

**Final Exam (40 pts)**

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

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

$$\begin{aligned} F(n) &= 0 && \text{if } n = 0 \\ &= 1 && \text{if } n = 1 \\ &= F(n-1) + F(n-2) && \text{otherwise} \end{aligned}$$

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

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

```
fun f [] = []
  | f (x::xs) = let val s = f xs
                in (map (fn y => y@[x]) s)
                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. Now formalize what  $f$  does using the Induction Principle.

**3. Writing Function Definition (10 pts)**

The following SML-definition specifies two concrete datatypes `etype` and `expr`.

```
datatype etype = Int | Real;
datatype expr = I | J | A | B
              | Plus of (expr * expr)
              | Mul of (expr * expr);
```

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

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

```
- typeInfer I;
val it = Int : etype
- typeInfer (Mul (A,Plus(J,I)));
val it = Real : etype
- typeInfer (Mul (A,Plus(B,J)));
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.