Ollscoil na hÉireann, Gaillimh **GX 1509**

National University of Ireland, Galway

Semester II Examinations, 2003/2004

Exam Code(s)	2IF1, 2BN1, 2BP1, 2PT1
Exam(s)	Second University Examination in Information Technology Second University Examination in Electronic Engineering Second University Examination in Electronic & Computer Engineering Second University Examination in Physics & Astronomy
Module Code(s)	CT229
Module(s)	Programming II
Paper No.	1
Repeat Paper	Special Paper
External Examiner(s)	Prof. P. Nixon
Internal Examiner(s)	Prof. G. Lyons, Dr M. Madden, Ms J. Griffith
Instructions:	Answer TWO questions from EACH section. Use a separate answer book for each section. All questions carry equal marks.
Duration	3 hrs
No. of Answer books	2
Requirements:	
Handout	
MCQ	
Statistical Tables	
Graph Paper	
Log Graph Paper	
Other Material	
No. of Pages	4
Department(s)	Information Technology

Section A

1. (a) Tammy's TAM Ratings has hired you to develop software for analysing TV viewing figures.

Write a C struct called **Rating** for storing the details of a TV programme's viewing figures. The struct must have member variables for storing the programme name (e.g. "The Big Bow Wow"), channel (e.g. "RTE1"), the programme duration in hours (e.g. 0.5), the day of the week and the number of viewers. The day of the week can be represented as a number from 1 = Monday to 7 = Sunday. (6)

(b) Ratings are stored in an array. Write a function called **AllocRatings** with a parameter specifying the maximum number of **Rating** items to store in the array. The function must return a pointer to dynamically-allocated memory for the array. (3)

(c) The Rating data must be read from a binary file. Write a C function with the following prototype:

```
void ReadRatings(Rating* all, int count, char* filename);
```

The first parameter is a pointer to an array that will hold **Rating** items, the second is a count of the number of items in the array, and the third is a file name. The function must display a warning if the file cannot be opened or the correct number of ratings cannot be read from it. (5)

(d) The company is interested in finding out which programme is the most popular on a given channel. Write a function with the following prototype:

```
Rating* GetTopRating(Rating* all, int numRatings, char* channel);
```

This searches the array **all**, which has **numRatings** items in it, to find the item with the highest rating for the specified **channel**, and returns a pointer to that item. (5)

(e) Viewing levels vary according to the day of the week. Write a function called **PrintDailyViewerHours** that has two inputs, the array of all ratings and the number of items in the array. It must print on screen the total number of viewer-hours for each day of the week (e.g. "Day 1: total viewer hours = 567890.12"). The viewer-hours for a single programme is its duration multiplied by its number of viewers, so this must be added up for all programmes on all channels for a given day. (6)

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

(1) Hash tables, hashing functions and collision processing

(2) Characteristics of structured programming and its advantages

(3) Complexity of sorting algorithms

(4) The importance of testing and organisation of functions to good program development (25)

3. (a) Explain, using snippets of code or diagrams, how to insert and delete nodes in a **doubly** linked list. (5)

(b) A Stack ADT may be implemented using a linked list or an array. Explain, using diagrams and/or code, both of these ways of implementing a stack. Which one would you recommend and why? (8)

(c) Write a C program to process a file containing a list of people's names and ages. Each line of the input file is of the format:

```
Thomas    57
```

The program must read each line from the input file, and write the people's names only (not the ages) to separate lines of an output file. It must also display a message saying what the average age is.

The names of the input file and the output file should be supplied on the command line. If not, a usage message must be displayed. Appropriate error messages must be displayed when working with the files. (12)

Section B

4. (a) In terms of efficiency, compare the two divide and conquer sorting approaches, *Quick Sort* and *Merge Sort*. For the Merge Sort algorithm, provide C code fragments for the algorithm. Use sample data to illustrate the behaviour of the algorithm. (8)

(b) With the aid of an example, describe how the *Count Sort* algorithm works. Outline, with the aid of C code fragments, how you would implement *Count Sort*. Comment on the efficiency of the algorithm and any advantages and disadvantages of the algorithm. (9)

(c) Provide C code for the in-order traversal of a Binary Tree. Show how a *Tree-Sort* algorithm operates. Discuss the efficiency of the *Tree-Sort* algorithm. (8)

5. (a) Describe the structure of a Binary Search Tree. Provide a C code function to search a Binary Search Tree for a given value. Describe, with the aid of an example, the average number of comparisons required to search for an item in a Binary Search Tree. (8)

(b) Provide a C code function to insert a given value in a Binary Search Tree. Outline, using pseudo-code, an approach to implement deletion of values from a Binary Search Tree. Comment on the efficiency of Binary Search Tree insertions and deletions. (8)

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

14, 10, 17, 18, 24, 25, 15, 16 (9)

6. (a) Discuss and compare approaches for representing the graph data structure. Provide code fragments to illustrate how you would implement these representations 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:

abracadabra (8)