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 3

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

After this lab session you should be able to....

- Use pattern matching
- Use simple data structures (tuples)
- Construct simple algorithms

Instructions

1. Take notes while you work, to make it easier to show what you have done, and to answer questions from the assistant who grades your lab assignment.
2. Remember to write function specifications for all (non-anonymous) functions that you write. Do this *before* you write your Haskell code. Also remember to follow the other parts of our coding convention (for identifiers and indentation).

The Task

PROBLEM 1

In the previous lab assignment you studied this function:

```
drJeep :: String -> String -> Bool
drJeep x y =
  not (length x < length y) &&  drop (length x - length y) x == y
```

1. Describe, in words, *what* the function does. No need to consider *how* the function works, but rather describe what the function returns (independent of the implementation details).
2. Think of a better (more descriptive) name for drJeep, a name that describes what the function does.
3. Similarly, think of descriptive names for the arguments *x* and *y*.

PROBLEM 2

Assume that a database contains entries with name and address information about persons. Every entry is represented as a tuple with the following components (in order):

1. Family name (String)
2. Given name (String)
3. Street address (String)
4. Postal number (Int)
5. Postal address (String)

Write a function *updatePersonName* which takes three arguments: one such tuple, an integer, and a string. The function should return the same tuple, except that either the family name or the given name has been swapped for the given string. Which name should be swapped is controlled by the integer: if it is 1 the given name should be swapped, and if it is 2 the family name should be swapped. If the integer is neither 1 nor 2, the function should simply return the argument tuple unchanged.

The function **must** be written so that it:

- extracts components from the tuple using pattern matching, not any other way.
- uses pattern matching to select which of the possible tasks to perform, not using *if*.

Some examples:

updatePersonName ("Stark", "Ned", "Winterfell", 55542, "Westeros") 1 "Slightly less tall Ned"
=> ("Stark", "Slightly less tall Ned", "Winterfell", 55542, "Westeros")

updatePersonName ("Stark", "Ned", "Winterfell", 55542, "Westeros") 2 "the Nogginless"
=> ("the Nogginless", "Ned", "Winterfell", 55542, "Westeros")

PROBLEM 3

Write an algorithm (in natural language, e.g., Swedish/English, as a flowchart, pseudo code or similar - *not* as program code) that translates a natural number (i.e., an integer greater than or equal to zero) into a string. For instance, if the number is 42, the algorithm should compute the string "42".

You can take the translation of numbers 0, ..., 9 to the corresponding strings "0", ..., "9" as given, and do not need to describe it further.

Hints: You will need to divide the number by 10 and separately compute the quotient and remainder. When you have written the algorithm, go through it by hand for a single digit, a double digit and a triple digit number to see that it works. Think about what your algorithm does when it receives the number 0. Do you have to change anything? If so, make sure that the change works for other numbers! For instance, make sure 42 is not translated as "042".

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.