GX 1610

Ollscoil na hÉireann, Gaillimh
National University of Ireland, Galway

Semester II Examinations, 2002/2003

Exam Code(s)	2IF1, 2BN1, 2BP1, 2PT1
Exam(s)	2nd BSc in Information Technology, 2nd BE (Electronic), 2nd BE (Electronic and Computing), 2nd BSc in Physics and Astronomy
Module Code(s)	CT229
Module(s)	Programming II
Paper No.	
Repeat Paper	Special Paper
External Examiner(s)	Prof. P. Nixon
Internal Examiner(s)	Prof. G. Lyons
	Dr. M. Madden
	Mr. C. O'Riordan
Instructions:	Answer 2 questions from each section. Use a separate answer book for each section. All questions carry equal marks.
Duration	3 hours
No. of Answer books	2
Requirements:	
Handout	
MCQ	
Statistical Tables	
Graph Paper	
Log Graph Paper	
Other Material	
No. of Pages	3
Department(s)	Information Technology

Section A

1. (a) Knocknacarra Dairies has hired your company to set up a new stock control system. Your responsibility is to develop software to assist with order management.

   Write a C struct called **OrderNode** for customer order details. The struct must have member variables for storing the customer name and account number, the number of litres of milk ordered and the number of tubs of cream ordered. (4)

   (b) Write a simple C struct called **OrderList**, containing a linked list for storing orders after they are received and before they are delivered. (Note: In this whole question, you may assume that a standard **LinkedList** struct and standard associated functions are defined.) (3)

   (c) It is company policy that orders are dealt with on a "first come first served" basis. Should the **OrderList** be treated as a stack or a queue? Explain your answer. (3)

   (d) Write C functions with the following prototypes:

```
void AddNewOrder(OrderList* all, OrderNode* order);
void RemoveDeliveredOrder(OrderList* all, OrderNode* order);
```

   In each case, the first parameter is a list of all orders and the second is an order to be added/removed. (10)

   (e) Write a C function to calculate the total cost of all orders in the list. You can assume that the costs for milk and cream are already defined in global variables called `MILK_PRICE` and `CREAM_PRICE`. (5)

2. (a) In the context of hash tables, explain what hashing functions and collision processing are. Describe some commonly-used techniques for hashing and collision processing. (7)

   (b) Discuss how functions and comments can both contribute to well-written code. (5)

   (c) Write a C program to make a character-by-character *backwards copy* of a file (i.e. if the original file starts with "ABCD ..." the copy will *end* with " ... DCBA"). The name of the original file and the copy are to be supplied on the command line; if not, the program must display a usage message and exit. The program must include error handling when opening the files. The code to actually perform the backwards copy must be contained in a separate function that takes two parameters, which are file pointers for the original file and the copy. (13)

3. Discuss in detail **all** of the following four pairs of topics, using snippets of code and/or diagrams for illustration as appropriate:

   (1) Structured programming and abstraction

   (2) Static memory allocation and dynamic memory allocation

   (3) Pointers and pointer arithmetic

   (4) The keywords `typedef` and `enum` (25)

SECTION B

4. (a) Discuss and compare approaches for representing the graph data structure. Provide code fragments to illustrate how you would implement a representation in C. (8)

(b) Outline an algorithm to find the shortest path between two nodes on a graph. Illustrate the workings of the algorithm with an example. Discuss how you might implement this algorithm in C. (9)

(c) Describe Huffman compression. Outline the algorithm and data structures used in Huffman compression. Illustrate the technique on the following string:

this is mississippi

(8)

5. (a) Describe the structure of a binary search tree. Derive an O-notation expression for the efficiency of searching a balanced binary search tree. (6)

(b) Provide C code functions to:

i) search a binary search tree for a given value.
ii) insert a given value into a binary search tree.

(10)

(c) Discuss the need to maintain balanced search trees. Outline any approach for maintaining a balanced search tree. Illustrate the approach with the insertion of the following data:

19, 15, 23, 21, 31, 35, 36, 67, 61, 7

(9)

6. (a) Describe the *quicksort* and *mergesort* sorting algorithms. Compare the performance of these sorting algorithms. Outline, with the use of C code fragments, how you would implement *either* algorithm. (9)

(b) Given a file of records containing details regarding employees, outline an efficient algorithm to sort the records based on the PPS number of employees. You may assume PPS numbers are unique in the file. Furthermore, each PPS number contains 7 digits and then a character (for example, 1234567A, 8628593W). (8)

(c) Provide pseudo-code for an *in-order* traversal of a tree. Illustrate its operation with an example. Show how a Tree-Sort algorithm operates. Discuss the efficiency of this algorithm. (8)