CS 153	Foundations of Software	Homework 8
Fall 2006		Due: Nov 29, 2006

1. [20] For the propositional calculus proof system given by the sequent-style natural deduction inference rules (Handout 2, all the rules that don't involve quantifiers), prove the following *Soundness Theorem* by structural induction on proofs.

Theorem: $\Gamma \vdash A \;\; \Rightarrow \Gamma \models A$.

2. [35] Define an SML datatype that represents natural deduction style proofs for the propositional calculus using the same deduction system as in the previous question. You will build this on the type for formulas (wffs) from wff-mod.sml, and you'll also need a type representing judgements ($\Gamma \vdash A$) where Γ is a set or list of wffs and A is a wff.

Do all values of your proof datatype represent correct proofs according to the inference rules? If not, write a function

```
val check : proof -> bool
```

that checks whether a proof is correct.

Use this datatype of proofs to implement a simple theorem prover that is clever enough to construct a proof of a simple formula like $A \rightarrow (A \vee B)$. I.e., you will define a function

```
val prove : wff -> proof
```

Use the check function to verify the correctness of the proof produced, if necessary.

3. [5] We can enrich the pure lambda calculus a bit by adding integer constants and primitive operations like $+$ and $-$, allowing terms like $\lambda x.x+2$. The expressions of this enriched calculus are defined by the following grammar:

$$e ::= n \mid x \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1\, e_2 \mid \lambda x.e$$

where n ranges over integer constants and x ranges over a set of variables. The juxtaposition of two expressions, $e_1 e_2$, denotes the application of a function (e_1) to an argument (e_2), and function application associates to the left, so that $e_1 e_2 e_3$ is read as $(e_1 e_2) e_3$. In a lambda abstraction $\lambda x.e$, the variable x is bound in its scope e. In this enriched calculus a *value* is either a lambda abstraction or an integer constant.

Let e be the lambda term given below. What variables appear free in e, and what variables appear bound? Mark bound variable uses with an underline, and free variables with an overline, and draw a line connecting each use of a bound variable with the corresponding binding occurrence of that variable. Note carefully the structure of the expression determined by the parentheses.

$$e = (\lambda g\,.\,g\;(f\;2))\;((\lambda y\,.\,\lambda z\,.\,\lambda x\,.\,z\;(\lambda u\,.\,u+x))\;x\;(\lambda f\,.\,f\;y))$$

4. [5] Perform the substitution specified by:

$$\lambda y.x(\lambda x.(xz)y)\ [\lambda u.xy\,/\,z]$$

Make sure that you avoid free variable captures.

Note: A couple additional questions may be added later.