OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Autumn Examinations 2022-2023

Course Instance Code(s)	1OA2, 3BCT1, 3BS9, 4BS2
Exam(s)	3rd Year Examination Computing Science and IT
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1

External Examiner(s)	Dr. Ramona Trestian
Internal Examiner(s)	Professor Michael Madden *Dr. Finlay Smith

Instructions: Answer any 3 questions. All questions will be marked equally. Each question is worth a maximum of 25 marks. The total (out of 75) will be converted to a percentage after marking.

Duration	2 hours
No. of Pages	5
Discipline(s)	Computer Science
Course Co-ordinator(s)	Dr Colm O'Riordan

Release in Exam Venue	Yes [ ]	No [X]
MCQ Answersheet	Yes [ ]	No [X]
Handout	None	
Statistical/ Log Tables	None	
Cambridge Tables	None	
Graph Paper	None	
Log Graph Paper	None	
Other Materials	None	
Graphic material in colour	Yes [ ] No [X]	

PTO

1)

a) What are the two ways that memory management is handled in ‘C’? Describe the advantages and disadvantages of both of them.
(8 marks)

b) In your own words describe the effect of the following functions in ‘C’.
i) malloc()
ii) calloc()
iii) free()
(3 marks)

c) What is the difference between *void** and other pointer declarations? Give a simple example, including defining and assigning a value to a variable of type *void**.
(3 marks)

d) Write ‘C’ code that defines a structured type called *lectureRoom* with members that store the name of the room as a character array, the names of the modules that use the room as an array of pointers to strings and an array of pointers to integers that represents the number of students that take each of the modules and an integer which stores the number of modules that use the room. Write a function called *deleteRoom* that accepts a pointer to a *lectureRoom* instance and frees all of the memory associated with the structure.
(11 marks)

2)

a) How do you define and use a function pointer in ‘C’? Illustrate your answer with code snippets.
(5 marks)

b) Write a generic function in ‘C’ that prints the contents of an array of 20 elements. Your function should be passed a pointer to a print function that prints the elements of the type contained in the array (e.g., print an integer etc.) and then prints the elements in the array using that function. You should write a print function named printInt, and finally show you would call your generic function using this print function. What changes would be needed if the array was to contain floating point numbers instead of integers?
(20 marks)

<u>PTO</u>

3)

a) How does Lisp differ from other programming languages (such as ‘C’). What are the fundamental operations that are written in Lisp?
(3 marks)

b) In Lisp, what would the following function calls return?

i) (car (cdr (cdr (cdr ‘(2 3 4 5 6)))))
ii) (car (car ‘((2 1 1 2))))
iii) (car (car (cdr ‘(h (7 4) y y z)))
iv) (car (cdr (cdr ‘((4 (g l) a l) n p 3))))
(4 marks)

c) In Lisp, use car and cdr to return the following – in each case you can assume the lists have been stored in a variable called lst:
i) Element 9 in the list ‘(s (m p) 3 (3 9))
ii) List (7) in the list ‘(8 (((7) 5)) p w)
iii) Element a in the list ‘(2 (9 (a 3) 8) c)
iv) Element ‘(1 2) in the list ‘((5 6) (1 (((1 2)))))
(4 marks)

d) Write a non-tail recursive function in Scheme which takes 2 arguments (both lists) and returns a list of all of the numbers from the first list that are also on the 2nd list – the function should return each instance of such numbers in the first list. You can assume that each item in the list is a number and that there are no nested lists. For example, if the function is called in_both_lists:

(in_both_lists ‘(1 2 3 4 5 7 7 8 9 2 3 4 2 5 9) (2 4 5)) returns *(2 4 5 2 4 2 5)*
(6 marks)

e) Write a tail recursive version of your answer to part d). Make sure both your versions return lists with the elements in the same order.
(8 marks)

PTO

4)

a) Write a function in Lisp that takes a single integer argument, *N*. Your function should return a list of *N* elements – the numbers 1..N. For example, if the function is called NList:
(NList 5) returns (1 2 3 4 5).
(10 marks)

b) Write a tail recursive function in Scheme that accepts a list of numbers and returns a list with every element of the list multiplied by its position in the list. For example if the function is called *listMult*:

(listMult '(2 3 4 5 6)) returns (2 6 12 20 30) – e.g. (2*1 3*2 4*3 5*4 6*5)

(15 marks)

5)

a) Give an example of facts, rules and queries all having the same name and arity. In your own words describe why this can be useful when writing Prolog code.
(5 marks)

b) In your own words describe the Closed World Assumption. How is the Closed World Assumption used in Prolog? What effect does it have on the facts that need to be provided to Prolog programs? What would be required if the Closed World Assumption did not apply?
(6 marks)

c) Write a Prolog rule that finds the sum of all of the odd numbers in a list. You can assume that all of the elements in the list are numbers greater than 0. For example:

?- findSumOdd([1, 2, 3, 4, 5, 6, 7, 8], SumOdd).
SumOdd = 16
(14 marks)

PTO

6)

a) In your own words describe how lists are handled in Prolog. Use examples to illustrate your answer, showing the operations you can perform on lists.
(7 marks)

b) Write code in Prolog that doubles every even number in a list – the odd numbers are left alone. You can assume that all of the elements in the list are numbers. For example:

?-double_even([1, 2, 3, 4, 5, 6, 7], X).
X = [1 4 3 8 5 12 7]
(8 marks)

c) Write Prolog rules which take two lists as arguments. They should count the number of elements in the 2nd list that are equal to the last element in the 1st list. For example:

?-countEqualLast([4,5,6,7,8,9,3], [1,2,3,4,3,5,3,6,7,3,3], Count).
Count = 5
As 3 is the last element in the first list and there are 5 3's in the 2nd list.
(10 marks)

END

Autumn Examinations 2021/ 2022

Exam Code(s)	3BCT1
Exam(s)	3rd Year Examination Computing Science and IT
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	No
External Examiner(s)	Dr Ramona Trestian
Internal Examiner(s)	Professor Michael Madden *Dr. Finlay Smith

Instructions: Answer any 3 questions. All questions will be marked equally.

Duration	2 hours
No. of Pages	5
Discipline(s)	Computer Science
Course Co-ordinator(s)	

Requirements: None

PTO

1)

a) Describe the differences between using the stack and the heap to store data in ‘C’. Highlight the advantages and disadvantages of both of them, giving an example of when you would use each of them.
(9 marks)

b) In your own words describe the purpose of the following in ‘C’. Which of them take arguments and what are the arguments used for?
i) malloc()
ii) free()
iii) realloc()
iv) &ptr
(6 marks)

c) Write ‘C’ code that defines a structured type called *examPaper* with members that store the name of the module as a character array, the module number of the exam as a character array, the year of the exam as a number of type char, and the names of the students taking the module as a pointer to an array of strings. Write a function called *examPaper* that accepts a pointer to an *examPaper* instance and frees all of the memory associated with the structure.
(10 marks)

2)

a) How can function pointers be defined and used in ‘C’? Write code snippets to illustrate your answer. Your snippets should define and call a function that can read elements into a structure that contains a course code (a string of 5 characters) and an array of 4 floating point numbers. What changes would you need to make to the function call if the function was now to read in an array of 10 integers (as well as the other elements)?
(9 marks)

b) Write a generic sort function in ‘C’ that sorts an array of 20 strings. Your function should be passed a pointer to a comparison function (eg greater, less etc) and then sorts the array using that function. You should write a comparison function named greaterthan, and finally show how you would call your generic function using this comparison function.
(16 marks)

PTO

3)

a) Describe the features of a functional programming language. In particular what operations can be performed by such a language?
(3 marks)

b) In Lisp, what would the following function calls return?

i) (car (cdr ‘(1 2 3 4)))
ii) (cdr ‘(5))
iii) (car (car (cdr ‘(4 (5 6) 8 9))))
iv) (cdr (cdr (car ‘((9 8 7) a b c))))
(4 marks)

c) In Lisp, use car and cdr to return the following – in each case you can assume the lists have been stored in a variable called lst:
i) Element b in the list ‘(1 5 7 b 3 (4 5))
ii) Element 2 in the list ‘(7 (((5 2))) 3 4)
iii) Element 2 in the list ‘(9 (3 (1 2 4) 4) 0)
iv) Element ‘(1 2) in the list ‘((5 6) ((1 ((1 2)))))
(4 marks)

d) Write a non tail recursive function in Scheme which takes 2 arguments (a list of integers and an integer) and returns a list of all of the numbers in the list argument multiplied by the integer argument. You can assume that each item in the list is a number and that there are no nested lists. For example, if the function is called multiply:

(multiply ‘(2 4 6 8 1 3 5 7 3) 4) returns *(8 16 24 32 4 12 20 28 12)*

What changes would you need to make to your function if it was now to be able to accept floating point numbers?

