NUI Galway
OÉ Gaillimh

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)

PTO

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)