

DUBLIN CITY UNIVERSITY

SEMESTER ONE EXAMINATIONS 2006

MODULE: Compiler Construction 1
(Title & Code) CA448/CA423

COURSE: B.Sc. in Computer Applications (SE)
B.Sc. in Computer Applications (CSSE)
B.Sc. in Computer Applications (IS)
B.Sc. in Computer Applications (Evening)
B.Sc. in Computational Linguistics

YEAR: 4

EXAMINERS: Mr. D. Dolan
Dr. P. Gibson
Dr. G. Hamilton (ext. 5017)

TIME ALLOWED: 2 Hours

INSTRUCTIONS: Please answer ALL questions.
All questions carry equal marks

Requirements for this paper
Please tick (X) as appropriate

<input type="checkbox"/>	<i>Log Table</i>
<input type="checkbox"/>	<i>Graph Paper</i>
<input type="checkbox"/>	<i>Attached Answer Sheet</i>
<input type="checkbox"/>	<i>Statistical Tables</i>
<input type="checkbox"/>	<i>Floppy Disk</i>
<input type="checkbox"/>	<i>Actuarial Tables</i>

**THE USE OF PROGRAMMABLE OR TEXT STORING
CALCULATORS IS EXPRESSLY FORBIDDEN**

**PLEASE DO NOT TURN OVER THIS PAGE UNTIL YOU ARE
INSTRUCTED TO DO SO**

1. A fixed point number is defined to be an optional sign (+ or -), followed by a sequence of zero or more digits, followed by a decimal point, followed by a sequence of zero or more digits; containing at least one digit (before or after the decimal point). Express a fixed point number as a regular expression. [10 marks]

2. Express a fixed point number as described in (a) as a deterministic finite automaton. [10 marks]

3. Show that the following grammar is ambiguous. [10 marks]

$$\begin{aligned} S &\rightarrow aSbS \\ S &\rightarrow bSaS \\ S &\rightarrow \epsilon \end{aligned}$$

By giving two different parse trees for the sentence *abab*.

4. Verify whether or not the following grammar is LL(1). [10 marks]

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow C \\ A &\rightarrow aA \\ B &\rightarrow a \\ C &\rightarrow b \\ C &\rightarrow \epsilon \end{aligned}$$

5. Convert the following grammar into an LL(1) grammar which recognises the same language. [10 marks]

$$\begin{aligned} E &\rightarrow E + T \\ E &\rightarrow T \\ T &\rightarrow id \\ T &\rightarrow id() \\ T &\rightarrow id(L) \\ L &\rightarrow E;L \\ L &\rightarrow E \end{aligned}$$

6. Verify whether or not the following grammar is LR(0). [10 marks]

$$\begin{aligned} S &\rightarrow xA \\ S &\rightarrow By \\ A &\rightarrow xA \\ A &\rightarrow y \\ B &\rightarrow xB \\ B &\rightarrow y \end{aligned}$$

7. Verify whether or not the following grammar is LR(1). [10 marks]

$S \rightarrow Aa$
 $S \rightarrow bAc$
 $S \rightarrow Bc$
 $S \rightarrow bBa$
 $A \rightarrow d$
 $B \rightarrow d$

8. Verify whether or not the grammar in question 7 is LALR(1). [10 marks]

9. Consider the following grammar for integer expressions: [10 marks]

$E \rightarrow E + T$
 $E \rightarrow E - T$
 $E \rightarrow T$
 $T \rightarrow T * F$
 $T \rightarrow T / F$
 $T \rightarrow F$
 $F \rightarrow \text{int}$
 $F \rightarrow (E)$

Add attributes and rules that calculate the integer value of an expression (hint: the only attribute that will be required for each non-terminal will be one denoting its integer value).

10. Consider the following grammar for simple expressions: [10 marks]

$E \rightarrow E + T \mid E - T \mid T$
 $T \rightarrow T * F \mid T / F \mid F$
 $F \rightarrow (E) \mid \text{id}$

Construct an abstract syntax tree for the following expression:

$a * (a - c) / (a - c)$

Convert this abstract syntax tree into an equivalent directed acyclic graph by eliminating all common subexpressions.

[Total marks: 100]