Types

(a) Give the axioms and rules for inductively generating ML typing judgements \( \Gamma \vdash M : \tau \), where \( \Gamma \) is a finite function from type variables to type schemes and \( M \) ranges over expressions built up from variables using only function abstraction \( (\lambda x(M)) \), function application \( (M_1 M_2) \) and local declarations \( \text{let } x = M_1 \text{ in } M_2 \). As part of your answer you should explain what it means for a type scheme to generalise a type. [7 marks]

(b) Consider the fixpoint combinator \( Y \), which is defined to be the expression \( \lambda x((\lambda y(x(y y)))\lambda y(x(y y))) \). State, with justification, whether there is a type \( \tau \) for which \( \emptyset \vdash Y : \tau \) is provable from the axioms and rules in part (a). [6 marks]

(c) Consider adding to the ML type system a ‘universal’ type \( \omega \) together with the axiom

\[
\text{(univ)} \quad \Gamma \vdash M : \omega
\]

asserting that any expression \( M \) has type \( \omega \). In this augmented type system show that \( x : \omega \rightarrow \alpha \vdash \lambda y(x(y y)) : (\omega \rightarrow \omega) \rightarrow \alpha \) is provable, where \( \alpha \) is any type variable. [3 marks]

Deduce that \( \emptyset \vdash Y : (\omega \rightarrow \alpha) \rightarrow \alpha \) is also provable, where \( Y \) is the expression in part (b). [4 marks]