Fall 2008
B522 — Programming Language Foundations
Midterm exam (30% of final grade)

Name (please print): ....

Username: ....

1	Short Questions: Calculi	20 pts	
2	Short Questions: Types	30 pts	
3	Calculi	30 pts	
4	Types	30 pts	
Total		100 pts	

1 Short Questions: Calculi

- Consider the following rewriting system:

$$\begin{array}{lcl} B & \rightarrow & 2 \\ B & \rightarrow & C \\ C & \rightarrow & B \\ C & \rightarrow & 3 \end{array}$$

  and the standard equivalence relation = defined over it.

  - Write the statement of the Church-Rosser theorem.
  - Give a counterexample to the Church-Rosser theorem.

- Consider the λ-calculus extended with constants (numbers, addition, and so on). Show that the observational equivalence relations for the call-by-value λ-calculus and for the call-by-name λ-calculus are different and that neither is included in the other. **Hint:** Give one call-by-value equivalence that is not a call-by-name equivalence and vice-versa.

2 Short Questions: Types

- Consider two expressions e_1 and e_2 that have the same type in the same type environment. A program p with e_1 as one of its subexpressions typechecks. Will the same program with e_1 replaced by e_2 typecheck? Your answer will obviously depend on what assumptions you make about the expressions and the type system: try to state these assumptions clearly.

- At the end of the exam, you will find a page from a recently published paper.[1] In the figure (Fig. 6), how is the type environment represented? (e.g., as a set, as a multiset, as a sequence, etc.). Justify your answer with a one-line explanation.

- We know that the preservation lemma holds for the simply typed call-by-value λ-calculus. In other words, if $\Gamma \vdash e : t$ and $e \rightarrow e'$ using a β_v-reduction, then $\Gamma \vdash e' : t$. Would the same lemma hold if we reversed the direction of the β_v reduction, i.e., if we used the following reduction:

$$e[v/x] \quad \rightarrow \quad (\lambda x.e)v$$

[1] p.7 of: David Walker, A type system for expressive security policies, POPL 2000.

3 Calculi

Consider the following small functional language with imperative extensions:

$$\begin{array}{llll} \text{(Programs)} & p & ::= & \textbf{ref}\ \underline{n}\ e \\ \text{(Expressions)} & e & ::= & x \mid \lambda x.e \mid ee \mid \underline{n} \mid e+e \mid \mathsf{inc} \mid \mathsf{read} \end{array}$$

The semantics of the language is defined using the following reduction relation:

$$\begin{array}{rcl} (\lambda x.e_1)\ e_2 & \to & e_1[e_2/x] \\ (e_1+e_2)+e_3 & \to & e_1+(e_2+e_3) \\ \underline{n}+e & \to & e+\underline{n} \\ \underline{n_1}+\underline{n_2} & \to & \underline{n_1+n_2} \\ \textbf{ref}\ \underline{n}\ \mathsf{read} & \to & \textbf{ref}\ \underline{n}\ \underline{n} \\ \textbf{ref}\ \underline{n}\ (\mathsf{read}+e) & \to & \textbf{ref}\ \underline{n}\ (\underline{n}+e) \\ \textbf{ref}\ \underline{n}\ \mathsf{inc} & \to & \textbf{ref}\ (\underline{n+1})\ \underline{n} \\ \textbf{ref}\ \underline{n}\ (\mathsf{inc}+e) & \to & \textbf{ref}\ (\underline{n+1})\ (\underline{n}+e) \end{array}$$

The evaluation of a program is defined as follows:

$$eval(p) = \begin{cases} n_2 & \text{if } p \to^* \textbf{ref}\ n_1\ n_2 \\ \texttt{proc} & \text{if } e \to^* \lambda x.e' \end{cases}$$

- Prove that $eval(\textbf{ref}\ 0\ ((\lambda x.x+x)\ \mathsf{inc})) = 1$

- Prove that $x+y$ is not observationally equivalent to $y+x$. **Hint:** Consider the context $((\lambda x.\lambda y.[\ ])\ \mathsf{inc}\ \mathsf{read})$.

- Consider all possible reduction sequences for the program:

$$\textbf{ref}\ 0\ ((\mathsf{inc} + \mathsf{read}) + (\mathsf{inc} + \mathsf{inc}))$$

  and show the result of each.

4 Types

You are given a small language, its type system, and its evaluation relation.

Syntax

$$\begin{array}{llcl} \text{(Expressions)} & e & ::= & \mathsf{zero} \mid \mathit{succ}\ e \mid \mathit{pred}\ e \\ \text{(Values)} & v & ::= & \mathsf{zero} \mid \mathit{succ}\ v \\ \text{(Evaluation contexts)} & E & ::= & [\,] \mid \mathit{succ}\ E \mid \mathit{pred}\ E \end{array}$$

Type System

$$\frac{}{\vdash \mathsf{zero} : \mathtt{int}} \qquad \frac{\vdash e : \mathtt{int}}{\vdash \mathit{succ}\ e : \mathtt{int}} \qquad \frac{\vdash e : \mathtt{int}}{\vdash \mathit{pred}\ e : \mathtt{int}}$$

Evaluation

$$\mathit{pred}\ (\mathit{succ}\ v) \to v$$

$$\frac{e_1 \to e_2}{E[e_1] \longmapsto E[e_2]}$$

$$\mathit{eval}(e) = v \quad \text{if } e \longmapsto^* v$$

- Find an expression e that typechecks but that does not evaluate to a value v.

- State and prove the preservation lemma.