Family Name: ____________________

Given Names: ____________________

Signature: ____________________

THE UNIVERSITY OF NEW SOUTH WALES

Mid-Session Sample Exam

Session 1 2007

# COMP3161/COMP9161

# Concepts of Programming Languages

- Time allowed: **45 minutes**
- Total number of questions: **5**
- Answer **all** questions
- The questions are **not** of equal value
- Do not write your answers on the question sheet — only answers in the provided examination booklets will be marked.
- This paper may **not** be retained by the candidate
- **Answers must be written in ink**, with the exception of graphs
- Drawing instruments or rules may be used
- There is a 3% penalty if you do not fill in your student number and name correctly

## Question 1 [2 Marks]

Consider the following expression $e$:

```
let x = 5 in
  let x = 10 + x  in
    let y = 5 + x in
       x + let x = 15 in x * y end
    end
  end
end
```

Find an expression $e'$ which is $\alpha$-equivalent to $e$ where every variable is only used at a single binding position (i.e., every variable name in the expression occurs only once in a left-hand side of a `let`-binding).

## Question 2 [6 Marks]

In the lecture we discussed the role of the lexer, the parser, and the (static) semantic analyser. For each of these components, give an example of a program error that can be detected by that component. (The error may be in terms of C, Haskell or the language of arithmetic expressions with let-bindings discussed in the lecture).

## Question 3 [6 Marks]

What is the difference between concrete and abstract syntax of a programming language? (keep your answer brief — it may be easiest to describe the difference if you use the possible concrete and abstract syntax of an actual language construct as an example)

**Question 4 [6 Marks]**

Given the following inference rules which define the big step semantics for algebraic expression in abstract syntax. What is the value of

```
plus (num(5), let (num(7), x. plus (x, num(1))))
```

Give the derivation and annotate each rule application with the number of the rule you used.

$$(1)\ \frac{}{\texttt{num}(n) \Downarrow \texttt{num}(n)}$$

$$(2)\ \frac{e_1 \Downarrow \texttt{num}(n_1) \quad e_2 \Downarrow \texttt{num}(n_2)}{\texttt{plus}\ (e_1, e_2) \Downarrow \texttt{num}(n_1 + n_2)}$$

$$(3)\ \frac{e_1 \Downarrow \texttt{num}(n_1) \quad e_2 \Downarrow \texttt{num}(n_2)}{\texttt{times}\ (e_1, e_2) \Downarrow \texttt{num}(n_1 * n_2)}$$

$$(4)\ \frac{e_1 \Downarrow \texttt{num}(n_1) \quad \{\texttt{num}(n_1)/x\}e_2 \Downarrow \texttt{num}(n_2)}{\texttt{let}\ (e_1, x.e_2) \Downarrow \texttt{num}(n_2)}$$

**Question 5 [20 Marks]**

Given a set of induction rules defining $nat$

$$(1)\ \frac{}{\texttt{0}\ nat}$$

$$(2)\ \frac{n\ nat}{\texttt{s}(n)\ nat}$$

and a set of rules defining $enat$:

$$(3)\ \frac{}{\texttt{0}\ enat}$$

$$(4)\ \frac{}{\texttt{s(0)}\ enat}$$

$$(5)\ \frac{n\ enat}{\texttt{s(s(}n\texttt{))}\ enat}$$

Using rule induction, show that both definitions are equivalent, that is, for all $x$, $x\ nat$ is derivable if and only if $x\ enat$ is derivable. Clearly state which cases you have to consider, and what the induction hypothesis is for each case. Annotate derivations with the number of the rule you used.

*Hint:* One direction of the proof is straight forward. For the other direction, you might find it helpful to first prove a Lemma.