(6 marks)

e) Write a tail recursive version of your answer to part d). Make sure both your versions return lists with the elements in the same order.
(8 marks)

PTO

4)

a) Write a function in Lisp that takes one argument, a list of n elements. Your function should return the list repeated *n* times. Each successive copy should have the last element removed. For example if the function is called duplicate: *(duplicate '(1 2 3 4))* returns (1 2 3 4 1 2 3 1 2 1).
(10 marks)

b) Write a tail recursive function in Scheme that accepts a list of numbers and returns a list with every 2nd number of the original list doubled. For example if the function is called *double_somet*:

(double_some '(1 2 3 4 5 6 7 8 10)) returns (2 2 6 4 10 6 14 8 20)

(15 marks)

5)

a) Give an example of facts, rules and queries all having the same name and arity. In your own words describe why this can be useful when writing Prolog code.
(5 marks)

b) In your own words describe the Closed World Assumption? How is the Closed World Assumption used in Prolog? What effect does it have on the facts that need to be provided to Prolog programs?
(5 marks)

c) Write a Prolog rule that finds the 2nd largest element in a list. You can assume that all of the elements in the list are numbers greater than 0. For example:

?- find SecondLargest([5, 9, 12, 4, 13, 22], Num).
Num = 13
(15 marks)

PTO

6)

a) In your own words describe how lists are handled in Prolog. Use examples to illustrate your answer, showing the operations you can perform on lists.
(7 marks)

b) Write code in Prolog that doubles every 2nd element in a list, you can assume that all of the elements in the list are numbers. For example:

?-double_list([1,2,3,4], X).
X = [1,4,6,8]
(8 marks)

c) Write Prolog rules which take three lists (of equal length) as arguments. The 3rd argument should have elements that are equal to the sum of the corresponding elements in the other two lists. For example the 1st element in the 3rd arguments value should be the sum pg the 1st elements in the other two lists. For example:

?-sumLists([1,2,3,4,5,6] , [7,8,9,10,11,12], Sum).
Sum = [8,10,12,14,16,18]
(10 marks)

Winter Examinations 2021/ 2022

Exam Code(s)	3BCT1
Exam(s)	3rd Year Examination Computing Science and IT
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	No
External Examiner(s)	Dr. Ramona Trestian
Internal Examiner(s)	Professor Michael Madden *Dr. Finlay Smith

Instructions: Answer any 3 questions. All questions will be marked equally. Each question is worth a maximum of 25 marks. The total (out of 75) will be converted to a percentage after marking.

Duration	2 hours
No. of Pages	5
Discipline(s)	Computer Science
Course Co-ordinator(s)	

Requirements: None

PTO

1)

a) How can you use the stack within ‘C’ code? Describe the steps involved – illustrate your answer with code snippets. What are the advantages and disadvantages of using the stack.
(9 marks)

b) In your own words describe the purpose of the following functions/operations in ‘C’.
i) sizeof()
ii) free()
(2 marks)

c) What are the differences between *malloc* and *calloc().*
(2 marks)

d) Write ‘C’ code that defines a structured type called *collegeStruct* with members that store the name of the College as a character array, the names of the Schools in the College as an array of pointers to strings and an array of integers that represents the number of students in each of the Schools. You can assume that there are 6 Schools in each College. Write a function called *deleteCollege* that accepts a pointer to a *collegeStruct* instance and frees all of the memory associated with the structure.
(12 marks)

2)

a) What are the advantages and disadvantages of using function pointers in ‘C’? Illustrate your answer with code snippets.
(5 marks)

b) Write a generic sort function in ‘C’ that sorts an array of 20 elements. Your function should be passed a pointer to a print function that prints the elements of the type contained in the array (e.g. print an integer etc.) and then prints the elements in the array using that function. You should write a print function named printInt, and finally show you would call your generic function using this print function.
(20 marks)

PTO

3)

a) Describe the features that distinguish Lisp from other programming languages. In particular what operations can be performed by Lisp?
(3 marks)

b) In Lisp, what would the following function calls return?

i) (car (cdr (cdr ‘(1 2 3 4 5))))
ii) (cdr ‘((4 5 6 7)))
iii) (car (cdr (cdr ‘(1 (3 4) 5 6 7)))
iv) (car (cdr (car ‘((d e f) 1 2 3))))
(4 marks)

c) In Lisp, use car and cdr to return the following – in each case you can assume the lists have been stored in a variable called lst:
i) Element b in the list ‘(1 2 (b 1) 3 (4 5))
ii) Element 2 in the list ‘(8 ((7 4)) 2 1)
iii) Element 2 in the list ‘(9 (3 (4 2) 3) 0)
iv) Element ‘(1 2) in the list ‘((5 6) (1 (((1 2)))))
(4 marks)

d) Write a non tail recursive function in Scheme which takes 2 arguments (a list and a number) and returns a list of all of the numbers in the list that are either equal to the number or twice the number, eg if the number is 5 your list should contain all instances of 5 and 10. You can assume that each item in the list is a number and that there are no nested lists. For example, if the function is called num_or_double:

(num_or_double ‘(2 4 6 8 1 3 5 7 4 3) 4) returns *(4 8 4)*
(6 marks)

e) Write a tail recursive version of your answer to part c). Make sure both your versions return lists with the elements in the same order.
(8 marks)

4)

a) Write a function in Lisp that takes one argument, a list of n elements. Your function should return the list repeated *n* times. Each copy of the list should have the first element removed. For example if the function is called duplicate: *(duplicate '(1 2 3 4))* returns (1 2 3 4 2 3 4 3 4 4).
(10 marks)

b) Write a tail recursive function in Scheme that accepts a list of numbers and returns a list with every 3rd number of the original list removed. For example if the function is called *shorten_list*:

(shorten_list '(1 2 3 4 5 6 7 8 10)) returns (1 2 4 5 7 8)

(15 marks)

5)

a) Give an example of facts, rules and queries all having the same name and arity. In your own words describe why this can be useful when writing Prolog code.
(5 marks)

b) In your own words describe the Closed World Assumption? How is the Closed World Assumption used in Prolog? What effect does it have on the facts that need to be provided to Prolog programs?
(5 marks)

c) Write a Prolog rule that finds the product of the largest and the smallest elements in a list. You can assume that all of the elements in the list are numbers greater than 0. For example:

?- findProduct([5, 9, 12, 4, 13, 22], Res).
Res = 88
as the smallest number is 4 and the largest number is 22
(15 marks)

PTO

6)

a) In your own words describe how lists are handled in Prolog. Use examples to illustrate your answer, showing the operations you can perform on lists.
(7 marks)

b) Write code in Prolog that triples every 3rd element in a list, you can assume that all of the elements in the list are numbers. For example:

?-triple_list([1,2,3,4,5,6,7], X).
X = [1,2,9,4,5.18,7]
(8 marks)

c) Write Prolog rules which take two lists as arguments. They should count the number of elements in the 2nd list that are less than the largest element in the 1st list. For example:

?-countNumber([4,5,6,7,8,9,3] , [1,2,3,4,5,6,7,8,9,10], Count).
Count = 8
As there are 8 numbers in the 2nd list that are less than 9 (the largest number in the 1st list).
(10 marks)

Semester 1 Examinations 2019/ 2020

Exam Code(s)	3BCT1
Exam(s)	3rd Year Examination Computing Science and IT
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	No
External Examiner(s)	Professor Jacob Howe
Internal Examiner(s)	Professor Michael Madden *Dr. Finlay Smith

Instructions: Answer any 3 questions. All questions will be marked equally.

Duration	2 hours
No. of Pages	4
Discipline(s)	Computer Science
Course Co-ordinator(s)	

Requirements: None

PTO

1)

a) In ‘C’, if a char requires 1 byte, an integer requires 4 bytes, a double requires 8 bytes and a pointer requires 8 bytes, how many bytes do variables of the following types require?

   *i) int **

   *ii) float **

   *iii) int ***

   *iv) char**

   v) char[10]*

   *vi) char**[10]*

   (6 marks)

b) In relation to ‘C’ answer the following questions using suitable examples:

   i) What are the advantages and disadvantages of using heap memory instead of stack memory in functions?(4 marks)

   ii) What is the purpose of type modifiers – do they always have an effect? (3 marks)

c) Write ‘C’ code that defines a structured type called *lecturerStruct* with members that store the name of the lecturer as a character array, the number of modules taught by the lecturer as an integer, the names of the modules taught by the lecturer as a pointer to an array of strings and the number of students taking each module as an array of integers. Write a function called *deleteLecturer* that accepts a pointer to a *lecturerStruct* instance and frees all of the memory associated with the structure. (12 marks)

2)

a) How can function pointers be defined and used in ‘C’? Write code snippets to illustrate your answer. (6 marks)

b) How can you create data structures in ‘C’ where the elements can be of any data type? Illustrate your answer with code snippets.(8 marks)

c) How can function pointers be used to write generic functions? Illustrate your answer with code snippets. What are the advantages of using function pointers? (11 marks)

3)

a) What are the differences between Lisp, Scheme and Racket? (3 marks)

b) In Lisp, what would the following functions return?

i) (list 1 2 3 4)
ii) (append ‘(a b) ‘(c d))
iii) (list 1 2 ‘(a b))
iv) (append ‘(1 2 3) 4)
(4 marks)

c) Write a non tail recursive function in Scheme which takes 2 arguments (a list and a number) and returns a list of all of the numbers in the list multiplied by the number. You can assume that each item in the list is a number and that there are no nested lists. For example, if the function is called multiply_list:

(multiply_list ‘(1 2 3 5 2) 3) returns (3 6 9 15 6)
(8 marks)

d) Write a tail recursive version of your answer to part c). Make sure both your versions return lists with the elements in the same order. (10 marks)

4)

a) How are Higher Order functions handled in Scheme? Do function definitions need to be changed to allow it to be used as a Higher Order function? Illustrate your answer with code snippets. (6 marks)

b) What is tail recursion in Scheme – what are the advantages and disadvantages of using it? (4 marks)

c) How does functional programming differ from sequential programming? (4 marks)

d) Write a tail recursive function in Scheme that accepts a list of numbers and returns a list with all of the numbers less than 10 doubled and all of the other numbers halved. For example, if the function is called double_half:

