TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2007

MID-YEAR

COMP 304

Programming Languages

Time Allowed: 3 Hours

Instructions: There are five (5) questions, each worth thirty (30) marks. You should attempt all of the questions.

Question 1.

(a) The functions `map`, `repeat` and `listfrom` can be defined in Haskell as follows:

```
map _ [] = []
map f (h:t) = f h : (map f t)

repeat f n = n : (repeat f (f n))

listfrom :: Integer -> [Integer]
listfrom = repeat (+1)
```

State the types of `map` and `repeat`. [3 marks]

(b) `zip :: [a] -> [b] -> [(a, b)]` is a function which allows us to 'zip' two lists together to produce a list of pairs. The nth item in the list of pairs is the nth item from the first list paired with the nth item from the second list. The list of pairs has the same length as the shorter of the two lists. Recall that a string is just a list of characters, so `zip` behaves like this:

```
*Exam> zip (listfrom 1) "abcde"
[(1,'a'),(2,'b'),(3,'c'),(4,'d'),(5,'e')]
```

Give a suitable definition in Haskell for `zip`. [6 marks]

(c) Rather than just putting the items from the two lists into a list of pairs we realise that we can apply a function to them, by zipping two lists together with a function. Instead of returning a list of pairs we will return a list of whatever the function returns. For example:

```
*Exam> zip2with max [1,2,3,4,5,6,7,8,9] [9,8,7,6,5,4,3,2,1]
[9,8,7,6,5,6,7,8,9]
```

Give a suitable definition in Haskell for `zip2with`. [7 marks]

(d) Haskell uses lazy evaluation. Briefly describe lazy evaluation, and illustrate its use. [7 marks]

(e) Haskell allows the programmer to define higher-order functions. Briefly explain what a higher-order function is, and give an example of a higher-order function. [7 marks]

Question 2.

(a) Briefly describe the following terms, as they are understood by a Prolog programmer:

(i). fact;

(ii). rule;

(iii). query. [6 marks]

(b) A New Zealand University has decided to automate the process it uses for checking students' degree programmes. The University needs to be able to check that students have the correct pre-requisites for any paper that they wish to take. The rules for pre-requisites for papers in the BSc in Surfing can be expressed in Prolog as:

```
/*

prereq(Course1, Course2) succeeds if Course2 is a pre-requisite for Course1

*/
prereq(surf113, surf112).

prereq(surf212, surf113).
prereq(surf213, surf113).
prereq(surf215, surf113).
prereq(surf216, surf113).

prereq(surf311, surf215).
prereq(surf312, surf215).
prereq(surf312, surf216).
prereq(surf313, surf212).
prereq(surf313, surf216).
prereq(surf314, surf212).
prereq(surf314, surf216)
prereq(surf315, surf213).
prereq(surf315, surf215).
prereq(surf316, surf213).
prereq(surf316, surf215).
prereq(surf389, surf311).
```

(Question 2 continued on next page)

(Question 2 continued)

Unfortunately the University has employed a Surfing graduate, rather than a Computer Science graduate, and he has written the following Prolog:

```
/*

requires(Course1, Course2) succeeds if Course2 is a requirement for Course1.
Course2 is a requirement for Course1 if it is a pre-requisite for a course
which is a requirement for Course1 or if is a pre-requisite for Course1.

*/

requires(Course1, Course2) :-
      requires(Course1, Other),
      prereq(Other, Course2).
requires(Course1, Course2) :-
      prereq(Course1, Course2).
```

Explain how Prolog will attempt to answer the following query, and what the result will be:

```
?- requires(surf389, surf112).
```

[9 marks]

(c) Give a suitable definition in Prolog for `requires/2`. [6 marks]

(d) Although Prolog is a very fine language it has some deficiencies. State three (3) features of Prolog which you believe to be flaws and explain what problems they cause. [9 marks]

Question 3.

(a) Describe *register machines*, and explain how they provide a model for imperative programming. [7 marks]

(b)

(i) Define the syntax of λ-terms. [3 marks]

(ii) Explain how to perform a single step of β-reduction. [4 marks]

(iii) Find a normal form for the following term:

- $(\lambda x \lambda y \lambda z.z\ x\ y)\ p\ q\ (\lambda q \lambda p.q)$ [6 marks]

(iv) Use the following term to show that not all terms have a normal form:

- $(\lambda x.x\ x)\ \lambda y.y\ y$ [4 marks]

(c) Write a short note explaining how the λ-calculus provides a model for functional programming. [6 marks]

Question 4.

(a) State three benefits of providing a formal semantics for a programming language. [6 marks]

(b) Briefly describe the role of:

(i) the abstract syntax;

(ii) the semantic algebra;

(iii) the valuation function;

in denotational semantics. [9 marks]

(c) You are defining a programming language and have defined the following abstract syntax:

$$
\begin{array}{rcl}
P & \in & \mathit{Program} \\
C & \in & \mathit{Command} \\
E & \in & \mathit{Expression} \\
B & \in & \mathit{BooleanExpr} \\
I & \in & \mathit{Identifier} \\
N & \in & \mathit{Numeral}
\end{array}
$$

$$
\begin{array}{rcl}
P & ::= & C. \\
C & ::= & C1\ ;\ C2 \\
 & | & \texttt{if}\ B\ \texttt{then}\ C \\
 & | & \texttt{if}\ B\ \texttt{then}\ C1\ \texttt{else}\ C2 \\
 & | & I := E \\
E & ::= & E1 + E2 \mid E1 - E2 \mid E1 * E2 \mid E1\ /\ E2 \mid I \mid N \\
B & ::= & E1 == E2 \mid \neg B \mid B1 \wedge B2
\end{array}
$$

(Question 4 continued on next page)

(Question 4 continued)

You have decided that the store will be a mapping from identifiers to numbers. Hence the semantic algebra for stores look like:

$$
\begin{array}{l}
\mathit{Domain\ Store} = \mathit{Id} \rightarrow \mathit{Nat} \\
\mathit{Operations} \\
\mathit{newstore} : \mathit{Store} \\
\mathit{newstore} = \lambda i.\mathit{zero} \\
\mathit{access} : \mathit{Id} \rightarrow \mathit{Store} \rightarrow \mathit{Nat} \\
\mathit{access} = \lambda i.\lambda s.s(i) \\
\mathit{update} : \mathit{Id} \rightarrow \mathit{Nat} \rightarrow \mathit{Store} \rightarrow \mathit{Store} \\
\mathit{update} = \lambda i.\lambda n.\lambda s.[i \mapsto n]s
\end{array}
$$

The valuation function for commands will be

$$\mathbf{C} : \mathit{Command} \rightarrow \mathit{Store}_{\perp} \rightarrow \mathit{Store}_{\perp}$$

Give a suitable valuation function for *Command*s, assuming that you have suitable valuation functions for *Program*, *Expression*, *BooleanExpr*, *Identifier*, *Numeral*. [15 marks]

Question 5.

(a) Write a short note explaining how the λ-calculus can help us to understand an imperative language like Java. [10 marks]

(b) Describe how Haskell's system of type classes makes *ad-hoc* polymorphism less *ad hoc*. [10 marks]

(c) Logic, functional and imperative programming languages all have variables. Carefully explain how the notion of variable differs between them. [10 marks]

****