1. Analysis of Definition (4 + 1 + 3 pts)

The Fibonacci function \( F(n) \) defined for natural numbers \( n \) is as follows.

\[
F(n) = \begin{cases} 
0 & \text{if } n = 0 \\
1 & \text{if } n = 1 \\
F(n-1) + F(n-2) & \text{otherwise}
\end{cases}
\]

Define an SML-function \( \text{cntCalls} \) that computes the number of calls to \( F \) generated for various input values. (E.g., \( \text{cntCalls}(0) = 1 \), \( \text{cntCalls}(3) = 5 \), etc.) What are the values of \( F(5) \) and \( \text{cntCalls}(5) \)?

2. Understanding Function Definition (2 + 3 + 3 pts)

\[
\text{fun } f \text{ [] } = \text{ [];}
| f (x::xs) = \text{ let val s = } f \text{ xs}
\text{ in (map (fn y => y@[x]) s) }
\text{ end;}
\]

1. What is the signature of \( f \)?

2. Informally describe the list function computed by \( f \). Give the value and type returned for \( f \ ["a"] \)?


3. Writing Function Definition (10 pts)

The following SML-definition specifies two concrete datatypes \( \text{etype} \) and \( \text{expr} \).

\[
\begin{align*}
\text{datatype etype} & = \text{Int} | \text{Real}; \\
\text{datatype expr} & = \text{I} | \text{J} | \text{A} | \text{B} \\
& \quad | \text{Plus of (expr * expr)} \\
& \quad | \text{Mul of (expr * expr)};
\end{align*}
\]

The type of \( I \) and \( J \) is \( \text{Int} \). The type of \( A \) and \( B \) is \( \text{Real} \). The type of a \( \text{Plus} \)-expression is \( \text{Int} \), if the subexpressions have \( \text{Int} \) type; otherwise it is an error. (E.g., if both arguments are \( \text{Real} \), it is an error!) The type of a \( \text{Mul} \)-expression is always \( \text{Real} \) as long as the subexpressions are well-typed; otherwise it is an error. (Note, a subexpression is well-typed if its type either \( \text{Int} \) or \( \text{Real} \).)

Write an SML-function \( \text{typeInfer} \) that infers the type of an expression wherever feasible (according to the specified notion of type inference) and throws an exception called \( \text{Error} \) otherwise. For example,

- \( \text{typeInfer I; val it = Int : etype} \)
- \( \text{typeInfer (Mul (A,Plus(J,I))); val it = Real : etype} \)
- \( \text{uncaught exception Error ...} \)

4. Functional Languages (4 pts)

Explain any two important trends in modern programming language design and illustrate your claim with sound examples from the languages presented/discussed in class.