(double_half ‘(14 8 10 4)) returns (7 16 5 8)

(11 marks)

PTO

5)

a) Describe the differences and similarities between facts, rules and queries in Prolog. Illustrate your answer with examples. (6 marks)

b) What is the Closed World Assumption? How is the Closed World Assumption used in Prolog? What effect does it have on the facts that need to be provided to Prolog programs? (6 marks)

c) Write Prolog rules that find the sum of numbers that are less than 10 in a list (ignore any numbers that are not less than 10), for example:

?- sumSmallNumbers([2, 11, 13, 5, 7], Sum).
Sum = 14
(13 marks)

6)

a) Describe the similarities and differences between lists in Prolog and Scheme. Use examples to illustrate your answer. (6 marks)

b) Write code in Prolog that doubles the last element in a list. For example:

?-double_last([2,7,3,2], X).
X = [2,7,3,4]
(9 marks)

c) Write Prolog code which succeeds if all of the elements of its first list argument are not members of its second list argument. For example:

?-disjoint([4,7,3] , [1,2,7,1]).
no
?-disjoint([6,2,3], [5,1,7,8]).
yes

Would your code work if some of the elements of either list were also lists?
(10 marks)

Winter Examinations 2018/ 2019

Exam Code(s)	3BCT1
Exam(s)	3rd Year Examination Computing Science and IT
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	No
External Examiner(s)	Professor Jacob Howe
Internal Examiner(s)	Professor Michael Madden *Dr. Finlay Smith

Instructions: Answer any 3 questions. All questions will be marked equally.

Duration	2 hours
No. of Pages	4
Discipline(s)	IT
Course Co-ordinator(s)	

Requirements: None

PTO

1)

a) In ‘C’, what do the following types define? Define a variable for each of the types and assign it a suitable value.
   i) char
   *ii) char **
   *iii) char ***
   (6 marks)

b) In relation to ‘C’ answer the following questions using suitable examples:
   i) What are the differences between stack memory and heap memory? How are they accessed in ‘C’?(4 marks)
   ii) What is the purpose of the *typedef* keyword? (3 marks)

c) Write ‘C’ code that defines a structured type called *moduleStruct* with members that store the name of the module as a character array, the number of students taking the module as an integer, the names of the students taking the module as a pointer to an array of strings and the results for the students stored as an array of floats. Write a function called *deleteModule* that accepts a pointer to a *moduleStruct* instance and frees all of the memory associated with the structure. (12 marks)

2)

a) What are the differences between the type modifiers *short*, *long, signed* and *unsigned*. Do they always have an effect? (6 marks)

b) How can function pointers be useful? Write code snippets to illustrate your answer. (4 marks)

c) How can the function *sizeof()* be used to make writing generic functions easier? How could *sizeof()* be used to help make code platform independent? (6 marks)

d) How can function pointers be used to write generic functions? Illustrate your answer with code snippets. What are the advantages of using function pointers? (9 marks)

PTO

3)

a) What are the differences between Lisp, Scheme and Racket? (3 marks)

b) Describe the differences between the functions *append* and *list* in Scheme? (3 marks).

c) Write a non tail recursive function in Scheme which takes 2 arguments (a list and a number) and returns a list of all of the numbers in the list less than the number. You can assume that each item in the list is a number and that there are no nested lists. For example, if the function is called less_than:

(less_than '(2 4 6 8 10) 7) returns (2 4 6)
(8 marks)

d) Write a tail recursive version of your answer to part c). Make sure both your versions return lists with the elements in the same order. (11 marks)

4)

a) How are Higher Order functions handled in Scheme? How does this differ from 'C'? (6 marks)

b) What are the advantages and disadvantages of tail recursion in Scheme? (4 marks)

c) How does functional programming differ from sequential programming? (4 marks)

d) Write a tail recursive function in Scheme that accepts a list of numbers and returns a list with all of the odd numbers doubled and all of the even numbers left alone. For example, if the function is called double_odd:

(double_odd '(1 2 3 4)) returns (2 2 6 4)

(11 marks)

5)

a) Describe the differences and similarities between facts, rules and queries in Prolog. Illustrate your answer with examples. (4 marks)

b) How is the Closed World Assumption used in Prolog? What effect does it have on the facts that need to be provided to Prolog programs? (6 marks)

c) Write Prolog facts and rules that find the sum of odd number in a list, for example:

?- sumOddNumbers([1, 2, 3, 4, 5], Sum).
Sum = 9
(15 marks)

PTO

6)

a) Describe the similarities and differences between lists in Prolog and Scheme. Use examples to illustrate your answer. (6 marks)

b) Write code in Prolog that deletes the last element in a list. For example:

?-delete_last([1, 2, 3, 4, 5], X).
X = [1, 2, 3, 4]
(9 marks)

c) Write Prolog code which succeeds if all of the elements of its first list argument are members of its second list argument. For example:

?-subset([3, 2, 7] , [1, 2, 3, 4]).
no
?-subset([3, 2, 1], [1, 2, 3, 4]).
yes

Would your code work if some of the elements of either list were also lists?
(10 marks)

Semester 1 Examinations 2017/ 2018

Course Instance Code(s)	3BCT, 3BS9
Exam(s)	B.Sc. in Computer Science & Information Technology
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
External Examiner(s)	Dr. Jacob Howe
Internal Examiner(s)	*Aidan Breen Dr Michael Schukat

Instructions: Answer 4 questions.
Answer 1 question from section A.
Answer 3 questions from section B.
All questions in section B carry equal marks.

Duration	2 hours
No. of Pages	6
Department(s)	Information Technology
Course Co-ordinator(s)	Dr Des Chambers

PTO

Section A

Answer one of the questions from this section

All questions carry 30 marks

Question 1

A) What is meant by a Programming Paradigm?
(5 marks)

B) With the aid of examples from programming languages of your choice, distinguish between the Procedural and Functional programming paradigms.
(5 marks)

C) In the context of programming paradigms and programming languages, distinguish between Syntax and Semantics.

Using examples from programming languages of your choice, demonstrate a difference in syntax between two languages from different paradigms.
(5 marks)

D) Identify three influences on programming paradigms and briefly describe how each may affect a programming language.
(3 x 5 marks)

Question 2

A) What is meant by Imperative and Procedural programming?
(5 marks)

B) With the aid of examples from programming languages of your choice, distinguish between the *Functional* and *Declarative* programming paradigms.
(5 marks)

C) In the C programming language, what is meant by the term *pointer*?

Describe, using code examples, two common uses of pointers in C.
(2 x 5 marks)

D) What is meant by the term *Side Effect* in relation to procedural programming?

Write a C function that uses pointers to demonstrate a *side effect*, and a line of C code calling that function. You do not need to write an entire C program.
(10 marks)

Section B

Answer three of the questions from this section

All questions carry 25 marks

Question 3

A) Using examples (including code), describe what is meant by the Stack and the Heap in relation to the C programming language.
(4 marks)

B) In relation to the C programming language, answer the following questions using suitable examples:

When handling sensitive data, what extra steps should be taken using the library function free() to delete data?

What is the purpose of the typedef keyword?
(7 marks)

C) Write C code that defines a struct called personStruct with members that store a name as a character array, a birth year as an integer and a description as a pointer to a character array.

Write a C function called deletePerson that accepts a pointer to a personStruct instance and frees all of the memory associated with that struct.
(14 marks)

Question 4

A) Using examples, describe what is meant by the type modifiers *short, long, signed* and *unsigned* in relation to the C programming language.
(4 marks)

B) In relation to the C programming language, answer the following questions using suitable examples:

What is a struct?
What is the function of sizeof()?
(7 marks)

C) Implement a generic swap procedure in C that takes any fundamental data type (int, float, short, string) as arguments. Your swap( ) prototype should look like:

```
void swap(void * vp1, void * vp2, int size);
```

In particular, write C code to show how you would use it to swap two strings, explaining how you would call the swap procedure and illustrating what is happening in memory.
(14 marks)

Question 5

A) Distinguish between the Scheme primitives car and cdr by writing sequences of car and cdr to extract the item “twenty” from the following lists:

```
(this week her mother bought twenty cupcakes)

((this week her) mother (bought twenty cupcakes))

((this week her) (bought ((twenty)) cupcakes))
```

(4 marks)

B) Write a recursive function in Scheme which returns a list of all numbers lower than some value when passed a list of numbers and that value. You may assume each item in the list is a number and there are no nested lists. For example, if the function is called nums_below:

```
(nums_below ‘(1 2 3 4 5) 3) returns (1 2)

(nums_below ‘(1 2 3 4 5) 2) returns (1)
```

(7 marks)

C) Write both a non-tail recursive and a tail recursive function in Scheme which calculates the sum of all numbers in a list. You may assume each item in the list is a number and there are no nested lists. For example, if the function is called get_sum:

```
(get_sum ‘(5 5 5 1)) returns 16

(get_sum ‘(1 2 3 4 5 6)) returns 21
```

Explain the approach taken highlighting the base case and reduction stages for each function.
(14 marks)

Question 6

A) "The Functional programming paradigm aims to minimize side-effects." Explain why minimizing side-effects might be desirable and the limitations of strict side-effect free code.
(4 marks)

B) Using suitable examples and code, explain what is meant by a *well defined recursive function*. What are the benefits of a well defined recursive function? What are the drawbacks of a recursive function that is not well defined?
(7 marks)

