Functional Programming

https://proglang.informatik.uni-freiburg.de/teaching/functional-programming/2021/

Exam Sheet

2021-09-06

How to do this exam

- You can earn 50 points from the questions on this sheet.
- You can earn 230 points from the programming task (plus bonus points).
- You need at least 100 points to pass the exam and you must do at least one question from this sheet. Otherwise, you are free to choose the questions from the sheet or the programming task.

Question 1 (SKI Combinators, 10 points)
Submit as a single text file `Question1.txt`.

SKI combinator calculus is a combinator logic that comprises three combinators **S**, **K** and **I**. The introduction on it is available on the wiki page

https://en.wikipedia.org/wiki/SKI_combinator_calculus

All the terms in lambda calculus can be translated into the SKI calculus by using *abstraction elimination* which is available on the page

https://en.wikipedia.org/wiki/Combinatory_logic#Completeness_of_the_S-K_basis

Translate the following four lambda terms into the SKI calculus using abstraction elimination.

1. $\lambda f.(\lambda x.\ f(x\ x))(\lambda x.\ f(x\ x))$
2. $\lambda m.\lambda n.\lambda f.\lambda x.\ m\ (n\ f)\ x$
3. $\lambda x.\ x\ (\lambda y.\ y\ x)$
4. $\lambda x.\lambda y.\ x\ (y\ y)$

Question 2 (Polymorphic Types, 20 points)
Submit as a single, working Haskell program file `Question2.hs`. Put explanations in comments.

Write Haskell terms according to the polymorphic type signatures given below. As the types are polymorphic, there can be more than one Haskell term with the same type, but different behavior (they return different results when applied to the same arguments). For each of the following polymorphic types, write the required number of Haskell terms for each type. All function should be total. Do not use `undefined` or `error`.

1. Write four different terms with the type

$$(a, a) \to (a, a)$$

2. Write two different terms with the type

$$[b] \to \texttt{Maybe}\ b$$

3. Write one term with the type

$$\texttt{Maybe}\ a \to (a \to \texttt{Maybe}\ b) \to \texttt{Maybe}\ b$$

4. Write one term with the type

$$(a \to b) \to \texttt{Either}\ c\ a \to \texttt{Either}\ c\ b$$

5. Write two different terms with the type

$$\texttt{Maybe}\ (\texttt{Either}\ a\ b) \to \texttt{Either}\ a\ (\texttt{Maybe}\ b)$$

6. Given the context `Functor f`, write one term with type

$$f\ a \to f\ [a]$$

Question 3 (Algebraic Datatypes, 20 points)

Submit as a single, working Haskell program file `Question3.hs`. Put explanations in comments.

Design three algebraic data types according to the following descriptions.

1. Define an algebraic data type `Currency` for currencies of five countries. You should use the official abbreviation of currencies for the constructor names.

   Write a function `nextCurrency :: String -> Currency` that reads the name of a currency from the string and returns the next currency according to the sequence in the type definition.

2. Define an algebraic data type `Point2D` for 2D points given either in Cartesian coordinates or in polar coordinates.

   Write a function `size :: Point2D -> Double` that calculates the length of the vector from $(0, 0)$ to the point given.

3. The syntax of Propositional logic is defined as follows.

   Definition 1 (Propositional Logic)**.** *Well-formed formulae of propositional logic are defined by the following grammar, where $a \in Atom$ stands for an atomic formula:*

   $$\alpha, \beta ::= a \mid \alpha \wedge \beta \mid \neg\alpha \mid \mathit{false}$$

   Define an algebraic data type `Prop atom` for propositional logic according to its definition that takes the type of atoms as a parameter.

   Write a function `holds :: Eq atom => Prop atom -> [atom] -> Bool` such that `holds alpha trueAtoms` is `True` iff the formula `alpha` is true under the assumption that all elements of `trueAtoms` are true (and any other atom is false).