Program Design and Data Structures, 20.0 c

Course code: 1DL201, Report code: DL201, 67%, DAG, NML,
week: 44 - 02 Semester: Autumn 2017
week: 03 - 11 Semester: Spring 2018

LAB ASSIGNMENT 11

This information is not available in English. Now showing the Swedish version.

After this lab you should be able to ...

- Work with trees.

Instructions

- Read the slides on trees (lecture 15-16, starting from slide 48) and on binary trees (lecture 17, until slide 15).
- Remember to write function specifications for all functions that you write, and representation conventions/invariants for all data types that you define! Do this *before* you write your Haskell code. (Recursion variants may be specified after the code has been written.)
- Also remember to follow the other parts of our coding convention (for identifiers and indentation).

The task

PROBLEM 1

a) Define a data type FamilyTree for a family tree (similar to slide 48 of lecture 20-21), but with the folllowing differences:

- The family tree shall be a (not necessarily full) binary tree.
- There shall be two different kinds of nodes: one for women, one for men. These shall be represented by two different constructors.
- Each node shall contain the person's name (of type String) and year of birth (of type Int).

b) Write a function oldestWoman :: FamilyTree -> Maybe (String,Int) that takes a family tree as an argument and returns a value that indicates the name and year of birth of the oldest woman in the tree. (If there are several women that might be oldest, the function shall return just one of them; it doesn't matter which one.) If there is no woman in the tree, the function shall return Nothing.

Some examples:

```
oldestWoman Void = Nothing
oldestWoman (Man "Arn" 1150 Void Void) = Nothing
oldestWoman (Woman "Anna" 1992 Void Void) = Just ("Anna", 1992)
oldestWoman (Woman "Anna" 1992 (Woman "Maria" 1989 Void Void) (Man "Arn" 1150 Void Void)) = Just ("Maria", 1989)
oldestWoman (Man "Arn" 1150 (Woman "Sigrid" 1120 Void Void) (Man "Magnus" 1111 (Man "Bengt" 1070 Void Void) (Woman "Gun" 1085 Void Void))) = Just ("Gun", 1085)
oldestWoman (Man "Arn" 1150 Void (Man "Magnus" 1111 Void Void)) = Nothing
```

Hint: One difficulty in this task is how to combine the results of recursive calls (and the data at the root node). If `oldestWoman` were to return just a year of birth, you could simply use min to combine years from different subtrees (and the year at the root node). However, here the recursive calls each return a value of type Maybe (String,Int), i.e., just a pair or Nothing. Start by writing an auxiliary function that takes two values of this type and returns the name and birthyear of the older person (or Nothing). Then you can use this auxiliary function to combine the results of recursive calls (and the data at the root node).

c) Instead of having separate constructors for women's and men's nodes, would it be better to have just one constructor for both women and men, but with an additional argument that indicates the sex of a person (e.g., of type Bool or some other two-valued enumeration type)? Give an answer (yes, no, it depends, ...) and explain in detail why you answered as you did.

PROBLEM 2

a) Give an example of information (other than the examples that were discussed in lectures: family trees, arithmetic expressions) that is conveniently represented in tree form.

b) Write a `data` declaration for representing your information from part a).

WHEN YOU ARE DONE

When you are done with all problems (the explorations below are optional), raise your hand or approach an assistant to have your solution graded.

If you pass this lab at least 30 minutes early and other groups are still working on it, we ask you to **help one other group**. Do not simply share your solution with them, but try to understand the (partial) solutions that they have developed so far (which may be different from yours) and the difficulties that they have. Assist them in coming up

with their own solutions. Once the group that you are helping completes the lab assignment, you may leave. (If it is still early, the group that received your help should then stay to help another group.)

EXPLORATIONS

If you have finished the lab exercises and are waiting to be graded, or if you wish to explore programming in Haskell further at any time, or want to get some extra practice, have a look at the Explorations page.

Exploration problems are optional and should only be attempted after other problems are completed.
You don't have to show these answers to the lab assistants for grading.