C) Write a tail recursive function in Scheme that accepts a list of numbers, and returns a list of two numbers. The first number in the returned list should contain the sum of all even numbers in the input list. The second number in the returned list should contain the sum of all odd numbers in the input list. For example, if the function is called split_sum:

```
(split_sum '(1 2 3 4)) returns (6 4)

(split_sum '(1 1 1 4 4)) returns (8 3)
```

How could split_sum be changed to become a higher-order function, so that the input list can be split up in some way other than by even and odd numbers based on an input function reference?
(14 marks)

Question 7

A) With the aid of examples for each, distinguish between *facts*, *rules* and *queries* in Prolog.
(4 marks)

B) Describe what is meant by the term *unification.* In your answer, you should provide an example set of facts, rules and queries to demonstrate unification.
(7 marks)

C) Write a set of prolog facts and rules that count the number of times a given element appears in a given list of numbers. For example, if the rules were named 'countInstances':

```
?- countInstances([1, 2, 3, 3, 4, 3, 5], 3, X).
X=3.
```

(14 marks)

Question 8

A) Describe, with the aid of examples, the *list* data structure in Prolog, outlining its representation and syntax.
(5 marks)

B) Write code in Prolog to merge two lists, explaining the steps taken in developing the code.
(10 marks)

C) Write Prolog code which returns true if a given list of items is sorted in *descending* order. Otherwise it will return false. For example:

```
?- sorted([1,2,4,7]).
false.

?- sorted([4,3,2,1]).
True.
```

(10 marks)

END

Ollscoil na hÉireann, Gaillimh ***GX_____***

National University of Ireland, Galway

Semester I Examinations 2016/ 2017

Exam Code(s)	3BCT , 3BS9
Exam(s)	B.Sc. in Computer Science & Information Technology
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	
External Examiner(s)	Dr. John Power
Internal Examiner(s)	Dr. Jim Duggan *Dr. Hugh Melvin *Dr. Frank Glavin

Instructions: Use separate answer books for each section.
Answer Q1 and one other question from Section A
Answer any two questions from Section B.
All questions carry equal marks.

Duration	2 hours
No. of Pages	5
Department(s)	Information Technology
Course Co-ordinator(s)	Dr. Des Chambers

Requirements:
MCQ
Handout
Statistical Tables
Graph Paper
Log Graph Paper
Other Material

Section A

Answer Q1 and one other question from this section

Q.1 (a) Briefly explain each of the following 4 concepts, describing why they are potentially useful in programming and use code snippets in an appropriate language to illustrate and explain your answer:

(i) Lazy Evaluation
(ii) Closure
(iii) Lambda function
(iv) Currying

(4 x 5)

(b) Regarding the distinction between 1st class functions and higher order functions, an explanation on Stack Overflow states that “ "has first-class functions" is a property of a language, and "is higher-order" is a property of a function”.
Briefly explain this statement.

(5)

Q.2 (a) “The Functional programming paradigm minimise side effects” – explain what this means and why it is important? (5)

(b) Implement a generic swap procedure in C that takes any fundamental data type (int, float, short, string) as arguments. Your `swap( )` prototype should look like

```
void swap(void * vp1,void * vp2, int size)
```

In particular, write C code to show how you would use it to swap two strings, explaining how you would call the `swap` procedure and illustrating what is happening in memory.

(10)

(c)

Write C code to process a student’s grade. It must use 2 separate functions as follows:

- 1st function determines Grade i.e. returns a string. It receives the average of 3 subject results and uses following criteria for Grade:
  - >= 40% : Pass
  - < 40% : Fail
- 2nd function determines average of 3 subject grades (0-100) and returns average as an integer .

You must use function pointers to nest functions. In `main()` read in 3 subject results as floats. The definition of your final procedure called from `main()` should look like,

```
char* Cal_Grade(char*(*fn1)(int),int(*fn2) (float,float,float),
                float sub1,float sub2,float sub3)
{
return fn1(fn2(sub1,sub2,sub3));
}
```

(10)

Q.3 (a) Briefly describe the role of, and relationship between **generators** and **iterators,** using an example in python to illustrate your answer.
Outline also how these concepts are related to **lazy evaluation.**

(7)

(b) Write a scheme function that determines the minimum value in a list of numbers (You cannot use the built in (`apply min` …) !

(13)

(c) Write a python function that takes a list of numbers and returns those elements in the list that are evenly divisible by 9. You must use **filter** and **lambda** in your answer.

(5)

Section B
Answer any two questions from this section

Q.4

a) With the aid of examples for each, distinguish between *facts*, *relations*, *rules* and *queries* in Prolog.

(8 marks)

b) Describe what is meant by the term *unification.* In your answer, you should outline how constants, structures and variables can be unified in PROLOG.

(5 marks)

c) Describe the main features of a Logic Programming language.
Using an example, explain what is meant by the term *Negation-As-Failure.*
What are the differences between *forward chaining* and *backward chaining?*

(3 x 4 marks)

Q.5

a) Describe, with the aid of examples, the *list* data structure in PROLOG, outlining its representation and syntax. Write code in PROLOG to merge two lists, explaining the steps taken in developing the code.

(8 marks)

b) Explain what is meant by the term *tail recursion*. Write PROLOG code to reverse the items (top level only) in a list, writing **both** a tail recursive and non-tail recursive version of the code. You should explain how each version works.

(9 marks)

c) Describe how the “is” operator works in Prolog. Write Prolog code which returns true if a given list of items is sorted in *descending* order. Otherwise it will return false. For example:

```
sorted([1,2,4,7]).
        false
sorted([4,3,2,1]).
        true
```

(8 marks)

Q.6

a) Describe what is meant by each of the following with respect to program translation:

- Just-In-Time Compilation.
- Lexical Analysis.
- Error Recovery.
- Semantic Analysis.
- Peephole Optimisation.

(5 x 2 marks)

b) Explain what is meant by a Finite State Automaton (FSA). Draw an FSA to recognise strings that contain ‘acat’ as a substring with alphabet acgt

(8 marks)

c) The following grammar describes a restricted set of assignment expressions that only use addition:

```
G = {N, T, S, P}
T = { =, +, 0, 1, 2, 3, 4, 5, 6,7 8, 9, a, b}
N = {<R>, <E>,<D>, <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>     ::=a|b
```

Are the following strings valid in the above defined grammar?

- "b = 0"
- "a = a + b + 3"

You should create the corresponding parse trees for your validations

(7 marks)

NUI Galway
OÉ Gaillimh

Semester 1 Examination 2015 / 16

Exam Code(s)	3BCT1, 3BS9, 1EM1, 1SWB1
Exam(s)	3rd B.Sc. Computer Science and Information Technology 3rd Bachelor of Science 3rd B.Sc. (Information Technology) Erasmus Science Without Borders
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	
External Examiner(s)	Dr. John Power
Internal Examiner(s)	Prof. Gerard Lyons Dr. Michael Madden Dr.EndaHowley* Dr.ColmO'Riordan*

Instructions:

Candidates should answer **Three** questions, with at least one from each section.

Each section should be answered in a separate answer book.

All questions carry equal marks.

Duration 2 hours

No. of Pages 3

Requirements:
MCQ
Handout
Statistical/ Log Tables
Cambridge Tables
Graph Paper
Log Graph Paper
Other Materials

Release to Library: Yes □ No □

SECTION A

Q. 1

(a) Write a tail recursive scheme function that reverses a list of items. Note that if a sub-list is encountered, the contents of the sub-list do not need to be reversed. **(7)**

(b) Describe what is meant by a *higher order function* in scheme. Write a higher order function that can sum together a list of numbers such that only numbers that satisfy some criterion specified in a separate function are included in the summation.

For example the following function should only sum together the odd numbers.

(sum odd? '(3 4 5 6 8)) should return 8. **(8)**

(c) Show how a binary search tree can be represented in Scheme. Outline functions, in Scheme, to:

i) Search the binary search tree for a value
ii) Insert a value into the binary search tree. **(10)**

Q.2

(a) Explain what is meant by a push-down automaton (PDA) by drawing a PDA and any associated data structures to recognise strings of the form 1^n0^n, i.e., any sequence of 1's of length n, followed by a sequence of 0's of the same length n. Illustrate how the PDA works with suitable examples. **(8)**

(b) Write Prolog code that counts the number of occurrences of an item in a list. **(8)**

(c) Write Prolog code to merge two lists, sorted in ascending order, into a single list. Note that there should be no duplicates in the final list. **(9)**

SECTION B

Q.3.

(a) Outline the main benefits and features of using the Object Oriented Paradigm? **(3)**

(b) With the aid of suitable examples (including code), explain each of the following OO concepts using examples from C#:

- Inheritance,
- Polymorphism,

(10)

(c) Outline the core concept behind delegates in C#, and with the aid of a suitable code implementation show how they can be beneficial? **(12)**

Q.4.

(a) With the aid of code examples, outline the differences between a *class* and a *struct*? **(5)**

(b) With the aid of a suitable example, show how you would implement all of the following: **(10)**

- A full Interface Implementation
- Interface Inheritance
- A Class that implements Multiple Interfaces
- *explicit* and *implicit* interface implementations.

(c) Show with the aid of code, how you would implement the following in C# with the given database table *Students*: **(10)**

- Instantiate the SqlConnection.
- Open the connection.
- Pass the connection to other ADO.NET objects.
- Insert a new row of data into the *Students* table
- Delete a row of data from the *Students* table
- Update some existing data in the *Students* table
- Close the connection.

