02157 Functional programming

Michael R. Hansen
DTU Compute
September 27, 2017

Mandatory assignment 1

This is the first of four mandatory assignments in 02157 Functional programming. It is a requirement for exam participation that 3 of the 4 mandatory assignments are approved. The mandatory assignments can be solved individually or in groups of 2 or 3 students.

- Your solution should be handed in **no later than Thursday, October 5, 2017**. Submissions handed in after the deadline will face an *administrative rejection.*
- To submit you should upload a single F# file (*file*`.fsx` or *file*`.fs`) to CampusNet under Assignment 1. The file should start with full names and study numbers for all members of the group. If the group members did not contribute equally to the solution, then the role of each student must be explicitly stated.
- Your solution should be submitted as a complete program that can be uploaded to F# Interactive without encountering compilation errors. Failure to comply with that will result in an *administrative rejection of your submission.*
- Be careful that you submit the right file. A submission of a wrong file will result in an *administrative rejection of your submission.*
- **DO NOT COPY solutions** from others and **DO NOT SHARE your solution** with others. Both cases are considered as fraud and will be reported.
- Further requirements to your solution are listed below.

Problem formulation

A company has a club for employees, and this club has a register containing names and descriptions of every club member. The description of a member is a tuple (no, yb, ths), where no is a telephone number, yb is the year of birth, and ths is the themes of interests of the member.

A club arrangement is given by a predicate[1] describing the club members that may be interested in the arrangement. Let, for example, the predicate $p_1(no, yb, ths)$ be true for a given description (no, yb, ths) of a club member when yb is greater than 1982 and ths includes "soccer" and "jazz". This predicate describes an arrangement directed to young club members that are interested in *both* soccer *and* jazz.

Your solution to this assignment should be presented using the model-based approach discussed in Section 4.6 in the textbook. In particular your solution should contain:

1. Types for the important concepts of the problem formulation including, at least, types for the register (a list type of some kind), themes of interests (a list type of some kind), descriptions and arrangements.

2. A declaration of a register `reg`, a declaration of an arrangement `p1` for the above described arrangement p_1, and a declaration of an arrangement `p2` that is directed to young club members that are interested in *either* "soccer" *or* "jazz" *or both.* These declarations should be constructed so that they can serve as illustrative examples.

3. A declaration of a function `extractInterested` p r that gives a list with names and phone numbers of the members in register r that may be interested in the arrangement p. State the type of `extractInterested` in a comment. Make use of the type names introduced under point 1. above, so that the type reflects the intention with the function. The function `extractInterested` provides the main functionality of your program — do *NOT* make a free-standing program with a `main` function.

4. Tests of `extractInterested` involving `reg`, `p1` and `p2`. Each concrete test must be accompained with the expected result of the test. The tests should be organized as follows:

   `let test1` $=$ `extractInterested p1 reg` $= [x_1; \ldots ; x_m];;$
   `let test2` $=$ `extractInterested p2 reg` $= [y_1; \ldots ; y_n];;$

   where $[x_1; \ldots ; x_m]$ and $[y_1; \ldots ; y_n]$ are the expected results of `test1` and `test2`.

5. Types for other functions declared in your solution. These types should be stated in comments above the respective functions and the types should reflect the intention with the functions (see point 3.).

6. Structural tests (that is, white-box tests). For each function declared in your program give test cases showing that every branch of the function is executed and works correctly. The expected result for each test must be stated explicitly. These tests should be organized as described in 4. above.

[1]A predicate is a function that returns a truth value