Advanced Functional Programming WS 2025/2026 LVA 703139

Exercise Sheet 3, 10 points Deadline: Tuesday, October 28, 2025, 4pm

- Solve the tasks in file `Exercise03.hs` and upload only this file in OLAT.
- Mark the solved exercises in OLAT.
- Your modified `Exercise03.hs` file must compile with `ghci` without error messages.

## Task 1 *Type Assignment* **2 p.**

Provide a type assignment for `fun`, so that the following defining equation is typeable:

```
fun c = c 'i' c
```

## Task 2 *Maps and Reachable Nodes* **8 p.**

Consider the Haskell program in `Exercise03.hs`. The first three of the upcoming four tasks can be solved independently.

1. The program contains a class definition `Map` and two instance declarations for `AList` and for `Tree`.

   Figure out the effect of the language extensions `MultiParamTypeClasses` and `FlexibleInstances`

   https://ghc.gitlab.haskell.org/ghc/doc/users_guide/exts.html

   and describe why these are crucial for this part of the Haskell program. 2 points

2. Complete one of the two instance declaration, i.e., either for `AList` or for `Tree`. It is not allowed to import `Data.Map` or other predefined maps. 2 points

3. The function `reach` computes for a given graph and starting node all reachable nodes, including the paths. For instance, `testE` evaluates to `[[E,A],[E,C,D,B],[E,C,D],[E,C],[E]]`, i.e., from `E` we can reach all of `A`,`B`,`D`,`C`,`E` by following the paths that are represented in the list.

   The implementation of `reach` via `reachMainList` is completely list-based.

   Provide a more general implementation `reachMain` that works with abstract maps instead of lists. 2 points

4. Define two instances of the reachability algorithm that are based on `reachMain`, one that internally uses `Tree`-maps, and one that used `AList`-maps. 2 points