No: 510.00

THE CITY UNIVERSITY

LONDON

B.Eng. Degree in Computing
B.Eng. Degree in Software Engineering
B.Sc. Degree in Mathematical Science with Computer Science

PART III EXAMINATION

Language Processors

10th May 2002 0900–1100

Answer THREE questions
All questions carry equal marks

1. (a) Give a definition for regular expressions which shows how arbitrary regular expressions can be constructed for a given alphabet (set) of symbols. Do not include shorthand notations you may know. [10]

(b) Write down regular expressions that define a pattern to recognise a decimal literal. Decimal literals start with a numeral consisting of one or more digits. This may be followed by an optional decimal part consisting of a decimal point '.' followed by another numeral consisting of one or more digits. Finally this may be followed by an optional exponent consisting of the letter 'E' followed by an optional sign followed by a numeral consisting of one or more digits. [25]

(c) Write down regular expressions that define a pattern to recognise identifiers that begin with a letter, and may be followed by any number of letters, digits, and underscores. Sequences of underscores are not permitted, and the last character cannot be an underscore. [30]

(d) Consider the following Tiger function:

```
function f(a:string, b:int, c:int)=
      (print_int(b+c);
       let var a := b
           var b := "hello"
       in print(b); print_int(a)
       end;
       print_int(b);
      )
```

Given an initial environment $\sigma_0 = \{\mathrm{a} \to \mathrm{int}, \mathrm{b} \to \mathrm{string}\}$, derive the type binding environments for the function at each use of an identifier and indicate in which type environment type lookups will occur. [35]

2. Consider the following expression grammar, which we will call E:

$$\begin{array}{lll} exp & \rightarrow & exp \texttt{ + } exp \\ & | & exp \texttt{ - } exp \\ & | & exp \texttt{ * } exp \\ & | & int \end{array}$$

Here *int* stands for any integer.

(a) What is an ambiguous grammar? By giving suitable examples show that E is ambiguous. [20]

(b) Use an expression derived from E to explain the term *shift-reduce conflict*. [15]

(c) Disambiguate E, arranging the productions so that the + and - operators have equal precedence lower than that of *, and so that expressions with the same precedence group left-to-right. [30]

(d) Show how either E, or the grammar you wrote down in your answer to part (c) , can be encoded in a parser-generator like CUP with appropriate semantic actions to act as a simple integer expression evaluator. If you choose to use E, you must show how CUP directives are used to make the operators + and - have equal precedence lower than that of *. [35]

3. The reference manual for a Tiger'01-like programming language contains the following definition for a kind of expression:

The for-expression

$$\textbf{for } id = exp_1 \textbf{ to } exp_2 \textbf{ do } exp_3$$

iterates exp_3 over each integer value of *id* from the value of exp_1 to that of exp_2. The variable *id* is a new variable implicitly declared by the **for** statement, whose scope covers only exp_3, and may not be assigned to. The lower and upper bounds exp_1 and exp_2 are evaluated only once, prior to entering the body of the loop exp_3. If the upper bound is less than the lower, the body is not executed.

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

(b) Sketch a possible abstract syntax for the for-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 for-expression. [30]

(d) Explain how the scope rules for the implicitly declared variable might be implemented. [15]

(e) Outline the intermediate representation that might be generated from your abstract syntax trees in translation of a for-expression. You may wish to use a simple example to explain your translation, eg:

```
for i = j to j+10 do ( x := i*i; print(x) )
```

[35]

4. (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. [35]

   (b) What is a stack frame? Outline a typical layout for a stack frame and describe each element of a frame and how it is pushed to the stack during program execution. Comment on how local variables, arguments and non-local variables are addressed by the code generated for a procedure or method. [35]

   (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]

External examiners:	Professor M.E.C. Hull	Examiners:	D. Bolton
	Professor M. Moulding		