**Indian Institute of Technology Delhi**
**Department of Computer Science and Engineering**

**CSL665** **Introduction to Logic and Functional Programming** **Major Exam**
**November 29, 2006** **13:00–15:00** **Maximum Marks: 100**

*Instructions: Write your name and entry number at the top of each sheet. Write your answers in the space provided. Use only a blue or black pen. Budget your time and space carefully.*

Q0 (12 marks) **Programming with lists in Prolog.** Suppose we represent the polynomial $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ as a list $[a_0, a_1, a_2, \ldots, a_n]$ of its coefficients. Write a **Prolog program** `addpoly(L1,L2,L3)` for adding two polynomials, $a_0 + \ldots + a_nx^n$ represented as `L1` and $b_0 + \ldots + b_mx^m$ represented as `L2`, returning the result as list `L3`. (Note $m$ and $n$ may not be equal). Example: `addpoly([2,5,4],[3,5,7,1],L3)` yields `L3=[5,10,11,1]`.

Q1 (12 marks) **Programming with lists in ML.** Assuming we still represent polynomials as lists of integer co-efficients (as in Q0), write a **functional program** `ddx` in ML that takes a polynomial and returns the polynomial that is its first differential. For example `ddx [2,5,3]` returns `[5,6]`. Recall that

$$\frac{d}{dx}(a_0 + a_1x + a_2x^2 + \ldots + a_nx^n) = a_1 + 2.a_2x + \ldots + n.a_nx^{n-1}$$

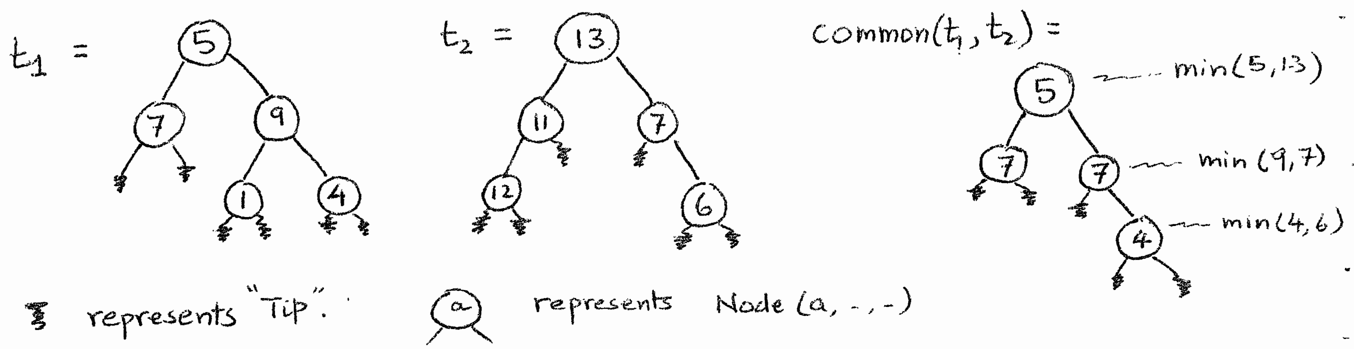
```
      int          int
(* ddx: ~~real~~ list -> ~~real~~ list *)

fun ddx
```

Q2 (16 marks) . Suppose we have the following ML datatype for labelled binary trees:

```
datatype 'a bintree = Tip | Node of 'a * ('a bintree) * ('a bintree)
```

Assume the existence of a function `min` that takes a pair of integers and returns the lower of the two. Write a functional program (in ML) `common` that takes two int-labelled binary trees $t_1$ and $t_2$ and returns a labelled binary tree that is the "common part" of both $t_1$ and $t_2$, with the nodes labelled by the minimum of the labels of both nodes. That is, for example if




```
(* common: int bintree -> int bintree -> int bintree *)

fun common

  | common
```

Q3 (10+10 marks) **Derived Rules in ND** Consider the following inference rule (for first-order logic), where $\Gamma$ and $\Delta$ are sets of formulas:

$$\frac{\Gamma \vdash \varphi \qquad \Delta, \varphi \vdash \psi}{\Gamma \cup \Delta \vdash \psi}$$

(a) Show that the rule is *sound* in first-order logic.

(b) A rule is called "derived" if it can be expressed as a combination of previous rules. Show that the given rule can be derived using the rules of Natural Deduction. That is, given a ND proof $\pi_1$ of $\Gamma \vdash \varphi$ and a ND proof $\pi_2$ of $\Delta, \varphi \vdash \psi$, one can construct (by modification F1, and using other rules such as $(\supset E)$, a proof $\pi'$ of $\Gamma \cup \Delta \vdash \psi$.

Q4 (15 marks) **Polymorphic Type Checking in ML.** Consider the following ML-like program that has type `int` (under no assumptions other than 3 has type `int`):

$$\textbf{let } id = \lambda x.x \textbf{ in } ((id\ id)\ 3) \textbf{ end}$$

Give the type-checking proof tree to show that this expression has type `int`, showing in particular that *id* can be applied to itself, where the first copy has type $(\texttt{int} \to \texttt{int}) \to (\texttt{int} \to \texttt{int})$ and the second $\texttt{int} \to \texttt{int}$.