Test

This test has 4 questions and pages numbered 1 through 5. This test is worth 250 points.

Reminders

This test is closed book and notes. If you need more space, use the back of a page. Note when you do that on the front.

This test is timed. We will not grade your test if you try to take more than the time allowed. Therefore, before you begin, please take a moment to look over the entire test so that you can budget your time.

For programs, indentation (part of the Haskell syntax) is important to us for “clarity” points; if your code is sloppy or hard to read, you will lose points. Correct syntax also matters. Check your code over for syntax errors. You will lose points if your code has syntax errors.
Problem 1

Data Types

1. An environment is a set of mappings from names (strings) to values (of some type a). Define a polymorphic data type Environment, which has two type constructors Empty, the empty environment and Entry, the environment that defines at least one mapping. (40 points)

2. Define data type Environment to be an instance of type class Show. (40 points)

3. Define data type Environment to be an instance of type class Eq. (Assume that both environments have been ordered the same way.) (50 points)

Answer:

1. data Environment a = Empty
   | Entry String a (Environment a)

2. instance Show a => Show (Environment a) where
   show Empty = "[]"
   show (Entry s v l) = "(" ++ s ++ "," ++ show v ++ ")":" show l

3. instance Eq a => Eq (Environment a) where
   Empty == Empty = True
   (Entry s1 v1 l1) == (Entry s2 v2 l2) = s1 == s2 && v1 == v2 && l1 == l2
   _ == _ = False
Model the data type \texttt{triple}(10 points) and the following operations in the \(\lambda\)-calculus:

1. \texttt{first} (10 points)
2. \texttt{second} (10 points)
3. \texttt{third} (10 points)

Answer:

1. \(\texttt{triple} \equiv \lambda x \ y \ z \ t . \ t \ x \ y \ z\)
2. \(\texttt{first} \equiv \lambda t . \ t \ (\lambda x \ y \ z . \ x)\)
3. \(\texttt{second} \equiv \lambda t . \ t \ (\lambda x \ y \ z . \ y)\)
4. \(\texttt{third} \equiv \lambda t . \ t \ (\lambda x \ y \ z . \ z)\)
Problem 3

\(\lambda\)-Calculus

Reduce the following lambda-terms using applicative-order-reduction strategy:

\[
(\lambda \ y \ . \ y)((\lambda \ x \ . \ (x \ x))(\lambda \ x \ . \ (x \ x)))
\]

(20 points)

\[
\lambda \ y \ . ((\lambda \ y \ . \ y)(\lambda \ x \ y . \ x)\lambda \ x \ . \ x)y
\]

(30 points)

Answer:

\[
(\lambda \ y \ . \ y)((\lambda \ x \ . \ (x \ x))(\lambda \ x \ . \ (x \ x)))
\]

\[\rightarrow\]

\[
(\lambda \ y \ . \ y)(\lambda \ x \ . \ (x \ x))(\lambda \ x \ . \ (x \ x))
\]

no normal form

\[
\lambda \ y \ . ((\lambda \ y \ . \ y)(\lambda \ x \ y . \ x)\lambda \ x \ . \ x) y
\]

\[\rightarrow\]

\[
\lambda \ y \ . ((\lambda \ x \ y . \ x)\lambda \ x \ . \ x) y
\]

\[\rightarrow\]

\[
\lambda \ y \ . (\lambda \ x \ . \ x) y
\]

\[\equiv\]

\[
\lambda \ y \ . \ \lambda \ x \ . \ x
\]
Problem 4

The following term has a normal form:

$$(\lambda x . ((\lambda y . x (y y))((\lambda y . x (y y))))(\lambda x . (\lambda x y . x)))$$

Find the normal form by simplifying the term using any suitable reduction strategy.

(30 points)

Answer:

$$(\lambda x . (\lambda y . x (y y))((\lambda y . x (y y))))(\lambda x . (\lambda x y . x))$$

$\to (\lambda y . (((\lambda x y . x) (\lambda x y . x)) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x))))$$

$\to (\lambda x . (\lambda x y . x)) (((\lambda y . ((\lambda x y . x) (\lambda x y . x))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x))))$$

$\to (\lambda x . (\lambda x y . x) (((\lambda y . ((\lambda x y . x) (\lambda x y . x))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x))))$$

$\to (\lambda x . (\lambda x y . x)) (((\lambda y . ((\lambda x y . x) (\lambda x y . x))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x))))$$

$\to (\lambda x . (\lambda x y . x)) (((\lambda y . ((\lambda x y . x) (\lambda x y . x))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x)))) (\lambda y . ((\lambda x y . x) (\lambda x y . x))))$$

$\to \lambda x y . x$