This exam is worth 20% of the course. There are five questions and a total of 100 points available:

1. (30 points) Explain your answers!

(a) (3 points) What does it mean for two \(\lambda\)-terms to be \(\alpha\)-equivalent? Is it possible to decide whether two terms are \(\alpha\)-equivalent?

(b) (3 points) What does it mean for two \(\lambda\)-terms to be \(\beta\)-equivalent? Is it possible to decide whether two terms are \(\beta\)-equivalent?

(c) (3 points) Write \(\lambda xy. (\lambda y.xy)(\lambda x.xy)\) in de Bruijn notation.

(d) (3 points) When is a term in \(\beta\)-normal form? Is the term in (c), above, in \(\beta\)-normal form? Provide two examples of terms which do not have a \(\beta\)-normal form.

(e) (3 points) Explain what it means to say that \(\beta\)-reduction is confluent. Why does each \(\lambda\)-term have at most one normal form?

(f) (12 points) Demonstrate leftmost outermost \(\beta\)-reductions on the following \(\lambda\)-terms:

(i) \((\lambda z x. z (xz))(\lambda y.xy)(\lambda x.xx)\)

(ii) \((\lambda xy.xy)(\lambda x. z(yx))\)

(iii) \((\lambda xy.xy(xx))(\lambda x.y)(\lambda x.xx)(\lambda x.xx)\)

(g) (3 points) What is an advantage of a leftmost outermost reduction strategy over a by-value reduction strategy?
2. (15 points) Consider the “printer” monad as defined by:

```haskell
data Printer a s = Print [a] s

instance Monad (Printer a) where
  return s = Print [] s
  (Print as s) >>= f = case f s of Print bs s' -> Print (as++bs) s'

pput str = Print str ()
```

and the data for arithmetic functions:

```haskell
data AExp a = Add (AExp a) (AExp a)
  | Mul (AExp a) (AExp a)
  | Num a
```

Recall that the translation from “do” syntax to core Haskell is given by:

\[
\begin{align*}
  [\text{do }\{e\}] & = e \\
  [\text{do }\{x \leftarrow t; r\}] & = t \triangleright\triangleright= q \text{ where } \\
  q x & = [\text{do }\{r\}] \\
  [\text{do }\{p; r\}] & = p \triangleright\triangleright= \lambda x. [\text{do }\{r\}]
\end{align*}
\]

Consider the following function:

```haskell
aprint :: Aexp Int -> Printer Char Int
aprint (Num n) = do pput (show n)
  return n
aprint (Add t1 t2) = do pput "(
  n1 <- aprint t1
  pput "+
  n2 <- aprint t2
  pput ")"
  return (n1+n2)
aprint (Mul t1 t2) = do pput "(
  n1 <- aprint t1
  pput "*
  n2 <- aprint t2
  pput ")"
  return (n1*n2)
```

Explain, briefly, what this function does. Translate the first two cases of the function into core Haskell leaving the sequencing operator untouched. On the first of these, demonstrate carefully how to remove the sequencing and return operators.
3. (15 points) Using the datatype \texttt{AExp \textit{a}} of the previous question:

(a) (5 points) Define the fold function in Haskell for arithmetic expressions.

(b) (10 points) Rewrite the function \texttt{aprint} of the previous question as a fold over the tree of arithmetic expressions to have type:

\[
\texttt{aprint'} :: \texttt{Aexp Int} \rightarrow ([\texttt{Char}], \text{Int})
\]

4. (35 points)

(a) (5 points) Explain how “triples” are represented in the lambda calculus. What are the definitions of the \textit{three} projection functions?

(b) (10 points) How do you represent the trees of

\[
data \texttt{Tree a b} = \texttt{Leaf a} \\
| \texttt{Node b (Tree a) (Tree a)}
\]

in the \(\lambda\)-calculus?

What are the \(\lambda\)-terms for the constructors, the fold, the map function (this takes in two functions), and the case combinator for trees?

(c) (5 points) Explain what a fixed point combinator is. Prove that

\[
\texttt{Y} := \Theta\Theta \quad \text{where} \quad \Theta := \lambda f.f(xx_f)
\]

is a fixed point combinator.

(d) (10 points) Explain how the recursive \texttt{factorial} function

\[
\texttt{factorial n} = \texttt{if} \ (\texttt{iszero n}) \ \texttt{then} \ (\texttt{succ zero}) \\
| \ \texttt{else n * (factorial (pred n))}
\]

is programmed in the \(\lambda\)-calculus (you may assume the \texttt{if} combinator and the arithmetic functions).

(e) (5 points) Explain briefly why all computable functions can be represented in the \(\lambda\)-calculus.

5. (5 points!)

(a) What was Turing’s first name?

(b) What is the Turing award? Name two Turing award recipients.

(c) Can you name a graduate student of Alonso Church? Is Church still alive?

(d) Can you name a graduate student of Haskell Curry? Is Curry still alive?

(e) The sentence: “If this sentence is true then the temperature today in Calgary is 35°C” is an example of Curry’s paradox. Explain why it is a paradox. What does it have to do with the \(\lambda\)-calculus?