Table: Students

Student_ID	**Student_Name**	**DOB**	**Address**
00000001	Mary Murphy	1/1/1985	Newcastle, Galway
00000002	John Smith	5/3/1987	Salthill, Galway
00000003	Joe Franks	8/2/1990	Barna, Galway

NUI Galway
OÉ Gaillimh

Semester 1 Examination 2014

Exam Code(s)	3BCT1, 1EM1, 1SWB1,
Exam(s)	3rd B.Sc. Computer Science and Information Technology 3rd B.Sc. (Information Technology) Erasmus Science Without Borders
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	
External Examiner(s)	Dr. John Power
Internal Examiner(s)	Prof. Gerard Lyons Dr. Michael Madden Dr.EndaHowley* Dr.ColmO'Riordan*
Instructions:	Candidates should answer **Three** questions, with at least one from each section. Each section should be answered in a separate answer book. All questions carry equal marks.
Duration	2 hours
No. of Pages	3

Requirements:
MCQ
Handout
Statistical/ Log Tables
Cambridge Tables
Graph Paper
Log Graph Paper
Other Materials

Release to Library: Yes □ No □

SECTION A

Q. 1.

(a) Write a Scheme function, *count_occur* that counts the number of times an item occurs in a list of items; note that the list may contain sub-lists and you should also count occurrences in those sub-lists.

The function should take an item and a list as arguments. For example:
(count_occur 2 '(4 7 2 (4 2) 2)) should return 3.

(7)

(b) Describe what is meant by a *higher order function* in Scheme. Write a higher order function, *filter,* that can filter items from a list according to some criteria defined in a separate function (for example, *even?*). You may assume the list does not contain sub-lists.

(filter even? '(2 4 5 6 7)) should return *(2 4 6).*

(8)

(c) A *2-3 tree* is a generalisation of a binary search tree. In a *2-3 tree*, each node contains two values and three pointers to subtrees. The values in the node are sorted and the search tree properties hold for all sub-trees. Suggest a representation for *2-3 trees* in Scheme and outline scheme code to display the contents of 2-3 tree in sorted order.

(10)

Q.2.

(a) Explain what is meant by a push-down automaton (PDA) by drawing a PDA and any associated data structures to recognise strings of the form 1^n0^n, i.e., any sequence of 1's of length n, followed by a sequence of 0's of the same length n. Illustrate how the PDA works with suitable examples. **(8)**

(b) Using a list representation for sets, write Prolog code for the two following predicates:

i) *member(X,L)* which is true if and only if *X* occurs in *L*
ii) *subset(K,L)* which is true if and only if*K* is a subset of *L*

(9)

(c) Explain what is meant by the term *tail-recursive*. Write Prolog predicates to reverse a list. Comment on whether your predicates are tail-recursive. **(8)**

SECTION B

Q.3.

(a) Outline the main benefits and features of using an Object Oriented Paradigm? **(3)**

(b) With the aid of suitable examples (including code), explain each of the following OO concepts using examples from C#:

- Objects,
- Methods,
- Inheritance,
- Polymorphism,

(12)

(c) Outline the core concept behind delegates in C#, and with the aid of a suitable code implementation show how they can be beneficial? **(10)**

Q.4.

(a) Outline the main features of the .NET development environment.

(5)

(b) Show with the aid of code, how you would implement the following in C# with the given database table *Students*: **(20)**

- Instantiate the SqlConnection.
- Open the connection.
- Pass the connection to other ADO.NET objects.
- Insert new row of data into the *Students* table
- Delete a row of data from the *Students* table
- Update some existing data in the *Students* table
- Close the connection.

Table: Students

Student_ID	**Student_Name**	**DOB**	**Address**
00000001	Mary Murphy	1/1/1985	Newcastle, Galway
00000002	John Smith	5/3/1987	Salthill, Galway
00000003	Joe Franks	8/2/1990	Barna, Galway

NUI Galway
OÉ Gaillimh

Semester I Examinations 2013/2014

Exam Code(s)	3BCT1, 3IF1, 1EM1
Exam(s)	3rd B.Sc. Computer Science and Information Technology 3rd B.Sc. (Information Technology) Erasmus
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	1
Repeat Paper	
External Examiner(s)	Prof. Liam Maguire
Internal Examiner(s)	Prof. Gerard Lyons Dr. Michael Madden Dr. Enda Howley* Dr. Colm O'Riordan*

Instructions:

Candidates should answer **Three** questions, with at least one from each section.

Each section should be answered in a separate answer book.

All questions carry equal marks.

Duration 3 hours

No. of Pages 4

Requirements:
MCQ
Handout
Statistical/ Log Tables
Cambridge Tables
Graph Paper
Log Graph Paper
Other Materials

Release to Library: Yes □ No □

SECTION A

Q.1.

a) Explain what is meant by a Finite State Automaton (FSA) by drawing an FSA to recognise strings that have no consecutive 1s with an alphabet of {1,0}. For example

(i) 01101 is not a valid string
(ii) 1001 is a valid string
(iii)1 is a valid string
(iv)0 is a valid string

Illustrate how your FSA works given the above sample strings.

(7)

b) Given the FSA developed in part (a) suggest you would implement it in SCHEME.

(4)

c) Explain what is meant by a push-down automaton (PDA) by drawing a PDA and any associated data structures to recognise strings of the form 1^n0^n, i.e., any sequence of 1's of length n, followed by a sequence of 0's of the same length n. Illustrate how the PDA works with suitable examples.

(7)

d) With respect to grammars, explain the terms: *terminals*, *non-terminals*, *production rules*. What is meant by an *ambiguous* grammar? Give an example of such a grammar and explain the problems that may arise in using such a grammar.

(7)

P.T.O.

Q.2.

a) Distinguish between the SCHEME primitives *car* and *cdr* by writing sequences of *car* and *cdr* to extract the item “passing” from the following lists:

(3)

i) (and one day he came passing by)
ii) ((and one day) he (came passing by))
iii) ((and one day) (he came ((passing)) by))

b) Write a recursive function in SCHEME which performs a linear search of a list of numbers when passed a list and an item. You can assume that the data in the list is in ascending sorted order. The function should return #f or #t.
The function should stop searching when the item is found or when some number greater than the item is found or when the end of the list is reached without finding the item. For example, if the function is called linsearch:

```
(linsearch '10 '(2 4 6 8)) returns #f
(linsearch '4 '(2 4 6 8)) returns #t
```

Explain the approach taken, highlighting the base cases and the reduction stage.

(7)

c) Assume that a list represents a binary search tree. Write SCHEME code that performs a binary search on a tree and returns true (#t) if the value is found and returns false (#f) otherwise. Highlight both the base and reduction stages.

(7)

d) Write both a non-tail recursive and a tail-recursive function in SCHEME which counts the number of occurrences of an item in a list, returning this number. You can assume valid input and that there are no sub-lists. For example, if the function is named *countoccurs*:

(countoccurs 'a '(a b b a)) returns *2*

Explain the approach taken highlighting the base case and reduction stages for each function.

(8)

P.T.O.

SECTION B

Q.3.

(a) Outline the main benefits of using Object Oriented Languages?

(3)

(b) With the aid of suitable examples (including code), outline the main features of the Object Oriented paradigm. Your answer should make specific reference to each of the following:

- Objects,
- Methods,
- Inheritance,
- Polymorphism,

(12)

(c) Outline the core concept behind delegates in C#, and with the aid of a suitable code implementation show how they can be beneficial?

(10)

Q.4.

(a) What is meant by the Event Driven Paradigm?

(3)

(b) Design patterns are a significant aspect of C# Development. Outline the main features of Design Patterns and make specific reference to the following:

- Behavioral Design Patterns,
- Creational Design Patterns,
- Structural Design Patters,

(10)

(c) With the aid of suitable code examples, outline the main steps in managing and storing data in a C# environment. Make specific reference to:

- Managing a Database resource
- Serializable Objects.

(12)

Semester 1 Examinations 2012 / 2013

Exam Code(s)	3BCT1 3IF1 1EM1
Exam(s)	3rd B.Sc. Computer Science and Information Technology 3rd B.Sc. (Information Technology) Erasmus
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	I
External Examiner(s)	Prof. Michael O'Boyle
Internal Examiner(s)	Prof. Gerard Lyons Dr. Michael Madden *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	7 (Including cover page)
Department(s)	Information Technology
Requirements	None

PTO

OLLSCOIL NA hÉIREANN
NATIONAL UNIVERSITY OF IRELAND, GALWAY

SEMESTER I, WINTER 2012-2011 EXAMINATION

Third Year Examination in Computer Science and Information Technology

Programming Paradigms (CT331)

Prof. Michael O'Boyle
Prof. Gerard Lyons
Dr. Michael Madden
Ms. Josephine Griffith
Dr. Jim Duggan

Time Allowed: 2 hours

Answer THREE questions.
Use separate answer books.
At least ONE question must be answered from each section.

