UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2004

MEng Honours Degrees in Computing Part IV
MSc in Advanced Computing
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C470

PROGRAM ANALYSIS

Monday 26 April 2004, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1 a Consider the following While program:

```
p := 1;
t := m + 1;
while n > 0 do
   p := p * (m + 1);
   n := n - 1;
```

Label this program. Write down the results of the kill and gen functions and the equations for an Available Expressions Analysis of this program. Write down the solution of the equations.

b A modification of Available Expressions Analyis detects when an expression is available in a particular variable. Write down the data flow equation schemes and any auxiliary functions for this analysis.

c Write down a transformation system that uses the results, AE, of the analysis of part b to replace non-trivial expressions by variables where appropriate. You need only write down the rules for assignment, sequencing and while loops. You may find it helpful to write $E(a)$ for the expression which contains a as a non-trivial arithmetic subexpression.

The three parts carry, respectively, 30%, 35%, and 35% of the marks.

2 Consider the While language with statements:

$$\begin{array}{rcl} S & ::= & \mathtt{x := a} \\ & | & \textbf{skip} \\ & | & S_1 \; ; \; S_2 \\ & | & \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \\ & | & \textbf{while } b \textbf{ do } S \end{array}$$

The *range* of a variable x is defined to be:

$$\text{range}(\mathrm{x}) = \begin{cases} \mathtt{<-1} & \text{if } \mathrm{x} < -1 \\ \mathtt{-1} & \text{if } \mathrm{x} = -1 \\ \mathtt{0} & \text{if } \mathrm{x} = 0 \\ \mathtt{+1} & \text{if } \mathrm{x} = +1 \\ \mathtt{>+1} & \text{if } \mathrm{x} > 1 \end{cases}$$

Define a Range Analysis Range as an instance of the monotone framework (similar to the Constant Propagation Analysis CP). For each program point, the Range Analysis will determine the range a variable may have, whenever execution reaches that point.

Assume the usual labelling. Take the flow, F, to be *flow*$(S_\star)$, and the extremal labels, E, to be $\{init(S_\star)\}$.

a Define the lattice of ranges **Range** and the lattice of states $\widehat{\textbf{State}}$ underlying the analysis Range (in particular specify their respective '$\sqsubseteq$' and '$\sqcup$').

b Define the (abstract) evaluation function:

$$\mathcal{A}_{\mathsf{Range}} : \textbf{AExp} \rightarrow (\widehat{\textbf{State}} \rightarrow \textbf{Range})$$

of arithmetic expressions involving addition, including the $\widehat{+}$ operation on **Range**.

c Define the transfer functions (for all labels ℓ):

$$f_\ell(\sigma)^{\mathsf{Range}} : \widehat{\textbf{State}} \rightarrow \widehat{\textbf{State}}.$$

The three parts carry, respectively, 30%, 40%, and 30% of the marks.

3 Consider the following simple functional programming language:

```
t ::=  c | x | fn x => e | e1 e2 | e1 + e2
e ::=  t^ℓ
```

a Consider the term:

```
(fn f => fn x => f (f x))(fn y => y + 2) 4
```

Show how this term is evaluated. Add labels to the term and guess the result of a 0-CFA of the term.

b Write down a syntax-based specification of a control flow analysis for this language (0-CFA) which takes account of the left to right evaluation order imposed by call-by-value semantics: for a call-by-value language in the evaluation of an application term ee' there is no need to analyse the argument e' if e cannot produce any closures.

c The control flow analysis is to be extended with a data flow analysis that tracks numerical values using modulo 3 arithmetic. A suitable set of abstract values is thus:

$$\mathbf{Mod3} = \{0, 1, 2\}$$

Write down a suitable definition of +, and write down the rules for the specification of the analysis (you need only write down rules that are different from your answer to part b).

The three parts carry, respectively, 30%, 35%, and 35% of the marks.

4 a Consider the following constraint system:

$$\begin{aligned} x_1 &= A \\ x_2 &= x_1 \\ x_3 &= x_2 \cup x_4 \\ x_4 &= x_3 \cup x_6 \\ x_5 &= x_4 \cup B \\ x_6 &= x_5 \cup C \\ x_7 &= x_2 \cup x_8 \\ x_8 &= x_7 \cup D \\ x_9 &= x_7 \cup x_3 \end{aligned}$$

where A, B, C and D are constant sets. Write down the least solution, showing your working.

b Using the constraints from part a, draw a graph of the constraints, write down an algorithm for constructing a depth-first spanning forest and apply it to your graph. Hence or otherwise, write down the rPOSTORDER ordering of the constraints.

c Write down the definition of *strong components* of a graph. Identify the strong components in the graph of part b and draw the reduced graph. Give a topological ordering of the strong components.

d Describe (in words rather than pseudocode) how strong components can be used to modify the way in which nodes are selected from the worklist. In particular, write down pseudocode for the `extract` function which selects the next node to process. Show how the modified worklist algorithm affects the iterative sequence to find the least solution of the constraints of part a.