N and, accordingly, the edges from 1 to N-1 in such a way, that for each edge K the difference of its node numbers equals to K. The conjecture is that this is always possible.

For small trees the problem is easy to solve by hand. However, for larger trees, and 14 is already very large, it is extremely difficult to find a solution. And remember, we don't know for sure whether there is always a solution!

Write a function that calculates a numbering scheme for a given tree. What is the solution for the larger tree pictured above?

Exercise 15. (Black.) Based on our search engine implementation, write a function that for a list of keywords returns three best "next keyword" suggestions (in some sense of "best", e.g. occurring in most of documents containing the given words).