University of Limerick

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

Summer 2003/2004

Semester : Semester 2	Academic Year : 2003/2004
Module Code : CS4518	Module Title : Programming Language Technology
Duration of Exam : 2.5 hours	
Lecturer(s) : Professor T. Cahill, Dr J. Buckley	

Instructions to Candidates :- Attempt two questions from part 1, and three questions from part 2. If you attempt more 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 marks assigned to each part of a question to decide the level of detail in answer.

1 Part 1

Q1

Explain the concepts of arity, vertical and horizontal fixity of operators and delimiters, as they occur in mathematical notation, giving examples to illustrate your explanation. (2 marks)

Given a set of 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. (6 marks).

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. (3 marks).

Q2

Explain the central purpose of Polish Notation and describe both its variants, and given a formal specification of expressions, formally specify the rules for converting an expression to Reverse Polish Notation, taking into account the priority and associativity of the operations. (7 marks)

Explain the practical advantages of Reverse Polish Notation in evaluation of expressions, and illustrate how it leads to very simple code generation of expressions. Illustrate your answer using the example $a + (2 * a - 3)$ (2 marks)

Describe why expression trees and Polish Notation expressions are derivable from one another, using the above example. (2 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) construct a parse tree for sentences generated by a context free grammar. (2 marks)

Why is it possible to construct a parse *tree* for Context Free Grammars, but not for Context Sensitive Grammars? (1 mark)

What shape will a parse tree for a regular Grammar take? (1 mark)

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. (4 marks)

Draw a parse tree for the expression, $a + b \times d \uparrow 2$, 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. (3 marks)

2 Part 2

Q4

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

   $S \to P\$, P\$ \to R, R \to R - T, R \to q, T \to l$ (5 Marks)

2. Build the LR(1) finite state automata for the grammar, showing how the problem has been removed. (6 Marks)

Q5

1. 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)

2. Discuss the role of look-ahead during LL(1) parsing. (3 Marks)
3. Explain the term sentential form (2 Marks)

Q6

1. Form a Flex regular expression for a variable which starts with a letter, but (subsequently) can contain letters, numbers and dashes (-). The definition should allow no 2 adjacent dashes. (3 Marks)
2. Draw a Transducer for the regular expression constructed above (3 Marks)
3. From this deterministic FSA, create a transition table. (2 Marks)
4. Discuss the role of look-ahead during LR(1) parsing. (3 Marks)

Q7

1. Generate a regular grammar that can parse the following sentences
   (a) The drunk sailor ate the pie
   (b) The sad tired sailor ate the cake
   (c) The fireman drank the delicious cider

   (6 Marks)
2. Generate a context free grammar that can parse these sentences (5 Marks)