PTO

SECTION A

Q1 **(a)** Distinguish between the SCHEME primitives `car` and `cdr` by writing sequences of `car` and `cdr` to extract the number “2” from the following lists: *(3)*

(i) `(+ two and 2)`
(ii) `(and then (2 more) please)`
(iii) `(and (twice (2 3 4) plus two (too)))`

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

(i) `(cons '(a b) '(c d))`
(ii) `(list 'a 'b '(c d))`
(iii) `(append '(a b) '(c d) '(e f))`
(iv) `(list (list 'a) (cons 'b (append '(c) '(d) '(e))))`

(c) Write a recursive function in SCHEME which performs a linear search of a list of numbers when passed a list and an item. You can assume that the data in the list is in ascending sorted order. The function should return #f or #t. The function should stop searching when the item is found or when some number greater than the item is found or when the end of the list is reached without finding the item. For example, if the function is called linsearch:
`(linsearch '10 '(2 4 6 8))` returns `#f`
`(linsearch '4 '(2 4 6 8))` returns `#t`

Explain the approach taken, highlighting the base cases and the reduction stage. *(8)*

(d) Write both a non-tail recursive and a tail-recursive function in SCHEME which counts the number of occurrences of an item in a list, returning this number. You can assume valid input and that there are no sublists. For example, if the function is named countoccurs:
`(countoccurs 'a '(a b b a))` returns `2`

Explain the approach taken highlighting the base case and reduction stages for each function. *(10)*

PTO

Q2 Given the following set of relations in PROLOG which specify some courses taken by some students:

```
takes(john, ct101).
takes(kate, ct101).
takes(john, ct103).
takes(ciara, ct229).
takes(kate, ma101).
takes(shane, ct101).
takes(ciara, ct230).
```

(a) With the aid of the above relations in the PROLOG database answer the following questions:

(i) Show how unification occurs in PROLOG, and the resulting output, using the following query *(2)*:

```
?- takes(Y, ct101).
```

(ii) Show how unification occurs in PROLOG, and the resulting output, using the following query *(2)*:

```
?- takes(ciara, X).
```

(iii) Write the additional line(s) of code needed in the database to represent a student named `ann` who takes `ct101` and `ma190`. *(1)*

(iv) Add a rule to the database which states that two people are classmates if they take the same class. *(2)*

(b) Describe, with the aid of examples, the list data structure in PROLOG, outlining its representation and syntax. *(2)*

(c) Write code in PROLOG to merge two lists, explaining the steps taken in developing the code. *(6)*

(d) Write PROLOG code to reverse the items (top level only) in a list, writing a tail recursive and non-tail recursive version of the code. Explain the steps taken for both versions. *(10)*

PTO

Q3 **(a)** Explain what is meant by a Finite State Automaton (FSA) by drawing an FSA to recognise strings that have no consecutive 1s with an alphabet of `{1,0}`. For example

(i) `01101` is not a valid string
(ii) `1001` is a valid string
(iii) `1` is a valid string
(iv) `0` is a valid string

Illustrate how your FSA works given the above sample strings. *(8)*

(b) Given the following grammar :

G = {N, T, S, P}
T = {a, b, c}
N = {X, Y}
S = Y
P =
<Y> ::= a<X>
<X> ::= b<X>
<X> ::= c

(i) Identify the terminals, non-terminals, productions and starting production in the grammar. *(1)*
(ii) What type of strings does the grammar generate and recognise? *(2)*
(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 using the operators of addition (+) and multiplication (×).

G = {N, T, S, P}
T = { =, +, ×, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b}
N = {<R>, <E>, <id>, <num> }
S = R
P =

```
<R>     ::= <id> = <E>
<E>     ::= <id> | <num>
<E>     ::=  <E> × <E> | <E> + <E>
<num> ::= 0|1|2|3|4|5|6|7|8|9
<id>     ::= a|b
```

Given the following string:

a = a + b × 3

(i) Show how two different parse trees are possible when parsing this string using the above grammar. *(5)*
(ii) Modify the grammar to correct the problem and, using the same sample string, show how only one parse tree results from parsing the string with the new grammar. *(5)*

PTO

SECTION B

Q4 (a) Describe the key idea of a delegate, and show how it is defined in C#.

(5)

(b) Define a delegate that can be used to send information on a sports event to subscribers. The information is specified as a *value type* with the following fields: Event ID, Time, Team Description, Scorer.

(8)

(c) Based on (b), implement a publisher/subscriber solution whereby subscribers can join/leave a publisher service, and are notified automatically once an event has happened. Make use of a List structure to store all subscribers. A list of all events should also be stored by the publisher, on a last-in first-out basis.

(12)

Q5 (a) Describe the purpose of the *Command Pattern*, show its overall structure, and discuss its advantages. As part of the answer, include a class diagram and a collaboration diagram.

(10)

(b) Using the Command Pattern, implement a solution in C# for the following stock control transactions.

- Create a class called StockListener, which creates an **Invoker**, and calls a **Client** with information when a stock change (increase or decrease) has happened. This information includes the StockId(String) and the amount it has changed by (integer). The **Client** will return an ICommand object (supertype), which the **StockListener** then passes on to the **Invoker**. [The Client also creates a **Receiver** object, and passes this to the concrete ICommand object].

- The **StockListener** has two methods called *ProcessBatch()* and *Rollback()*, and these methods call the Invoker to carry out these operations. The Invoker should arrange for the logic of both batch updates and complete rollback of all transactions.

- For the **Invoker**, use a List class to store information on all ICommand objects. All commands should be able to (1) Execute and (2) Rollback.

- Two concrete commands objects should be created, one to increment the stock, the other to decrement the stock. All stock changes are controlled with the **Receiver** object.

(15)

PTO

Q6 (a) Explain the following RegEx sequences:

- [abc]
- [^abc]
- [a-z]
- \d
- \D

(8)

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

(5)

(c) Summarise the main benefits of overriding operators in C#.

Create a class that contains a one-dimensional array of integers. Override both the + and - operators so that:

- An array can be added to the current array
- An array can be subtracted from the current array

Also, a complete list of all previous states of the array should be stored.

(12)

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)

Semester 1 Examinations 2010 / 2011

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)	*Dr. Jim Duggan Mr. Alan Cunningham

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	6 (Including cover page)
Department(s)	Information Technology
Requirements	None

OLLSCOIL NA hÉIREANN
NATIONAL UNIVERSITY OF IRELAND, GALWAY

SEMESTER I, WINTER 2010-2011 EXAMINATION

Third Year Examination in Information Technology

Programming Paradigms (CT331)

Prof. Michael O'Boyle
Dr. Jim Duggan
Mr. Alan Cunningham

Time Allowed: 2 hours

Answer THREE questions.
Use separate answer books.
At least ONE question must be answered from each section.

SECTION A

Q1 **(i)** With respect to features of the functional programming paradigm, and with the with the aid of examples, describe what is meant by each of the following statements:

(a) Programs are constructed as a composition of functions. (4)
(b) Control is mainly achieved through recursion. (3)
(c) Theoretically, functions have no side effects. (3)

(ii) Distinguish between the SCHEME primitives cons, list and append by explaining what the output from each of the following SCHEME expressions is: (6)

(a) (cons ‘a ‘(nice example))
(b) (list ‘a ‘(nice example))
(c) (append ‘(a) ‘(nice example))
(d) (list ‘(a b) ‘(c d) ‘(e f))
(e) (append ‘(a b) ‘(c d) ‘(e f))
(f) (cons ‘(a b) ‘((c d e)))

(iii) Distinguish between the SCHEME primitives car and cdr by writing sequences of cars and cdrs to extract the symbol “and” from the following expressions: (4)

(a) (this and that them they those)
(b) (those they (this and that) then)

(iv) Write a function in SCHEME which, when passed a list and a number, adds the number to the first element of the list if that element is a number, otherwise it returns the list unchanged. (8)
e.g. if the function is named sum_it:

(a) (sum_it 3 ‘(1 2 a b)) returns (4 2 a b)
(b) (sum_it 3 ‘(a b 1 2)) returns (a b 1 2)
Explain the approach taken. (2)

Q2 **(i)** Describe the main features of the logic programming paradigm. (4)

With the aid of examples, distinguish between facts, relations, rules and queries in PROLOG. (4)

With the aid of an example, show how implicit and explicit unification occurs in PROLOG. (2)

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

Write code in PROLOG to merge two lists, explaining the steps taken in developing the code. (6)

(iii) Write PROLOG code to reverse the items (top level only) in a list, writing a tail recursive and non-tail recursive version of the code. (8)

Explain the steps taken for both versions. (2)

Q3 **(i)** Describe what is meant by each of the following with respect to program translation:

(a) Just In Time Compilation. (2)
(b) Lexical Analysis. (2)
(c) Error Recovery. (2)
(d) Semantic Analysis. (2)
(e) Peephole Optimisation. (2)

(ii) Explain what is meant by a Finite State Automaton (FSA). (2)

Illustrate, with the aid of a diagram, an FSA which combines automata to recognise the following lexical tokens: (8)

(a) if keyword
(b) positive real numbers
(c) all strings that contain “acat” as a substring, with an alphabet of acgt

(iii) The following production rules, P1 and P2, are two alternative production rules for P in the grammar given which should describe a restricted set of algebraic expressions. By parsing the sample strings:

a + b × c
a × b × c

obtain parse trees and discuss the problems, if any, with P1 and P2: (10)

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

P1 =
<goal> ::= <expr>
<expr> ::= <term> | <term> × <expr>
<term> ::= <factor> | <factor> + <term>
<factor> ::= a | b | c

P2 =
<goal> ::= <expr>
<expr> ::= <expr> + <expr>| <expr> × <expr>| <factor>
<factor> ::= a | b | c

SECTION B

Q4 **(i)** For an Account class, define a property that can set or retrieve the AccountNumber (as a String). Summarise the benefits of properties in object-oriented languages. (*6*)

(ii) Define the format of a delegate, and explain how it is used in C#. Highlight the difference between an event and a delegate. (*6*)

(iii) Using delegates, write a publish/subscribe program where subscribers are updated with the latest weather information. The publisher should keep a list of delegates, and provide subscribers with a mechanism to register their interest in a particular city. When the weather changes, all subscribers to that city should be notified automatically. A simple data structure (struct) should be used to pass information, and it should capture important weather-related information (*18*).

