Homework 5: Erlang and Programming Language Concepts

See Webcourses and the syllabus for due dates.

Purpose

In this homework there are problems to try to bring out and reinforce concepts related to Erlang and programming languages in general, including the relation of these concepts to languages like Java, C, and C++. [Concepts] [MapToLanguages].

Directions

For this homework we suggest that you work individually. (However, per the course's grading policy you can work in a group if you wish, provided that carefully follow the policy on cooperation described in the course's grading policy.)
Don't hesitate to contact the staff if you are stuck at some point.

What to Turn In

For English answers, please paste your answer into the assignment as a "text answer" in the problem's "assignment" on Webcourses (or upload them as a file).

What to Read

A good reference for this homework should be your class notes.
For supporting answers about questions about various languages, we recommend reading the language's reference manual. For Erlang, see The Erlang Reference Manual (Sep. 2019). For Haskell, see the Haskell 2010 Language Report (2010). For Java, see the Java Language Specification (Java SE 13 Edition, August 2019). There is an ANSI standard for C and C++, which should be at the UCF library.

Problems

1. (10 points) [UseModels] Consider the function `mapInside`, specified as follows in Erlang,

   ```
   -spec mapInside(F::fun((T) -> R), Llt::list(list(T))) -> list(list(R)).
   ```

   that for all types `T` and `R`, takes a function `F` from `T` to `R`, and a list of lists, `Llt`, whose elements are lists of type `T`, and returns a list of lists whose elements are the lists resulting from mapping `F` over the lists of type `T` inside `Llt` in order. The following are examples.

   ```
   -module(mapInside_tests).
   -export([main/0,fromTo/2,fromToBy/3]).
   -import(testing,[dotests/2,eqTest/3]).
   -import(mapInside,[mapInside/2]).
   main() ->
       compile:file(mapInside), dotests("mapInside_tests Revision : 1.1", tests()).
   tests() ->
       [eqTest(mapInside(fun (N) -> N+1 end, []), "==", [])
       ,eqTest(mapInside(fun (N) -> N+1 end, [[]]), "==", [[]])
       ,eqTest(mapInside(fun (N) -> N+1 end, [[6,10]]), "==", [[7,11]])
       ,eqTest(mapInside(fun (N) -> N+3 end, [[5],[],[10,20,3]]),
               "==", [[8],[],[13,23,6]])
       ,eqTest(mapInside(fun (I) -> (I*2+1)*I end, [[],[4],[9,8,3,5],[]]),
               "==", [[],[36],[171,136,21,55],[]])
       ,eqTest(mapInside(fun (I) -> I*2 end, [fromTo(1,100),[3],[],[1]]),
               "==", [fromToBy(2,200,2),[6],[],[2]])
       ,eqTest(mapInside(fun (I) -> I rem 2 == 0 end, [[4,8,5],[3],[],[4]]),
               "==", [[true,true,false],[false],[],[true]])     ].

   % The following functions are for testing purposes only.
   fromTo(LB,UB) -> fromToBy(LB,UB,1).
   fromToBy(LB,UB,N) ->
       case LB =< UB of
           true -> [LB|fromToBy(LB+N,UB,N)];
           false -> []
       end.
   ```

   Which of the following correctly uses the `foldr` function from the Erlang `lists` module to implement `mapInside`? (Assume that both `foldr/3` and `map/2` have been imported from the `lists` module.)

