15 Types (nk480)

(a) In a simply-typed lambda calculus augmented with first-class continuations, booleans, a list type and its iterator (i.e., fold, but not full recursion), write a function

\[ \text{every} : (X \to \text{Bool}) \to \text{List} X \to \text{Bool} \]

such that \( \text{every} \ p \ x s \) returns \text{true} if every element of \( x s \) satisfies \( p \), and \text{false} otherwise. This function should also stop iterating over the list as soon as it finds a false element. You may use SML- or OCaml-style notation if desired, but explain any notation used beyond the basic lambda calculus.

[4 marks]

(b) In the monadic lambda calculus with state, suppose we change the typing rule for reading locations to not cause a monadic effect: If we suggest changing the monadic lambda calculus to permit treating reads as pure:

\[ l : X \in \Sigma \]
\[ \Sigma; \Gamma \vdash !l : X \]

(i) Is this rule still typesafe? Informally but carefully justify your answer.

[2 marks]

(ii) Is the following \text{common subexpression elimination} transformation sound? Either give an argument why it is, or supply a counterexample and explain why it shows it is not.

\[ \text{let } x = \text{return } e_1; \quad \text{let } x = \text{return } e_1; \]
\[ \text{let } y = e_2; \quad =====> \quad \text{let } y = e_2; \]
\[ \text{let } z = \text{return } e_1; \quad [z/x]e_3 \]
\[ e_3 \]

[6 marks]

(c) In System F augmented with existential types, give an existential type for the interface of the natural numbers, and give an implementation for it.

[8 marks]