13 Types (nk480)

(a) Consider the OCaml option type

\[
\text{type } 'a \text{ option } = \text{None | Some of } 'a
\]

In this question we will look at its encoding in System F.

(i) For a fixed \( A \), give a suitable System F type for a Church encoding of the \( A \text{ option} \) type.\[1 \text{ mark}\]

(ii) Give an implementation of the Some and None constructors for this encoding.\[2 \text{ marks}\]

(iii) Give a type and encoding of an eliminator named case for the option type.\[2 \text{ marks}\]

(iv) Give the reduction rules for case, and show that your encoding models them correctly.\[5 \text{ marks}\]

(b) All of the questions in this part are about the monadic lambda calculus.

(i) Give a well-typed term of type \( T(T(A)) \rightarrow T(A) \), and explain briefly in prose what this function does.\[2 \text{ marks}\]

(ii) Give a well-typed term of type \( T(A) \rightarrow (A \rightarrow T(B)) \rightarrow T(B) \), and explain briefly in prose what this function does.\[2 \text{ marks}\]

(iii) Give a type and definition of a monadically-typed fixed point operator suitable for defining recursive functions on integers.\[6 \text{ marks}\]