UNIVERSITY OF DUBLIN

TRINITY COLLEGE

CS3BA71

Faculty of Engineering and Systems Sciences

School of Engineering

BA (Mod) Computer Science
Junior Sophister Examination

Trinity Term, 1998

3BA7 — Software Engineering and Compiler Design

Monday 8 June, 1998 Examination Hall 14:00 - 17:00

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. Discuss the Waterfall Software Life Cycle model of software development, defining the phases and using diagrams, including a cost curve, to illustrate your ideas. Give the advantages and disadvantages of the WSLC model.

   ii. The two fundamental user problems of requirement acquisition are:

   - Articulation of requirements and
   - Visualisation of the proposed system.

   Briefly discuss these problems with respect to the WSLC, and indicate several ways by which they can be reduced.

... / Contd

1 Contd / ...

iii. How do Object-Oriented Design methods such as Responsibility Driven Design (with Classes, Responsibilities, Collaborations) (RDD with CRC) overcome the disadvantages of the Waterfall Software Life Cycle.

iv. What automatic support would you like to see provided for RDD with CRC?

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

   i. Individual psychology — give McClelland's method of rating an individual.

   ii. The programmer's self-perception — discuss how reality differs from perception.

   iii. Organisational psychology — discuss McGregor's theory of organizational psychology (Theory X vs. Theory Y).

3. i. Discuss Mantei's three programming teams giving the management structure, communication channels, advantages and disadvantages of each. Summarize the characteristics of each in a table.

   ii. You have been appointed project manager of a project which is estimated to take three years and it is estimated that the final product will consist of 2 million source lines of code. Assuming a productivity rate of 10 source lines of completed code per day per programmer, estimate the number of programming teams needed, show how you would structure them and estimate the amount of administrative staff necessary.

4. i. Describe the formal steps used to identify classes in the Responsibility Driven Design approach.

ii. Given the following requirements definition and the list of classes arising from it, explain why the underlined classes should or should not be included in the candidate class list:

An automated teller machine (ATM) is a machine through which bank customers can perform a number of the most common financial transactions. The machine consists of a display screen, a bank card reader, numeric and special input keys, a money dispenser slot, a deposit slot and a receipt printer. When the machine is idle, a greeting message is displayed. The keys and deposit slot will remain inactive until a bank card has been entered. When a bank card is inserted, the card reader attempts to read it. If the card cannot be read, the user is informed that the card is unreadable, and the card is ejected. If the card is readable, the user is asked to enter a personal identification number (PIN). The user is given feedback as to the number of digits entered at the numeric keypad, but not the specific digits entered. If the PIN is entered correctly, the user is shown the main menu. Otherwise, the user is given up to two additional chances to enter the PIN correctly. Failure to do so on the third try causes the machine to keep the bank card. The user can retrieve the card only by dealing directly with an authorised bank employee.

numeric input key, feedback, display screen, key, element, digit, machine, numeric keypad, PIN, deposit slot, greeting message, automated teller machine, bank card, money dispenser slot, personal identification number, ATM, authorised bank employee, bank card reader, card, special input key, user, receipt printer, third try, failure, bank customer, chance, financial transaction, main menu

iii. Define what is meant by a Responsibility in RDD. Describe the steps necessary to identify Responsibilities. Give the guidelines for assigning Responsibilities.

iv. Define what is meant by a Collaboration in RDD. Give the steps for assigning Collaborations.

Section B

5. Using finite state techniques, design a simple lexical analyser to recognize XML tokens for *processing instructions* described by the following grammar; a complete set of test inputs (designed to visit all non-error entries in the transition table) should be included with the design description.

```
S           ::=   (#x20 | #x9 | #xD | #xA)+
Name        ::=   (Letter | '_' | ':') (NameChar)*
NameChar    ::=   Letter | Digit | '.' | '-' | '_' | ':'
PI          ::=   '<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'
PITarget    ::=   Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
```

Where `Letter` represents any upper or lower case character in the range `'a'` to `'z'`, `Digit` represents any digit in the range `'0'` to `'9'`, and `Char` represents any ascii character in the range `#x21` to `#xFF` or a sequence of spaces `S` as defined above.

For example, given the input:

```
<?MicrosoftWord "Replace 'sgml' 'xml'"?>
```

the lexical analyser should return the following token:

```
(ProcessingInstruction, MicrosoftWord, "Replace 'sgml' 'xml'")
```

6. Consider the following translation grammar with starting non-terminal **<E>**:

```
<E>  ->  <E> + <T>  {ADD}
<E>  ->  <T>
<T>  ->  <T> * <F>  {MULT}
<T>  ->  <F>
<F>  ->  <P> ↑ <F>  {POWER}
<F>  ->  <P>
<P>  ->  ( <E> )
<P>  ->  Ident
```

... / Contd

6 Contd / ...

where **Ident** is a lexical token representing a scalar variable, and the action symbols **{ADD}**, **{MULT}** and **{POWER}** represent calls to the code generator.

By eliminating left recursion, find an equivalent LL(1) grammar that generates the same language (sequences of terminal symbols), compute the selection set for each production in the grammar and finally add attributes and associated attribute evaluation rules so that the starting non-terminal will have a synthesized attribute describing the value of the arithmetic expression generated by **<E>**.

7. 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. Outline the differences between local and global error recovery during recursive descent parsing.

8. i. In relation to translation grammars, show by example the differences between inherited and synthesized attributes of non-terminal symbols. How are these two different types of attribute handled in a recursive descent parser?

   ii. Design an attributed translation grammar for a loop statement of the form:

   ```
   while <expression> do <statement>
   ```

   Describe fully and clearly the information represented by the attributes and the function of the action symbols.