Types

Briefly explain what is meant by *capture-avoiding substitution.* [3 marks]

What is a *principal typing* and why is it useful? [5 marks]

Suppose that a constant fix is added to the expressions of System F, with the typing rule

\[ \Gamma 
\vdash \text{fix} \in \text{All}(X) (X \to X) \to X \]  

(TM-Fix)

and principal evaluation rule:

\[
\frac{f \ (\text{fix} \ X \ f) \Downarrow r}{\text{fix} \ X \ f \Downarrow r}
\]  

(EM-Fix)

Also, suppose we are given a built-in type operator *List* and the following expression constants:

\[
\begin{align*}
nil &\in \text{All}(X) \ (\text{List} \ X) \\
\text{cons} &\in \text{All}(X) \ X \to (\text{List} \ X) \to (\text{List} \ X) \\
\text{car} &\in \text{All}(X) \ (\text{List} \ X) \to X \\
\text{cdr} &\in \text{All}(X) \ (\text{List} \ X) \to (\text{List} \ X) \\
null &\in \text{All}(X) \ (\text{List} \ X) \to \text{Bool}
\end{align*}
\]

Use these primitives to write a polymorphic function *fold* of type

\[
\text{fold} \in \text{All}(X) \ \text{All}(Y) \ (X \to Y \to Y) \to Y \to (\text{List} \ X) \to Y
\]

that “folds a function across a list.” For example, applying *fold* to +, 0, and a list of numbers should return the sum of the list. [8 marks]

Which of the following existential packages is most useful, and why?

\[
\begin{align*}
&[\text{Int}, \{x = 5, \ f = \text{fun}(i \in \text{Int}) \ i + 1\}] \in \text{Some}(X) \ \{x \in X, \ f \in X \to \text{Int}\} \\
&[\text{Int}, \{x = 5, \ f = \text{fun}(i \in \text{Int}) \ i + 1\}] \in \text{Some}(X) \ \{x \in \text{Int}, \ f \in X \to \text{Int}\} \\
&[\text{Int}, \{x = 5, \ f = \text{fun}(i \in \text{Int}) \ i + 1\}] \in \text{Some}(X) \ \{x \in \text{Int}, \ f \in X \to \text{Int}\} \\
&[\text{Int}, \{x = 5, \ f = \text{fun}(i \in \text{Int}) \ i + 1\}] \in \text{Some}(X) \ \{x \in \text{Int}, \ f \in \text{Int} \to \text{Int}\}
\end{align*}
\]

[4 marks]