NUI Galway
OÉ Gaillimh

Semester 1 Examinations 2011 / 2012

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)	Prof. Gerard Lyons
	*Ms. Josephine Griffith
	*Dr. Jim Duggan

Instructions: Answer 3 questions.
Use a separate answer book for each section. At least one question must be answered from each section.

Duration	2hrs
No. of Pages	6 (Including cover page)
Department(s)	Information Technology
Requirements	None

SECTION A

Q1 **(a)** Distinguish between the SCHEME primitives `car` and `cdr` by writing sequences of `car`'s and `cdr`'s to extract the symbol "`and`" from the following expressions: *(6)*

(i)`(here is your crown and your seal)`
(ii)`(here is your (and here it is) here it is)`
(iii)`((here is your) (crown (and your)))`

(b) Distinguish between the SCHEME primitives `cons`, `list` and `append` by explaining the output from each of the following SCHEME expressionss: *(4)*

(i) `(cons 'x '(y z))`
(ii) `(list 'x '(y z))`
(iii) `(append '(x) '(y) '(z))`
(iv) `(cons 'a (append (list 'y) (list 'z)))`

(c) Write a recursive function in SCHEME which, when passed a list of numbers adds the numbers, e.g. if the function is named sum:
`(sum '(2 4 6 8))` returns `20`

Explain the approach taken, highlighting the base case and reduction stage. *(10)*

(d) Write a recursive function in SCHEME which performs a linear search of a list, given a list of symbols and a symbol to find. The function should return `#t` or `#f`, e.g. if the function is named `exists?`
`(exists? 'is '(here it is))` returns `#t`

Explain the approach taken highlighting the base case and reduction stage. *(10)*

Q2 Given the following PROLOG database:

```
sunny.
hot.
happy(ann).
likes(ann, sun).
likes(ann, books).
likes(ann, beach).
likes(kim, sun).
likes(kim, beach).
likes(X, holidays) :-
    likes(X, sun),
    likes(X, beach).
likes(ann, Y) :-
    likes(Y, holidays).
```

(a) Distinguish between facts, relations and rules in the PROLOG database given. *(4)*

(b) Show how unification occurs in PROLOG using the following query:

```
?likes(ann, holidays).
```

What is the output? *(4)*

(c) Show how unification occurs in PROLOG using the following query:

```
?likes(ann, Y).
```

What is the output? *(4)*

(d) Write the additional lines of code needed in the database to represent a person named john who likes sun and sport. *(4)*

(e) Describe, with the aid of examples, the list data structure in PROLOG, outlining its representation and syntax. *(4)*

(f) Write code in PROLOG to merge two lists, explaining the steps taken in developing the code. *(10)*

Q3 **(a)** Explain what is meant by a Finite State Automaton (FSA) by drawing an FSA to recognise strings of the form $a^r b^s$, where `r>0` and `s>=0` and with an alphabet of `{a,b}`, i.e. there must be at least one `a`, zero or more `b's` and all occurrences of `a` must preceed any occurrence of `b`. *(7)*
Illustrate, how your FSA works given the following sample strings: *(3)*

(i) `abbb`

(ii) `aaaa`

(ii) `aba`

(b) Given the following grammar :

G = {N, T, S, P}
T = {x, y, z}
N = {A, B, C}
S = A
P =
<A> ::= x<B>
<A> ::= x<C>
<B> ::= x<B>
<B> ::= y
<C> ::= x<C>
<C> ::= z

(i) Identify the terminals, non-terminals, productions and starting production in the grammar. *(2)*

(ii) What type of strings does the grammar generate and recognise? *(6)*

(iii) Draw the FSA that can be used to recognise the strings of the form identified in part (ii). *(4)*

(c) The following grammar describes a restricted set of assignment expressions

G = {N, T, S, P}
T = { =, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x, y}
N = {<R>, <E>, <id>, <num> }
S = R
P =

<R> ::= <id> = <E>
<E> ::= <D>| <E> - <D>
<D> ::= <id> |<num>
<num> ::= 0|1|2|3|4|5|6|7|8|9
<id> ::= x|y

Given the following string:

x = x-y-1

(i) Parse the string to obtain the associated parse tree and abstract syntax tree. *(6)*

(ii) By using the appropriate tree indicate if the string is valid in the grammar and the evaluation order of the string. *(2)*

SECTION B

Q4 (a) Distinguish between a value type and a reference type, and explain what the term "boxing" means in C#.

(5)

(b) Define a value type that can represent a three-dimensional point (with public properties X, Y and Z, and respective private attributes). Specify a Move operation that moves the point in all three dimensions. For this Move operation, a new point should be created and returned.

(15)

(c) For the value type defined in (b), override the "+" operator so that two 3D coordinates can now be added together using "+". Assume that this operation will add each x, y and z coordinate of the two operands.

Discuss the benefits of overriding operators in C#.

(10)

Q5 (a) Describe the purpose of the *Adapter Design Pattern*, show its overall structure, and discuss its advantages.

(6)

(b) The List<T> type in C# represents a strongly typed list of objects, and its methods include:

```
public void Add(T item);   // method
public bool Remove(T item);   // method
public T this[ int index] { get; set; }  //property to get/set values
public int Count { get; } // property to get number of elements
```

Use the Adapter Design Pattern to implement a generic MyStack<T> class, which extends this List<T> type, and implements a newly defined Stack<T> interface. The Stack<T> interface has the following definition (all 3 methods should be implemented).

```
interface Stack<T>
   {
      void Push(T item);
      T      Pop ( );
      int   Size ( );
   }
```

(18)

(c) Using the solution from part (b), write a short program that (i) creates a MyStack object of integers that is referenced by the Stack interface; (ii) pushes two integers onto the stack; (iii) displays the size of the stack and (iv) pops the two integers off the stack.

(6)

Q6 (a) Describe the advantages of regular expressions (using Regex type) in C#.

Explain the following sequences in Regex, and determine whether or not they would return a match for the string @"One two three 21 az"

\d{1,3}
\w+
\d{5}
[abc].

(10)

(b) Using a regular expression, write a method that receives a string, and returns the number of occurrences of vowels (uppercase and lowercase) in the string.

(10)

(c) Using a regular expression, write a function that validates an IP Address, where an IP Address has the following format: nnn.nnn.nnn.nnn, where n is usually a digit in the range 0-9, but no 3 digit sequence can exceed 255.

(5)

(d) Write a function, using regular expressions, that checks whether or not a string contains only uppercase characters (i.e. if all are uppercase, the function returns true, otherwise false is returned).

(5)