University of Limerick

College of Informatics and Electronics
Department of Computer Science and Information Systems

2003/2004

Semester : Semester 1	Academic Year : 2003/2004 (Repeat)
Module Code : CS4158	Module Title : Programming Language Technology
Duration of Exam : 2.5 hours	Marks : 100% of Module Marks (for Repeat Students)
Lecturer(s) : Professor T. Cahill, Dr J. Buckley	

Instructions to Candidates :- Attempt 5 questions, with at least two questions from each section. If you attempt more than 5 questions, you must clearly indicate which answers are not to be marked, otherwise the examiners will mark the earlier attempts in each part. All questions carry equal marks. Note that the examiners can take into account the quality of presentation and exposition as well as the content. Note carefully the marks assigned to each part of a question in order to decide the level of detail for your answer.

---

1 Section 1

Q1

Explain the concepts of arity, horizontal fixity of operators and delimiters, as they occur in mathematical notation, giving examples to illustrate your explanation. (4 marks)

Explain what is meant by a stretchable operator symbol, such as the vertical infix division symbol, and the stretchable square root operator. Show how structure of the expression $\sqrt{((a+b) \div (a-b))}$ can be written more simply by using stretchable operators. Draw an expression tree for the expression. (6 marks)

Given a set of unary and binary operator symbols O with only horizontal fixities, a set of literal symbols L, and a set of variable symbols V, as well as an outfix delimiter pair (), and ',' as an infix delimiter, give a generic formal specification of the set of expressions constructible from O, V, L. Show how these rules can be easily specialized by providing actual sets for O, L, V to deal respectively with Boolean and Arithmetic expressions. (10 marks).

Q2

Explain the concepts of priority(precedence) and associativity of binary infix operators and what their value is to writers (and readers) of mathematical notation. Use a 'gobble diagram' to illustrate both concepts. Explain the circumstances where priority may also be needed for unary operators. (6 marks)

Explain the central purpose of Polish Notation and describe both its variants. Given a formal specification of expressions, (you need only focus on unary and binary operators) formally specify the rules for converting an expression to Reverse Polish Notation, taking into account the priority and associativity of the operations. (14 marks)

Q3

Explain what is meant by leftmost and rightmost derivations of a sentence generated by a context free grammar, and explain how (and why we) con-

struct a parse tree for sentences generated by a context free grammar. (6 marks)

Explain how a context free grammar for expressions containing operators of differing priorities and associativities is defined. You are recommended to make use of a diagram linking the Non Terminal Symbols (via the operator symbols in the right hand sides of the productions and otherwise), to explain the method. (6 marks)

Draw a parse tree for the expression, $a + (b + c) \times d \uparrow 2$, and from that, derive a syntax tree for the same expression, justifying your work, and then derive an abstract syntax tree for the same expression, also justifying your work. Explain the importance of abstract syntax trees for expressions. (8 marks)

2 Section 2

Q4

(a) Describe, by building an $LR(0)$ finite state automata for the following grammar, why it is $LR(1)$.

$S \rightarrow P, P \rightarrow G, G \rightarrow G * Q, G \rightarrow h, Q \rightarrow n$ (5 marks)

(b) Build the $LR(1)$ finite state automata for the grammar, showing how the problem has been removed. (5 marks)

(c) Describe why $LR(1)$ analysis of many grammars is considered impractical. (4 marks)

(d) Briefly discuss two alternative (practical) techniques for analyzing $LR(1)$ grammars. (6 marks)

Q5

(a) Using the $LL(1)$ grammar given below, calculate the predict set for each production.

$E \rightarrow P[E], E \rightarrow tQ, P \rightarrow G, P \rightarrow \lambda, G \rightarrow p, Q \rightarrow -E, Q \rightarrow \lambda$ (6 marks)

(b) Some grammars are not $LL(1)$. Describe, in detail, two conditions which conflict with a grammar being $LL(1)$. (8 marks)

(c) For one of the conditions in part (b), show in detail how it can be overcome, using a 'grammar re-structuring' technique. (6 marks)

Q6

(a) Form a 'Flex' regular expression for:-

- A variable name composed of all letters, at least 1 letter in length;
- A variable as above, but composed of numbers or letters, starting with

a letter;

- A variable name as above, with 0, 1 or more dashes (-) in it, again starting with a letter and ending with a question mark.

(9 marks)

(b) Draw a Transducer for the third regular expression constructed above. (6 marks)

(c) From this deterministic FSA, create a transition table. (5 marks)

Q7

(a) Differentiate between 'Context Sensitive', 'Context Free' and 'Regular' Grammars. (9 marks)

(b) A 'C' programmer carelessly assigns a value to a variable without previously declaring the variable. Briefly, explain, as per your definition above, how a Context Sensitive Grammar could be used to check for this. (4 marks)

(c) Describe how the limitations of context free grammars can be overcome, in carrying out such checks. (7 marks)