UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

FACULTY *of* SCIENCE *and* ENGINEERING

Department of Computer Science
and Information Systems

End-of-Semester Assessment Paper

Academic Year:	2008/09	Semester:	Autumn
Module Title:	Programming Language Technology	Module Code:	CS4158
Duration of Exam:	2½ Hours	Percent of Total Marks:	80
Lecturer:	Richard F. E. Sutcliffe	Paper marked out of :	100

Instructions to Candidates:

- Answer any FIVE questions
- Only the FIRST FIVE questions attempted will be marked
- All questions carry equal marks

Q1. a) Define the terms *terminal symbol*, *non-terminal symbol*, *string*, *sentence* and *language*.

5 Marks

b) According to Noam Chomsky, a grammar can be defined as a quadruple. What are the four parts?

4 Marks

c) Consider the following grammar:

E → E + E | E * E | (E) | - E | **id**

Write out one leftmost derivation and one rightmost derivation for **id** * **id**, starting with E.

4 Marks

d) What does it mean for a grammar to be *ambiguous*? **Hint:** Answer in terms of derivations.

3 Marks

e) Write down two parse trees for **id** * **id** + **id**, using the grammar of part (c).

4 Marks

Q2. a) What is a *compiler*? Give any **two** definitions of the three given in lectures.

4 Marks

b) A compiler can be said to involve four phases: *Lexical Analysis*, *Syntax Analysis*, *Semantic Analysis* and *Code Generation*. Write a short of account of each of these.

10 Marks

c) In addition, there are two associated processes, *Symbol Table Management* and *Error Handling*. Describe these briefly.

6 Marks

Q3. a) What is *parsing*?

2 Marks

b) Explain the difference between top-down and bottom-up parsing. Illustrate your answer using the string 'abde' and the following grammar:

1. $S \rightarrow a\ A\ B\ e$
2. $A \rightarrow A\ b\ c$
3. $A \rightarrow b$
4. $B \rightarrow d$

4 Marks

c) Explain exactly what is meant by the terms LL(1) and LR(1) in relation to parsing.

4 Marks

d) Identify any instances of left recursion or right recursion in the grammar of part (b). For each, explain exactly why it is left recursive or right recursive.

2 Marks

e) For what kind of parser must left recursion be eliminated from the underlying grammar? Explain carefully why this is so by choosing a left recursive rule from one of the grammars in this paper, and showing what will happen when that rule is used during parsing.

4 Marks

f) What is *indirect* left recursion? To illustrate your point, alter the grammar of part (b) in any way you choose, so that the resulting grammar contains an instance of indirect left recursion. **Note:** Your new grammar can recognise a different language.

4 Marks

Q4. a) What is a recursive descent parser and how does it work? Is it a top-down or a bottom-up method?

4 Marks

b) Here is a grammar followed by the start of a recursive descent parser for it:

1. E → T E'
2. E' → + T E'
3. E' → ε
4. T → F T'
5. T → - F T'
6. T' → * F T'
7. T' → ε
8. F → (E)
9. F → **id**

```
proc E();
{
    T();
    E'();
}

proc E'();
{
    if input-symbol = '+' then
    {
        advance();
        T();
        E'();
    }
}
```

Complete the parser, based on the above grammar.

12 Marks

c) State one advantage and one disadvantage of the recursive descent approach.

4 Marks

Q5. a) The Earley parsing algorithm uses a *predictor*, a *scanner* and a *completer*. Explain what these do.

6 Marks

b) Here is a grammar followed by part of a parse for the input **id** + **id** * **id**:

$P \rightarrow S$
$S \rightarrow S + M$
$S \rightarrow M$
$M \rightarrow M * T$
$M \rightarrow T$
$T \rightarrow$ **id**

Stateset S(0): • id + id * id

```
1. P → • S        [0,0] start rule
2. S → • S + M    [0,0] predict from (1)
3. S → • M        [0,0] predict from (1)
4. M → • M * T    [0,0] predict from (3)
5. M → • T        [0,0] predict from (3)
6. T → • id       [0,0] predict from (5)
```

Stateset S(1): id • + id * id

```
1. T → id •       [0,1] scan from S(0)(6)
2. M → T •        [0,1] complete from S(0)(5)
3. M → M • * T    [0,1] complete from S(0)(4)
4. S → M •        [0,1] complete from S(0)(3)
5. S → S • + M    [0,1] complete from S(0)(2)
6. P → S •        [0,1] complete from S(0)(1)
```

For the following line, explain the meaning of the dot and the notation `[0,0]`:

```
S → • S + M   [0,0]
```

4 Marks

c) Now complete the parse, using the two statesets of part (b) as a start. There should be four additional statesets.

