A27217 *No calculator permitted in this examination*

UNIVERSITY OF BIRMINGHAM

School of Computer Science

First Year – BSc Artificial Intelligence and Computer Science
First Year – UG Affiliated German
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 – BA Political Economy
First Year – BSc Computer Science with Business Management
First Year – BSc 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
First Year – BA/BSc Liberal Arts and Sciences

06 22754

Foundations of Computer Science

Summer Examinations 2014

Time allowed: 3 hours

[Answer ALL Questions]

[Answer EACH Section is a different Answer Book]

No calculator

Section A

[Answer THIS Section in a different Answer Book]

[Answer ALL Questions]

1. (a) Write a function that takes a list of pairs and produces a list of the first elements of the pairs.

```
proj1 : ('a * 'b) list -> 'a list
```

Example: `proj1 [(1, "a"); (2, "b"); (3, "c")] = [1; 2; 3]` [5%]

(b) Show the step-by-step symbolic execution of this expression: `proj1 [(1, "a"); (2, "b")]` [5%]

(c) Using structural induction show that for any list `abs` the list produced by `proj1` has the same length as `abs`. [10%]

2. In this question we will use the canonical (unary) data-type of natural numbers, represented in OCaml as

```
type nat = Zero | Suc of nat;;
```

(a) Give the OCaml implementation for the function `add`, which computes the sum of two natural numbers (`nat`). [5%]

(b) Give the OCaml implementation for the function `sub`, which computes the difference of two natural numbers (`nat`). For simplicity we assume that subtracting a larger number from a smaller one returns zero. [5%]

(c) Show that for any natural numbers `m` and `n`, we have `sub (add m n) m = n.`

[10%]

3. (a) Explain in a few words what is a search tree and why it is useful. Give an OCaml implementation of a search tree datatype. [2%]

   (b) Write an OCaml function which computes the histogram (i.e. how many times each element occurs) of a list. The argument is a list and the result a list of pairs, in no particular order. For maximum points adapt a search tree to improve the performance of your function. [8%]

   Example:

```
histo [‘a’;’b’;’c’;’d’;’a’;’b’;’c’;’a’] =
       [(‘a’,3);(‘c’,2);(‘d’,1);(‘b’,2)]
```

Section B

4. (a) List and describe a minimal set of primitive constructors, selectors and conditions that are sufficient to build and manipulate *Binary Trees*. [3%]

   (b) Use your primitive operators to write a recursive pseudocode algorithm `differentBT(t1,t2)` that returns true if two binary trees `t1` and `t2` are different, and false if they are identical. What is the time complexity of your algorithm? [4%]

   (c) Specify the additional properties that a Binary Tree requires to make it a *Binary Search Tree*. [1%]

   (d) Explain what is meant by a *tree rotation* and why it might be useful to perform one. [2%]

   (e) Draw the binary search tree that results from inserting the items [20, 30, 35, 10, 40, 25] in that order into an initially empty tree. Show how a tree rotation can usefully be applied to that tree. [2%]

5. (a) Explain what is meant by *Priority Queue* and *Heap Tree*, and how one may be used to implement the other. [3%]

   (b) Explain what the processes *Bubbling Up* and *Bubbling Down* do in the context of heap trees, and what they are used for. [3%]

   (c) Suppose you already have a procedure `bubbleDown(i,a,n)` that bubbles down item `i` of a heap tree array `a` of size `n`, and a procedure `swap(i,j,a)` that simply swaps items `i` and `j` of array `a`. Use those to write a procedure `heapSort(a,n)` that starts with an array `a` and finishes with that array sorted. [3%]

   (d) Suppose you ran a bank and regularly wanted to sort your nine million customers according to how much money they kept in your bank, but you were only interested in seeing the results for your top ten customers. Explain which sorting algorithm you would use and why. [3%]

6. (a) Explain what is meant by the terms *hash table*, *hash function* and *hash collision*. [3%]

   (b) Comment on how the computational costs associated with good hash tables vary with the number of entries, and what advantages and disadvantages that gives them. [3%]

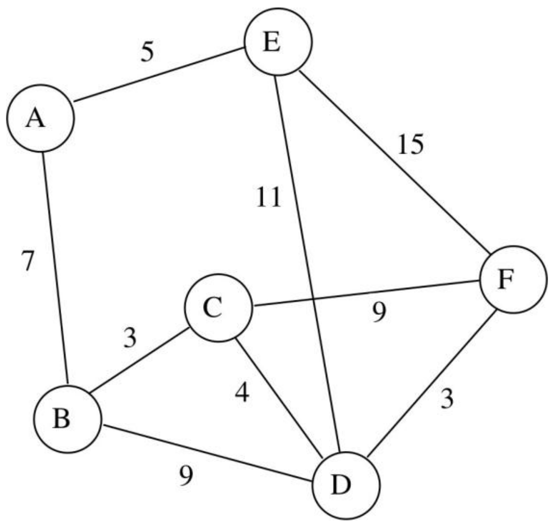
   (c) Suppose strings of four digits are to be stored in a hash table represented as an array of size 11. The primary hash function is the sum of the third and fourth digits modulo 11. Show the result of starting with an initially empty hash table and inserting the following strings into it using linear probing: "1041", "1292", "1130", "1323", "1214", "1112" and "1010". What is the load factor of the resulting table, and how many collisions occurred? [4%]

   (d) Show what the resulting hash table would look like if direct chaining had been used rather than linear probing. [3%]

   (e) Comment on the relative efficiencies of linear probing and direct chaining. [2%]

7. (a) Explain what is the difference between *breadth first traversal* and *depth first traversal* of a connected graph. Suggest what kind of data structure would be best to implement each of them. [3%]

   (b) Outline the general idea of *Dijkstra's algorithm* for finding the shortest path between two nodes in a weighted graph. Make clear what information needs to be maintained at each stage when the algorithm is used. Apply the algorithm to find the shortest path from A to F in the following weighted graph, showing the current information at each stage. [8%]



5
E
A
15
11
7
F
C
9
3
4
3
B
9
D