A27217 *No calculator permitted in this examination*

UNIVERSITY OF BIRMINGHAM

School of Computer Science

First Year – BSc Artificial Intelligence and Computer Science
First Year – BSc Computer Science
First Year – MSci Computer Science
First Year – MEng Computer Science/Software Workshop
First Year - BSc Mathematics and Computer Science
First Year - MSci Mathematics and Computer Science
First Year – Computer Science with Study Abroad
First Year – BSc Computer Science with Business Management
First Year – BSc Mathematics and Computer Science with Industrial Year
First Year – MSci Mathematics and Computer Science with Industrial Year
First Year – BSc Computer Science with Industrial Year
Frist Year – MEng Computer Science/Software Engineering with Industrial Year
First Year – BSc Artificial Intelligence and Computer Science with Industrial Year
First Year – BSc Computer Science with Business Management with Industrial Year
First Year - MSci Computer Science with Industrial Year

06 22754

Foundations of Computer Science

Summer Examinations 2015

Time allowed: 3 hours

[Answer ALL Questions]

No calculator

Part One

[Use a Separate Answer Book for THIS Part]

[Answer ALL Questions]

1. Consider *cat-list,* the following alternative implementation of a list data type:

```
type 'a catlist =
  | Empty
  | Elem of 'a
  | Cat of ('a catlist * 'a catlist)
```

(a) Implement a *length* function that returns the number of elements in a *cat-list*. [3%]

(b) Give a 3-element *cat-list* and show the step-by-step execution of the function *length* applied to it. [3%]

(c) Implement a function called *nth* which returns the n-th element of a *cat-list* (if it exists), and fails otherwise. (Do not worry about the efficiency of this function.) [5%]

(d) Give one example of a standard list operation which is more efficient on *cat-lists* than on standard lists. Explain why. [2%]

(e) Give one example of a standard list operation which is less efficient on *cat-lists* than on standard lists. Explain why. [2%]

2. (a) Define a data-type for boolean expressions that can express *conjunction*, *disjunction*, *negation*, *constant true* and *constant false*. [3%]

(b) What is the representation of the expression “(true and false) or (not true)” in this data-type? [3%]

(c) Write a function that will *evaluate* a boolean expression expressed in the data-type above and returns an element of the built-in Ocaml bool type as a result. [3%]

(d) Extend the data-type with with *boolean variables*. [2%]

(e) Write a function that checks whether an expression contains any variables or not. [4%]

(f) Write a function that takes an expression and replaces all subexpressions that contain no variables with their value, which is a constant. [5%]

3. (a) Implement *merge-sort* in OCaml. [5%]

   (b) Implement *quick-sort* in OCaml. [5%]

   (c) How does the efficiency of *merge-sort* compare to that of *quick-sort*? [5%]

Part TWO

[Use a Separate Answer Book for THIS Part]

[Answer ALL Questions]

4. (a) Give a formal inductive/recursive definition of a *Binary Tree*, and then specify what additional conditions must be satisfied for a given Binary Tree to be (i) *Complete*, and (ii) a *Binary Heap Tree*. [3%]

   (b) Explain in words what the standard `bubbleDown(i,a,n)` procedure is used for in the context of Binary Heap Trees, and how exactly it works. [3%]

   (c) Write an efficient pseudocode procedure `heapify(a,n)` using `bubbleDown(i,a,n)` to create a Binary Heap Tree from any given array `a` of size `n`. [3%]

   (d) Explain whether Binary Heap Trees have better or worse time complexity than Binary Search Trees as a method for sorting an array of `n` items. How does that change if n is large, but only the `m` << `n` largest items need to be returned sorted? [4%]

5. (a) Describe in words the general idea of *divide and conquer* approaches to sorting, and how the *Quicksort* algorithm applies that idea. [4%]

   (b) What does it mean to say that a sorting algorithm is *stable*? Discuss the stability of the *Quicksort* algorithm and its relation to the computational costs of the algorithm. [4%]

   (c) Two-phase Radix sort is said to be a *non-comparison-based* sorting algorithm. Explain how it can put items in order without comparing them, and what properties the data must satisfy for it to work well. [4%]

6. (a) Give standard definitions of the terms *hash table*, *hash function* and *hash collision*. [3%]

   (b) Outline why hash tables are useful and what their main disadvantage is. [2%]

   (c) Outline the three principal approaches to dealing with hash collisions, together with the main advantages and disadvantages of each. [5%]

   (d) Suppose an online retailer wanted to store details of its current products using a hash table based on their nine digit identification numbers. At any given time it has about 10,000 products available, and changes its range of products at the rate of about 700 per month. Suggest suitable hash table approaches for them. Explain the reasons for your suggestions and state any assumptions they rely on. [3%]

7. (a) Explain the three general approaches to implementing weighted graphs generally known as *array-based*, *pointer-based* and *mixed*. Outline the main advantages and disadvantages of each approach. [4%]

   (b) Describe what a *minimal spanning tree* of a weighted graph is, and give a practical example of when one may be useful. [3%]

   (c) Describe a greedy edge-based algorithm for determining a minimal spanning tree of a weighted graph, outline an efficient approach for implementing it, and explain what its time complexity is. [5%]