PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Final Examination: Semester 1 **Academic Year**: 2006-2007

Date: Monday 9th October, 2006 **Time**: 13:30 – 16:30 (3 hours)

Subject Number: 240-340 **Room**: Hua Hun (Robot)

Subject Title: Compiler Structures

Lecturer: Aj. Andrew Davison

Exam Duration: 3 hours

This paper has 3 pages.

Authorized Materials:

- Writing instruments (e.g. pens, pencils).
- Books (e.g. dictionaries) and calculators are **not** permitted.

Instructions to Students:

- *Answer questions in English.* Perfect English is **not** required.
- Attempt all questions.
- Write your answers in an answer book.
- Start your answer to each question on a new page
- Clearly number your answers.
- Any unreadable parts will be considered wrong.
- When writing programs, use good layout, and short comments; marks will not be deducted for minor syntax errors.
- The marks for each part of a question are given in brackets (...).

Question 1 (80 minutes; 80 marks)

a) Use shift-reduce parsing to evaluate the string "(y-y-x)" against the grammar:

S → '(' A ')'
A → y '-' A | x

S and A are non-terminals, and '(', ')', '-', y, and x are terminals. S is the starting non-terminal. (10)

b) **Briefly** explain the following LR table generation techniques:

- LR(0) items (5)
- the closure() function (5)
- the goto() function (5)

c) Produce a LR parse table for the grammar in part (a) using the techniques of part (b). Show **all** your working. (40)

d) Evaluate the string "(y-y-x)" using your LR parse table from part (c). (15)

Question 2 (55 minutes; 55 marks)

a) Explain yacc by specifying the main sections of a typical yacc program. (10)

b) What is an attribute grammar? (10)

c) Define an attribute grammar for the context free grammar given below:

Pos → Path
Path → Path Move | ε
Move → left | right | fwd | back | up | down

Pos, Path, and Move are non-terminals, while left, right, fwd, back, up, and down are terminals. Pos is the starting non-terminal.

A Pos sentence specifies how an object is moved from its starting position at the coordinates (0,0,0) to a new position in (x,y,z) space. (15)

d) Write a yacc grammar which implements your attribute grammar of part (c). Explain in words what data types you have defined.
Note: do **not** write a lex grammar. (20)

Question 3 on Next Page.

Question 3 (45 minutes; 45 marks)

a) What is intermediate code? Give some **brief** examples of the different kinds. (15)

b) Describe the stack-based intermediate code used by the expressions language. Do **not** include any parser code, but include diagrams where possible. (10)

c) Translate the expressions program:

```
let x = 2
let y = 3 + x
```

into intermediate code. Explain the translation in words. Do **not** include any parser code, but include diagrams where possible. (10)

d) Evaluate the intermediate code of part (c). Show all your working. Do **not** include any parser code, but include diagrams where possible. (10)

--- End of Examination ---