Q5 **(i)** Examine the following code segment, and determine what output will be printed from each statement. (*6*)

```
String a = “HelloWorld”;
String b = “HelloWorld”;
String c = “HELLOWORLD”;

Console.WriteLine(a == b);
Console.WriteLine(b==c);
```

(ii) Explain the use of regular expressions in C#, and show give a simple example of a function that determines whether a string is uppercase. (*6*)

(iii) Using a regular expression, write a function that validates a mobile phone number. The format of the number is as follows:

Mobile Phone Number = Start Character Symbol + Country Code + Hyphen + Prefix + Hyphen + Main Number
Start Character Symbol = +
Country Code = 1{digit}3 // between 1 and three digitas
Prefix = 2{digit}2 // three digits
Hyphen = -
Main Number = 7{digit}7
Digit = [0-9]

For example, a valid phone number would be +353-86-1111111. *(18)*

Q6 **(i)** Explain the difference between interface and implementation inheritance. (*8*)

(ii) Summarise the key ideas behind the adapter design pattern. Use a class diagram to show its structure, and a collaboration diagram to highight its behaviour. (*8*)

(iii) Use the adapter design pattern to create a Last in First Out (LIFO) container object, which is “wrapped” around C#’s ArrayList class. The interface for the LIFO list is:

```
public interface LIFO
{
        public void insert (Object o);
        public Object getFirstElement();
        public Object removeFirstElement();
        public int getListSize();
}
```

For the ArrayList, the following property and methods are relevant. Also, an index should be used to retrieve a value from an ArrayList. For example, the first element of an ArrayList x is x[0].

Properties:
Count – Returns the number of elements in the ArrayList

Methods
Add – Add an object to the end of the ArrayList *(14)*

NUI Galway
OÉ Gaillimh

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).

Semester 1 Examinations 2009 / 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

OLLSCOIL NA hÉIREANN
NATIONAL UNIVERSITY OF IRELAND, GALWAY

SEMESTER I, WINTER 2009-2010 EXAMINATION

Third Year Examination in Information Technology

Programming Paradigms (CT331)

Prof. Michael O'Boyle
Prof. Gerard J. Lyons
Ms. Josephine Griffith
Dr. Jim Duggan

Time Allowed: 2 hours

Answer THREE questions.
Use separate answer books.
At least ONE question must be answered from each section.

SECTION A

Q. 1. (i) Describe the main features of the logic programming paradigm. *(4)*

With the aid of examples, distinguish between *facts*, *relations*, *rules* and *queries* in PROLOG. *(4)*

With the aid of an example, show how implicit and explicit unification occurs in PROLOG. *(2)*

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

Write code in PROLOG to merge two lists, explaining the steps taken in developing the code. *(6)*

(iii) Write PROLOG code to reverse the items (top level only) in a
list, writing a tail recursive and non-tail recursive version of the code. *(8)*

Explain the steps taken for both versions. *(2)*

Q.2. **(i)** With the aid of an example, explain what is meant by “fully parenthesised prefix notation” with respect to the functional programming language SCHEME. *(4)*

(ii) Distinguish between the SCHEME primitives `cons`, `list` and `append` by explaining what the output from each of the following SCHEME expressions is: *(6)*

```
(a)    (cons ‘a ‘(nice example))
(b)    (list ‘a ‘(nice example))
(c)    (append ‘(a) ‘(nice example))
(d)    (list ‘(a b) ‘(c d) ‘(e f))
(e)    (append ‘(a b) ‘(c d) ‘(e f))
(f)    (cons ‘(a b) ‘((c d e)))
```

(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 factorial (n!) of an integer number *n* where:

n! = n × n-1 × n-2 × …. × 1

For both functions, explain the approach taken and explain the difference between the tail recursive and non-tail recursive versions. *(10)*

(iv) Write a function in SCHEME which checks if two occurrences of a given list occur beside each other in a second list, returning `#t` or `#f` *(6)*
e.g.,

```
 (two_conseq? ‘(a b) ‘( (x y) (a b) (a b)))
      returns #t

(two_conseq? ‘(a b) ‘((a b)  (x y) (a b)))
      returns #f
```

Explain the approach taken. *(4)*

Q. 3. **(i)** What is meant by a programming paradigm? *(4)*

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. *(6)*

(ii) With respect to program translation describe the purpose of the lexical analysis stage. *(2)*

Illustrate, with the aid of a diagram, a FSA which recognises strings of the form `c.(a|d).(a|d)*.r`, e.g., the FSA should recognise `car`, `cdr`, `cadr`, etc. *(8)*

(iii) With respect to program translation describe the purpose of the syntactic analysis stage. *(2)*

The given grammar describes a restricted set of algebraic expressions. By parsing the sample string:

```
x + y × z
```

obtain a parse tree and highlight the problems, if any, with the grammar. *(8)*

```
Grammar = {N, T, S, P}
N = {<goal>, <expr>, <term>, <factor>}
T = {x, y, z, +, ×}
S = <goal>

                        P=
<goal> ::= <expr>
<expr> ::= <term> | <term> × <expr>
<term> ::= <factor> | <factor> + <term>
<factor> ::= x | y | z
```

SECTION B

Q. 4. **(i)** For a Student class, define a property that can set or retrieve the StudentNumber. Summarise the benefits of properties in object-oriented languages. (*6*)

(ii) Define the format of a delegate, and explain how it is used in C#. Highlight the difference between an event and a delegate. (*6*)

(iii) 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. (*18*).

Q. 5. **(i)** Explain the difference, using examples, between a class constructor and an object constructor. (*6*)

(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. (*12*)

(iii) Expand on the solution in (ii), so that the PrinterSpooler can keep track of 10 printers, and it returns a reference to the first available printer. (*12*)

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

(ii) For a Student (ID, Name) object with a collection of Result (Subject Code, Grade) objects, use the default iterator to return each result object from the collection. (*15*)

(iii) Extend the solution in **(ii)** by developing custom iterators that implement the IEnumerator interface (e.g. public Object Current; public bool MoveNext(); public void Reset()). These should return the results in descending order. Assume that – if needed – a sorting object is available that will sort a list of results into descending order [i.e. there is no requirement to implement a sorting algorithm]. (*9*).

Ollscoil na hÉireann, Gaillimh **_GX_____**

National University of Ireland, Galway

Autumn Examinations 2008 / 2009

Exam Code(s)	3IF1
Exam(s)	Third B.Sc. (Information Technology)
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	I
External Examiner(s)	Prof. J.A. Keane
Internal Examiner(s)	Prof. Gerard Lyons Dr. Jim Duggan

Instructions: Answer any THREE Questions

Duration	2 HOURS
No. of Pages	3
Department(s)	Information Technology
Course Co-ordinator(s)	Dr. Des Chambers

Requirements:

OLLSCOIL NA hÉIREANN
NATIONAL UNIVERSITY OF IRELAND, GALWAY

AUTUMN 2008-2009 EXAMINATION

Third Year Examination in Information Technology

Programming Paradigms (CT331)

Professor J.A. Keane
Prof. Gerard J. Lyons
Dr. Jim Duggan

Time Allowed: 2 hours

Answer any THREE questions

1. (a) On the .NET platform, distinguish between *managed code* and *unmanaged code*. (40% of marks).

   (b) For a Student class, define a property that can set or retrieve the *studentID.* Summarise the benefits of properties in object-oriented languages. (30% of marks).

   (c) In C#, write a class method that takes in three integers, and returns (by reference) the sum and product of the numbers. Explain why this would be difficult to implement in Java. (30% of marks).

2. (a) Describe an event, and explain how an event can play an important role in building modern software applications. Distinguish between an event and a delegate. (30% of marks).

   (b) Make use of events to code a simple publisher/subscriber system that broadcasts football scores to registered clients. Subscribers should have a way of registering, and as they register, they should provide a callback function. This function will receive the update in string format. The publisher should keep an archive of all football scores, using an ArrayList. (70% of marks).

3. (a) Draw a diagram showing the main classes in the .NET Reflection API, and describe the main purpose of each class. (30% of marks).

   (b) For a given Type t, show how you would list (1) all methods of the type, (2) all constructors of the type. (40% of marks).

   (c) Given a string representation of a class, clearly show the steps you would take in order to instantiate an object of that class. Outline the advantages that this facility provides for software developers. (30% of marks).

4. (a) Define an iterator, and summarise why it is a useful feature of C#. (25% of marks).

   (b) For a student with a collection of exam result objects, use the default iterator to return each result in reverse order from the collection. (35% of marks).

   (c) 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()). (40% of marks).

5. Use an abstract class to create a proxy solution for access to a Bank Account object. The Bank Account (attributes Id, Password and Balance) can be viewed as the original, and the proxy contains a reference to the original. When a *GetBalance(Id, Password)* request arrives at the proxy, the id and password are authorised. If the authorisation fails, the original is not invoked. If the authorisation is a success, the original is invoked, and the balance passed back. As well as coding the solution, show the design using a class diagram.

