NUI Galway
OÉ Gaillimh

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)