universität innsbruck

Advanced Functional Programming | WS 2026/2027 | LVA 703139

Exercise Sheet 5, 10 points | Deadline: Tuesday, November 10, 2026, 4pm

- Solve the tasks in file `Exercise05.hs` and upload only this file in OLAT.
- Mark the solved exercises in OLAT.
- Your modified `Exercise05.hs` file must compile with `ghci` without error messages.

Task 1 *State monad* **3 p.**

The datatype `Term` (known from the implementation of the embedding-relation) represents first-order terms in Haskell.

```
data Term f v = Var v | Fun f [Term f v]
```

Implement a function that labels each subterm of a term by a unique number.

```
type LTerm f v = Term (Int,f) (Int,v)
labelTerm :: Term f v -> LTerm f v
```

For instance, the Haskell representation of the term $f(x, g(y), x)$ might be labeled as $f_0(x_1, g_2(y_3), x_4)$. Note that both occurrences of x are labeled differently.

Important: Use a monadic programming style and the `State` monad to implement the labeling function.

Task 2 *Minsort and Mutable Arrays* **7 p.**

Minsort is a sorting algorithm that works via the following idea:

- if the list is not empty, then determine its minimum, and move it to the front of list;
- repeat with the remaining list, if that is non-empty.

1. Implement `minSort :: Ord a => [a] -> [a]` in a purely functional style. (1 point)
2. Implement `minSortM :: (MArray a e m, Ord e) => [e] -> m (a Int e)` in a monadic style, using mutable arrays. Here, from the input list a mutable array should be created, that is then sorted in-place and returned. (3 points)
3. Define two wrapper functions `minSortST :: Ord a => [a] -> [a]` and `minSortSTU :: [Int] -> [Int]` around `minSortM`. The former selects arrays of type `STArray` and the latter arrays of type `STUArray`.

   Run `cabal run` to evaluate the performance of `minSort`, `minSortST` and `minSortSTU`. (1 point)
4. (Challenging) Change the type of `minSortSTU` to something more generic, so that in particular `minSortSTU` can be invoked on lists of various fixed-byte-width types such as `[Int]`, `[Float]`, `[Char]`, ... (2 points)