Autumn Examinations 2010

Exam Code(s)	3IF1
Exam(s)	3rd B.Sc. (Information Technology)
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	I
External Examiner(s)	Prof. Michael O'Boyle
Internal Examiner(s)	Professor Gerard J. Lyons
	*Ms. Josephine Griffith
	*Dr. Jim Duggan

Instructions:

Answer 3 questions. All questions will be marked equally.
Use a separate answer book for each section. At least one question must be answered from each section.

Duration	2hrs
No. of Pages	5
Department(s)	Information Technology
Requirements	None

SECTION A

Q. 1. **(i)** With respect to programming paradigms, distinguish between the imperative and declarative programming paradigms, highlighting the main features of each paradigm. Use sample code from a language of each of the two paradigms to support your answer. *(10)*

(ii) With respect to program translation, and with the aid of a diagram, describe what is meant be a finite state automata (FSA). *(5)*

Illustrate, with the aid of a diagram, a FSA which recognises positive and negative real numbers, e.g., -32.1, 13.4, etc. *(8)*

(iii) The given grammar describes a restricted set of algebraic expressions. By parsing the sample string using this grammar:

```
a + b / c
```

obtain a parse tree and an abstract syntax tree, explaining the steps taken at each stage. *(10)*

```
Grammar = {N, T, S, P}
N = {<goal>, <expr>, <term>, <factor>}
T = {a, b, c, +, /}
S = <goal>

P=
<goal> ::= <expr>
<expr> ::= <term> | <term> + <expr>
<term> ::= <factor> | <factor> / <term>
<factor> ::= a | b | c
```

Q.2. **(i)** Describe the main features of the functional programming language SCHEME. *(5)*

(ii) With respect to the SCHEME primitives `car`, `cdr`, `cons`, `list` and `append`, explain what the output from each of the following SCHEME expressions is: *(12)*

(a) `(car '(a b c))`
(b) `(cdr '(a b c))`
(c) `(append '(a) '(b c))`
(d) `(list '(a b) '(c d) '(e f))`
(e) `(append '(a b) '(c d) '(e f))`
(f) `(cons '(a b) '(c d e))`
(g) `(car '((a b) (c d) (e f)))`
(h) `(cdr '((a b) (c d) (e f)))`

(iii) Control in SCHEME is mainly achieved through recursion. Distinguish between tail recursive and non-tail recursive functions by writing both a tail recursive and non-tail recursive version of a function in SCHEME that finds the sum of n numbers, e.g. for n with value 4:

```
(sum 4)  = 4 + 3 + 2 + 1
```

For both functions, explain the approach taken and explain the difference between the tail recursive and non-tail recursive versions. *(16)*

Q. 3. **(i)** Given the following sample database in PROLOG, identify, and distinguish between, the *facts*, *relations*, *rules* and *queries* in the PROLOG database. *(9)*

```
bitter(coffee).
likes(sara, tea).
likes(sue, coffee).
likes(john, X):-
     bitter(X).
?- likes(john, tea).
?- likes(john, coffee).
```

(ii) Describe, with the aid of the following example, the list data structure in PROLOG, explaining the representation and syntax: *(4)*

[a, b, c, d].

Indicate the unifications that occur for `H` and `T` given each of the following: *(6)*

(a) `[H|T] = [a, b].`
(b) `[H|T] = [a, b, c, d].`
(c) `[H|T] = [[a, b], [c, d]].`

(iii) Write PROLOG code to reverse the items (top level only) in a list, e.g., *(12)*

```
?- (reverse [a, b, c], R).
```

should return:

```
R = [c, b, a].
```

Explain the steps taken. *(2)*

SECTION B

Q. 4. **(i)** For an Account class, define a property that can set or retrieve the AccountNumber. Summarise the benefits of properties in object-oriented languages. (*9*)

(ii) Using delegates, write a publish/subscribe program where subscribers are updated with the latest football scores. The publisher should keep a list of delegates, and provide subscribers with a mechanism to register. When a score occurs, all subscribers should be notified automatically. (*24*).

Q. 5. **(i)** Explain the difference, using examples, between a class constructor and an object constructor. (9)

(ii) In C#, write a class method that takes in two integers, and returns (by reference) the sum, product, and difference of the two numbers. Explain why this would be difficult to implement in Java. (9)

(ii) Propose a solution that would ensure that only one object of a given type is created. Use the example of a **PrintSpooler**, which receives a request from an object that needs a printing service, and returns a reference to the single available **Printer** object. (*15*)

Q. 6. **(i)** Define an iterator, and summarise why it is a useful feature of C#. (6)

(ii) For an account with a collection of transaction objects, use the default iterator to return each transaction in reverse order from the collection. (15).

(iii) Extend the solution in (b) by developing custom iterators that implement the IEnumerator interface (e.g. public Object Current; public bool MoveNext(); public void Reset()). (12).