COMP6463: Untyped λ-Calculus
Sample Solution to the Assignment

1. Write down four rules for *strict* or *applicative*, left-to-right order evaluation of λ-calculus terms in the style of the rules given for normal order evaluation in the lectures. In other words, define a binary relation $\rightarrow_a$ that ensures that all functions and arguments are evaluated before substitutions are performed. Here's one rule for free:

$$\frac{N \rightarrow_a N' \quad M \text{ is in } \beta\text{-normal form}}{M\ N \rightarrow_a M\ N'}$$

This rule guarantees that evaluation moves from left to right because N is not allowed to be evaluated before M has been reduced to normal form. You need to write three more rules. **[5 marks]**

The remaining rules are

$$\frac{M \rightarrow_a M'}{(\lambda v.\, M) \rightarrow_a (\lambda v.\, M')} \qquad \frac{M \rightarrow_a M'}{M\ N \rightarrow_a M'\ N}$$

$$\frac{M \text{ and } N \text{ are in } \beta\text{-normal form}}{(\lambda v.\, M)\ N \rightarrow_a M[v := N]}$$

2. Prove the following "inequalities" by giving a derivation of $\boldsymbol{\lambda} \vdash X = Y$ for arbitrary terms X and Y when the inequality is assumed to be an equation:

   (a) $\mathsf{K}\ \mathsf{I}\ \#\ \mathsf{K}$ **[2 marks]**

   One chain of reasoning is as follows:

Assume	$\mathsf{K}\ \mathsf{I} = \mathsf{K}$	
then	$\mathsf{K}\ \mathsf{I}\ X\ Y = \mathsf{K}\ X\ Y$	
then	$\mathsf{I}\ Y = X$	(by behaviour of K)
then	$Y = X$	(by behaviour of I; done)

(b) $x \;\#\; y$ (with x and y distinct λ-calculus variables) **[3 marks]**

One chain of reasoning is as follows:

$$\begin{array}{lrcll}
\text{Assume} & x & = & y & \\
\quad\text{then} & (\lambda x.\, x) & = & (\lambda x.\, y) & \text{(rule for equality of abstractions)} \\
\quad\text{then} & (\lambda x.\, x)\ X & = & (\lambda x.\, y)\ X & \\
\quad\text{then} & X & = & y & \text{(by } \beta\text{-reduction)}
\end{array}$$

Analogously, Y must equal y too, so by transitivity of equality $X = Y$.

3. Define the λ-term corresponding to the following recursive function f *without* using the Y (or any other recursion) combinator:

$$\begin{array}{rcl}
f(0) & = & 3 \\
f(n+1) & = & 2 \times f(n) + 3
\end{array}$$

Use primitive recursion and the λ-terms corresponding to 2, 3, $+$ and $\times$. *Don't* solve the recurrence relation and define the function with a closed form using exponentiation. **[5 marks]**

The answer is $(\lambda n.\, n\ (\lambda m.\ +\ \ (\times\ 2\ m)\ 3)\ 3)$

Many people thought that answers involving things like $\mathsf{Pr}\langle f, g\rangle$ were correct. But this is a construction for creating a (primitive) recursive function over the natural numbers, not a λ-term, which is what the question asks for.

4. Again, using primitive recursion, define a function on lists that adds 1 to each element of a list. Thus:

$$\begin{array}{rcl}
f([]) & = & [] \\
f(h :: t) & = & (h+1) :: f(t)
\end{array}$$

Recall that $h :: t$ is the list consisting of the element h followed by the list t (a "cons" cell). **[5 marks]**

The answer is $(\lambda \ell.\, \ell\ (\lambda h.\, \mathsf{cons}\ (+\ 1\ h))\ \mathsf{nil})$

The middle abstraction might be (η-)expanded to $(\lambda h\, t.\, \mathsf{cons}\ (+\ 1\ h)\ t)$