UNIVERSITY OF DUBLIN

TRINITY COLLEGE

CS3BA71

Faculty of Engineering and Systems Sciences

School of Engineering

BA (Mod) Computer Science
Junior Sophister Examination

Trinity Term, 2001

3BA7 — Software Engineering and Compiler Design

Wednesday 23 May, 2001 Mansion House 9:30 - 12:30

Dr JA Redmond, Dr DM Abrahamson

Attempt five questions, at least two from each section.
Please use separate answer books for each section.

Section A

1. i. Write a note on the Rational Unified Process (RUP).

   ii. Write a brief note on OPEN (Object-Oriented Process, Environment and Notation) comparing and contrasting it with RUP.

   iii. What is SAP?

2. Discuss the importance of the following on the software programmer/designer's performance:

   i. Organizational psychology:
   Discuss McGregor's theory of organizational psychology (Theory X vs. Theory Y).

   ii. Individual psychology:
   Give McClelland's method of rating an individual.

   iii. The programmer's self-perception
   Discuss how the reality differs from the perception.

3. i. Write a note on the Level-5 Object Knowledge-Based System describing its major components.

   ii. Briefly discuss Level-5 Object as a vehicle for implementing Object-Oriented specifications (such as CRC with RDD) with particular reference to the projects which you implemented.

   iii. Give its advantages and disadvantages.

   iv. What features would be desirable in a future version of Level-5 Object and why?

4. i. Discuss briefly ways for improving the Requirements Acquisition process.

   ii. Discuss what is meant by the "Tangibility Effect".

   iii. Briefly discuss CRC (Classes, Responsibilities and Collaborations) cards and comment on their use as a design tool. Give some guidelines for their use. What are their advantages and disadvantages? When are computer-based CRC cards most useful?

Section B

5. Using finite state techniques, design a lexical analyser for processing octal and hexadecimal constants described by the regular expression [0-9a-fA-F]+"@"(8|16).

   Note: i. If the value of the base is 8, the maximum value for any individual digit is 7.

   ii. A complete set of test inputs designed to visit all non-error entries in the transition table should be included with the design description.

6. Consider the following **LL(1)** arithmetic translation grammar with starting symbol **<S>**, where **const** is a token representing a numeric constant:

<S>	→	<E> **{answer}**
<E>	→	<T> <E-LIST>
<E-LIST>	→	+ <T> <E-LIST>
<E-LIST>	→	ε
<T>	→	<F> <T-LIST>
<T-LIST>	→	* <F> <T-LIST>
<T-LIST>	→	ε
<F>	→	<P> <F-PART>
< F-PART>	→	↑ <F>
< F-PART>	→	ε
<P>	→	(<E>)
<P>	→	**const**

   i. Add attributes to the productions so that the output action symbol **{answer}** will inherit a value equal to the numerical value of the expression generated by <s>.

   ii. Compute the selection set for each production, and design a recursive descent interpreter for the grammar.

7. i. In relation to pushdown processing, describe the prefix property and outline the difference between local and global error recovery.

   ii. Demonstrate how a well structured symbol table can cater for the overloading of symbols in a block structured language.

8. i. Describe the function and design of a simple register manager and demonstrate its use by generating object code for a machine with a single accumulator.

   ii. Design an attributed translation grammar for the following productions:

   <statement> → if <condition> then <statement>
   <statement> → if <condition> then <statement> else <statement>

   Describe clearly the information represented by the attributes and the function of the action symbols, and explain in detail how the productions may be processed by a pushdown machine.