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)