No: 509.xxx

THE CITY UNIVERSITY

LONDON

B.Eng. Computing
B.Eng. Software Engineering
B. Eng. Computing (DISC)
B.Sc. Business Computing Systems
Professional Pathway 3 - Software Engineering

PART III EXAMINATION

Language Processors

May 9th 2003 9.00-11.00

Answer THREE questions
All questions carry equal marks

1. (a) The following regular expression recognises certain strings consisting of the letters a, b and c:

$$a((ab)|(ac)) * c$$

Indicate which of these five strings are recognised by the above regular expression:

$aacc$, $abac$, ac, $abababacac$, $aabacc$

Also, show three more strings that are recognised by the above expression. Finally, show two more strings consisting of the letters a, b and c that are *not* recognised by the above regular expression. [30]

(b) Most programming languages allow the same symbol to denote both the subtraction operator (in $x-y$) and the unary negation operator (in $-x$). Make clear the difficulty this causes for a parser-generator (for example, CUP) and explain how it may be overcome. [20]

(c) Explain what it means for a context-free grammar to be ambiguous. Write down an ambiguous grammar and show why it is ambiguous. [20]

(d) Consider the following Tiger function:

```
1 function f(a:string, b:int, c:int)=
2       (print_int(b+c);
3        let var c := "hi"
4            var a := b
5            var b := "hello"
6        in print(b); print_int(a)
7        end;
8        print_int(c); print_int(b);
9       )
```

Given an initial environment $\sigma_0 = \{\text{a} \rightarrow \text{int}, \text{b} \rightarrow \text{string}\}$, derive the type binding environments for the function at each use of an identifier and indicate where type lookups will occur. [30]

2. The reference manual for a Tiger-like programming language contains the following definition for a kind of expression:

   The if-expression

   if exp_1 **then** exp_2 **else** exp_3

   evaluates the expression exp_1. If the result is non-zero the if-expression yields the result of evaluating exp_2; otherwise it yields the result of evaluating exp_3.

   (a) Write down a BNF concrete syntax for the if-expression. [10]

   (b) Sketch a possible abstract syntax for the if-expression. [10]

   (c) Show how semantic actions in a grammar for a parser-generator such as CUP can be used to produce abstract syntax trees for the if-expression. [20]

   (d) Informally describe an appropriate typecheck for the if-expression. [20]

   (e) Suppose a Tiger compiler translates all expressions and subexpressions into intermediate code (eg expression trees). Outline the intermediate code that might be generated in translation of the if-expression:

   ```
   if a < b then c := a else c := b
   ```

   You can assume that the expression tree for any variable `v` is simply `TEMP v`. [40]

3. (a) Choose a programming language you know well and describe how run-time storage is organised and managed during program execution. Clearly associate any storage structures you mention with the implementation of particular language features. [25]

   (b) What is a stack frame? Outline a typical layout for a stack frame and describe each element of a frame. Comment on how local variables, arguments and non-local variables are addressed by the code generated for a procedure or method. [25]

   (c) Explain why registers might be used for parameter passing and suggest situations where passing in registers is particularly appropriate. Outline situations where it is necessary for the code generated for a procedure or method to write registers to the stack. [30]

   (d) Explain the difference between *caller-save* and *callee-save* registers. Why might caller-save registers sometimes not be saved? [20]

4. (a) Parser-generators such as CUP provide commands for specifying the *precedence* and *associativity* (or *non-associativity*) of operators in a programming language. Using examples, explain the difference between precedence and associativity and show how both are specified in a parser-generator you know (eg CUP). [40]

   (b) A version of the Tiger language allows logical expressions to include both the short-circuiting Boolean operators that do not evaluate the right-hand operand subexpression if the result is determined by the left-hand one (written & and |), and Boolean operators that always evaluate both of their operand subexpressions (written && and ||).

      i. Give a grammar for these logical expressions omitting the semantic actions for the construction of the abstract syntax representation. Include specifications of precedence and associativity. Write your answer in the form of input to a parser-generator (eg CUP), and assume that the rest of the expression productions have been written. [10]

      ii. The two short-circuiting boolean operators can be regarded as abbreviations for other expressions. They can be removed by a parser and replaced by an equivalent expression. The two operators that do not perform short-circuiting evaluation can also be treated in this way. Explain how, for all four cases. Logical expressions in Tiger produce the integer 1 for true, and 0 for false. [30]

      iii. Show how your answer for the two short-circuiting operators in part (ii) can be expressed in semantic actions for a parser-generator producing an abstract syntax representation. You may assume that appropriate abstract syntax representation definitions are already written. [20]