10 Marks

Q6. a) An LR parser uses an *input*, an *output*, a *stack*, a *parsing table* and a *driver program.* It has two tables called 'action' and 'goto'. During parsing, it consults the top of the stack and the current input. There are four types of action. Explain what these are and describe exactly how they are determined using the parsing tables.

8 Marks

b) Here is a grammar followed by the corresponding parsing table. For state 2, the action under '*' is s7. For state 11 the action under ')' is r5. What exactly do s7 and r5 mean?

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → **id**

State	**Action**						**Goto**		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

2 Marks

c) Here is the start of a parse of **id** * **id** + **id** using the grammar and parsing tables of part (b). Complete the parse. There should be ten further lines.

Stack	**Input**	**Action**
0	**id** * **id** + **id** $	shift
0 **id** 5	* **id** + **id** $	reduce 6. F → **id**
0 F 3	* **id** + **id** $	reduce 4. T → F
0 T 2	* **id** + **id** $	shift

10 Marks

Q7. a) The LR(0) collection of items is constructed from a grammar using two operations, *closure* and *goto*. Explain the closure operation using the grammar below and performing it on the set of items { $F \to (\cdot E)$, $E \to \cdot E + T$ }.

0. $E' \to E$
1. $E \to E + T$
2. $E \to T$
3. $T \to T * F$
4. $T \to F$
5. $F \to (E)$
6. $F \to$ **id**

4 Marks

b) Now explain the goto operation when applied once to the set { $F \to (\cdot E)$, $E \to \cdot E + T$ }.

4 Marks

c) Here are the first six states (numbered zero to five) in the LR(0) collection of items for the grammar of part (a). Complete the collection by adding six further states.

I_0: $E' \to \cdot E$
$E \to \cdot E + T$
$E \to \cdot T$
$T \to \cdot T * F$
$T \to \cdot F$
$F \to \cdot (E)$
$F \to \cdot$ **id**

I_1: $E' \to E \cdot$
$E \to E \cdot + T$

I_2: $E \to T \cdot$
$T \to T \cdot * F$

I_3: $T \to F \cdot$

I_4: $F \to (\cdot E)$
$E \to \cdot E + T$
$E \to \cdot T$
$T \to \cdot T * F$
$T \to \cdot F$
$F \to \cdot (E)$
$F \to \cdot$ **id**

I_5: $F \to \text{id} \cdot$

8 Marks

d) In state 1 on reading a '+', what state should the parser go to and will it be a shift or a reduce? Explain carefully in terms of the collection of items you have constructed.

4 Marks

Q8. a) What is meant by the term *lexical analysis*?

2 Marks

b) What is the relationship between Regular Expressions, Regular languages and the Chomsky Hierarchy?

2 Marks

c) Here are regular definitions for various tokens in a language:

if	$\rightarrow$ if
then	$\rightarrow$ then
else	$\rightarrow$ else
relop	$\rightarrow$ < | <= | = | <> | > | >=
id	$\rightarrow$ **letter** (**letter** | **digit**) *
num	$\rightarrow$ **digit**$^+$ (. **digit**$^+$)? (E(+|-)? **digit**$^+$)?

For each of the following tokens, state whether or not it is an instance of **num** and explain why:

426C 5280 6.33E4 4.2E3.1 3.14 3EXP2D

6 Marks

d) Write down a transition diagram for **relop**.

6 Marks

e) Assuming we have transition diagrams for all the token types of part (c), outline how these are used to create a tokeniser.

4 Marks