   A. `mapInside(F,Llt) -> foldr(fun(Ls, Res) -> map(F,Ls) ++ Res end, [], Llt).`

   B.
   ```
   mapInside(F,Llt) ->
       foldr(fun(Ls, Res) -> foldr(fun(E,R) -> [[F(E)]|Res] end end, [], Llt).
   ```

   C. `mapInside(F,Llt) -> foldr(fun(Ls, Res) -> [[map(F,Ls)]|Res] end, [], Llt).`

   D. `mapInside(F, Llt) -> foldr(fun(Ls, Res) -> [[map(F,Ls)],[Res]] end, [], Llt).`

   E. `mapInside(F, Llt) -> foldr(fun(Ls, Res) -> [map(F,Ls)|Res] end, [], Llt).`

   F. `mapInside(F, Llt) -> foldr(fun(Ls, Res) -> [Res|map(F,Ls)] end, [], Llt).`

   G. None of the above is correct.

2. [Concepts] This problem is about free and bound identifier occurrences. We define an *identifer* in Erlang as either a function name or a variable name. That is, in Erlang we consider function names to be identifiers, even though they syntactically look like atoms. Variable names are also considered to be identifiers. However, atoms that are not function names are not identifiers (and neither are reserved words nor numeric literals).

   Consider the following Erlang expression.

   ```
   H(fun(X,Y) -> K(foo(bar({the_fun, fun(A,H) -> minus(A,4020) end}))) end, K)
   ```

   (a) (8 points) In set brackets ({ and }), list the complete set of all identifiers that occur free in the above expression.

   (b) (4 points) In set brackets ({ and }), list the complete set of all identifiers that occur bound in the above expression.

3. [Concepts] This is another problem about free and bound identifier occurrences.

   Consider the following Erlang expression.

   ```
   P(foo(Q(map(fun (E) -> bar(P, Q, P(Q(E)), F) end,
               [zero, false, 1, fun(F,G) -> fun(X) -> F(F(X)) end end]))))
   ```

   (a) (12 points) In set brackets ({ and }), list the complete set of all identifiers that occur free in the above expression.

   (b) (4 points) In set brackets ({ and }), list the complete set of all identifiers that occur bound in the above expression.

4. [Concepts] This problem is about scope rules.

   (a) (5 points) Suppose a programming language has a way to create function values at runtime, like Erlang does. Explain how the semantics of function values can tell you about whether that language has static or dynamic scoping for variable names?

   (b) (5 points) If a language makes closures for function values that are created at runtime, then with respect to the closure expression itself, what kind of identifiers that occur within the text of the function closure expression's code does it need to remember values for: free identifiers or bound identifiers?

   (c) (5 points) Is static scoping more useful for variables than dynamic scoping? Explain why (or why not).

5. (5 points) [MapToLanguages] How are exception handlers in Java (or C# or C++) scoped (statically or dynamically)?

6. [Concepts] This problem is about type checking.

   (a) (5 points) Give a brief example of an expression in Erlang that generates a type error.

   (b) (5 points) Which kind of type checking allows the programmer more flexibility: static or dynamic type checking?

   (c) (5 points) Give an example, in Erlang, of an expression or program that will run without a type error that would not type check if it were translated into Haskell.

   (d) (5 points) In Erlang, does the representation of every value need to be encoded in such a way that the runtime system can tell what its type is during each program's execution?

   (e) (5 points) In Haskell, does the representation of every value need to be encoded in such a way that the runtime system can tell what its type is during program execution?

7. (5 points) [UseModels] [Concepts] What kind of problems can be correctly solved using a stateless server in Erlang? (Briefly explain.)
8. [Concepts] This question is about modules, data abstraction, and information hiding.
   (a) (5 points) Briefly describe how one could implement an abstract data type, such as a stack of items, in Erlang in such a way that the internal implementation details of the stack are completely hidden from client code, so that clients cannot even read or print the internal representation of the type (e.g., the stack)? (You don't have to provide code, but you can show what it would look like if you wish.)
   (b) (5 points) How is representing an abstract data type, such as a stack of items, different in Haskell than in Erlang?
   (c) (5 points) Is data abstraction (the hiding of implementation details) enforced in Java (or C#) in a way that is more like Haskell or more like Erlang? Choose either Haskell or Erlang, and give a brief explanation of why
9. (5 points) Briefly describe some syntactic form (e.g., an expression) in Erlang that is an example of a syntactic sugar, and say what other expression it translates into.

Points

This homework's total points: 108.