Ollscoil na hÉireann, Gaillimh **GX_____**

National University of Ireland, Galway

Semester I Examinations 2008 / 2009

Exam Code(s)	3IF1
Exam(s)	Third B.Sc. (Information Technology)
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	I
External Examiner(s)	Prof. J.A. Keane
Internal Examiner(s)	Prof. Gerard Lyons Dr. Jim Duggan

Instructions: Answer any THREE Questions

Duration	2 HOURS
No. of Pages	3
Department(s)	Information Technology
Course Co-ordinator(s)	Dr. Des Chambers

Requirements:

OLLSCOIL NA hÉIREANN
NATIONAL UNIVERSITY OF IRELAND, GALWAY

SEMESTER I, WINTER 2008-2009 EXAMINATION

Third Year Examination in Information Technology

Programming Paradigms (CT331)

Professor J.A. Keane
Prof. Gerard J. Lyons
Dr. Jim Duggan

Time Allowed: 2 hours

Answer any THREE questions

1. (a) On the .NET platform, distinguish between *managed code* and *unmanaged code*. (40% of marks).

   (b) For a Bank Account class, define a property that can set or retrieve the *accountNumber*. Summarise the benefits of properties in object-oriented languages. (30% of marks).

   (c) 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. (30% of marks).

2. (a) Describe an event, and explain how an event can play an important role in building modern software applications. Distinguish between an event and a delegate. (30% of marks).

   (b) Make use of events to code a simple publisher/subscriber system that broadcasts headline news items to registered clients. Subscribers should have a way of registering, and as they register, they should provide a callback function. This function will receive the update in string format. The publisher should keep an archive of all news events, using any standard collection object. The collection object should be declared using generics. (70% of marks).

3. (a) Draw a diagram showing the main classes in the .NET Reflection API, and describe the main purpose of each class. (30% of marks).

   (b) For a given Type t, show how you would list (1) all methods of the type, (2) all constructors of the type, and (3) all the static fields of the type. (40% of marks).

   (c) Given a string representation of a class, clearly show the steps you would take in order to instantiate an object of that class. Outline the advantages that this facility provides for software developers. (30% of marks).

4. (a) Define an iterator, and summarise why it is a useful feature of C#. (20% of marks).

   (b) For an account with a collection of transaction objects, use the default iterator to return each transaction in reverse order from the collection. (30% of marks).

   (c) 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()). (50% of marks).

5. (a) Use an abstract class to create a proxy cache solution for a collection lookup. The collection (e.g. an ArrayList of Student objects) can be viewed as the original, and the proxy contains a reference to the original, as well as a simple array cache of N objects, where N is defined at startup, and expected to be far smaller than the size of the original collection.

   Each object in the collection will have a unique identifier (student id). When a request arrives at the proxy, if the object is on the cache, it is returned. If the object is not on the cache, it is retrieved from the collection, placed in the cache, and returned. If the cache is full, the new object will replace the oldest object on the cache.

   (b) Describe the key idea and advantage of asynchronous delegates, and outline, using a simple example, how they can be used in C#. (30% marks)

Ollscoil na hÉireann, Gaillimh ***GX_____***

National University of Ireland, Galway

Semester I Examinations 2007 / 2008

Exam Code(s)	3IF1 1EM1
Exam(s)	3rd Year Examination in Information Technology Erasmus
Module Code(s)	CT331
Module(s)	Programming Paradigms
Paper No.	
Repeat Paper	
External Examiner(s)	Professor J.A. Keane
Internal Examiner(s)	Professor G. Lyons Ms. J. Griffith

Instructions: Answer **THREE** questions.
All questions carry equal marks.

Duration	**2 hours**
No. of Pages	5
Department(s)	Information Technology
Course Co-ordinator(s)	

Requirements:
MCQ
Handout
Statistical Tables
Graph Paper
Log Graph Paper
Other Material

Q. 1. (i) What is meant by a *programming paradigm? (4)*

With the aid of examples from programming languages of your choice, distinguish between the imperative and declarative programming paradigms. *(6)*

(ii) Describe what is meant by syntax and semantics. *(4)*

With respect to syntax, what syntactic elements do the languages Python and VB.NET use to group code into blocks? Provide sample code fragments to illustrate your answer. *(6)*

(iii) VB.NET is an example of an event-driven paradigm. Using examples from VB.NET, describe the main features of the event-driven paradigm. *(4)*

Describe the three main approaches to event processing that can be taken by an event-driven language. *(6)*

Q. 2. (i) Describe the main features of the logic programming paradigm. *(4)*

With the aid of examples, distinguish between *facts*, *relations*, *rules* and *queries* in PROLOG. *(4)*

(ii) Control in SCHEME and PROLOG is mainly achieved through recursion. With the aid of an example in SCHEME describe: *(2)*

(a) what is meant by recursion? *(3)*
(b) what is the difference between tail recursion and non-tail recursion? *(3)*
(c) what is the role of the call stack (run time stack) when using non-tail recursive functions? *(2)*
(d) show the contents of the call stack (run time stack) for a small sample run of the recursive code you develop. *(2)*

(iii) With the aid of an example, describe a representation of binary trees in SCHEME. *(3)*
Write code in SCHEME to implement a depth first search using the binary tree representation you developed. *(5)*
Explain the approach taken. *(2)*

Q.3. (i) Describe the main features of the functional programming paradigm. *(2)*

With the aid of an example, explain what is meant by "fully parenthesised prefix notation" with respect to the functional programming language SCHEME. *(2)*

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

(a) `(this and that them they those)`
(b) `(those they (this and that) then)`
(c) `((then those) they (this and that))`

(ii) Distinguish between the two main data objects in SCHEME: atoms and lists. *(4)*

Distinguish between the SCHEME primitives `cons`, `list` and `append` by explaining what the output from each of the following SCHEME expressions is: *(6)*

(a) `(cons 1 ‘(a b c))`
(b) `(list 1 ‘(a b c))`
(c) `(append ‘(1) ‘(a b c))`
(d) `(list ‘(3 2) ‘(1 2 3))`
(e) `(append ‘(3 2) ‘(1 2 3))`
(f) `(cons ‘(3 2) ‘(((1 2 3))))`

(iii) Write a function in SCHEME which, when passed a list and a number, adds the number to the first element of the list if that element is a number, otherwise it returns the list unchanged. *(8)*
e.g. if the function is named `sum_it`:

```
(sum_it 3 ‘(1 2 a b)) returns (4 2 a b)
(sum_it 3 ‘(a b 1 2)) returns (a b 1 2)
```

Explain the approach taken. *(2)*

Q. 4. (i) Distinguish between strong and weak typed languages using the following as an example which shows sample output in a strong and weak typed language: *(8)*

```
Python >>>> 2+ “day”
→ TypeError: unsupported operand type(s) for +: ‘int’
  and ‘str’

Perl:DB<1> 2+“day”
→ 2day
```

(ii) Explain the difference between collateral, sequential and recursive bindings. *(6)*

Given the following code fragments in SCHEME, distinguish between the binding times in the SCHEME special forms `let*` and `let`, identifying the type of binding that is used and explaining the output in each case. *(6)*

```
(define x 100)
(let ( (x 2)
       ( y (+ x x))
      )
  (+ x y))
```

```
(define x 100)
(let* ( (x 2)
        ( y (+ x x))
       )
  (+ x y))
```

(iii) The following VB.NET procedure inserts a value into an integer array, where the procedure is passed the array (`A`), the upper bound of the array (`ub`), the size of the array (total space allocated to the array) and the item to insert (`item`):

```
Private Sub Insert(ByRef A() As Integer, ByRef ub As
Integer, ByVal size As Integer, ByVal item As Integer)

        If ub = size-1 Then
            MsgBox("Error: Not enough Space")
        Else
            ub = ub + 1
            A(ub) = item
        End If
 End Sub
```

Using some sample data explain the difference between passing data to the procedure by value (`ByVal`) and passing by reference (`ByRef`). What is the effect of using the following procedure definition: *(10)*

```
Private Sub Insert(ByVal A() As Integer, ByVal ub As
Integer, ByVal size As Integer, ByVal item As Integer)
```

Q.5. (i) Describe what is meant by *each* of the following with respect to program translation:

(a) Just In Time Compilation. *(2)*
(b) Lexical Analysis. *(2)*
(c) Error Recovery. *(2)*
(d) Semantic Analysis. *(2)*
(e) Peephole Optimisation. *(2)*

(ii) Explain what is meant by a Finite State Automaton (FSA). *(2)*
Illustrate, with the aid of a diagram, an FSA which combines automata to recognise the following lexical tokens: *(8)*

- identifiers
- `if` keyword
- positive real numbers

(iii) The following specifies a simple grammar which should describe a restricted set of algebraic expressions:

Grammar = {N, T, S, P}
N = {<goal>, <expr>, <term>, <factor>}
T = {`2, 4, 6, +, ×` }
S = <goal>
P =
<goal> ::= <expr>
<expr> ::= <term> | <term> × <expr>
<term> ::= <factor> | <factor> + <term>
<factor> ::= 2 | 4 | 6

(a) Outline the general steps in Recursive Descent Top Down Parsing to recognise (parse) expressions. *(2)*

(b) By parsing the sample expression `2+4*6`, obtain a parse tree and abstract syntax tree and discuss the problem(s) with the productions P. *(4)*

(c) Rewrite the productions to correct the error(s) in the productions. *(4)*