Sample Final Examination

Michael P. Fourman

February 2, 2010

University of Western Australia
CS201 Final Examination
will be held on Friday 3rd June, 1994

Information

This document contains three sample questions. These supplement the sample short questions issued earlier in manuscript. While the sample questions indicate the general tenor of the questions you should anticipate on the final paper, the omission of a topic from this sample should not be taken as an indication that it will not appear; nor should you infer that such topics will appear. Indeed, *any* attempt to use this sample to predict which topics will be covered in the final examination may be misguided.

In the examination you will be expected to attempt three short questions (5 marks each), and two (out of three) longer questions (10 marks each).

Examination The examination will take place on Friday 3rd June. You should go to the appropriate venue (lists have been posted outside the laboratory), and be seated by 2.00pm. Ten minutes reading time will be allowed, starting at 2.05, and the examination will finish at 3.15pm.

Project Although most project demonstrations have been straightforward, I anticipate that I will want to review a small number of cases to ensure that appropriate credit is given for work done. A list of names of those affected, and proposed times for the review, will be available at the end of the examination.

B1 *(10 marks)*

(a) *(2 marks)*

Give an ML signature `QueueSig` suitable for the abstract data type *priority queue.* Your signature should include a type `Item`, as the type of the items in the queue.

(b) *(3 marks)*

For each of the operations in your signature, give the complexity of an implementation based on

i. ordered lists

ii. unordered lists

iii. heaps

(make a table).

(c) *(5 marks)*

Write a functor with the header

```
functor SORT( structure PQ: QueueSig ) :
sig
    type Item
    val sort : Item list -> Item list
end
```

that uses a priority queue to implement a sorting function `sort`. The result of applying the function `sort` should be a permutation of the items in the argument list, arranged in order of increasing priority.

B2 *(10 marks)*

(a) *(2 marks)*

What is an *efficient* hash function?

(b) *(8 marks)*

An implementation of sets of integers as ordered lists is "improved" by hashing with a hash table of size h and the hash function

```
fun hash n = n mod h
```

(each entry in the hash table is itself an ordered list).

For each of the following set operations, give the complexity (in terms of the size, N, of the set) for the original implementation, and say what speedup (or slowdown) you would expect to obtain from the introduction of hashing. (Assume that the hash function is efficient for the data encountered, and that the cost of computing the hash function may be ignored.)

i. `empty`

ii. `isEmpty`

iii. `member`

iv. `insert`

v. `delete`

vi. `union`

vii. `intersect`

B3 *(10 marks)*

Prim's algorithm for finding a minimal-cost spanning-tree uses two auxiliary datastructures: a priority queue of edges, and a set of vertices.

(a) *(3 marks)*
Describe the operations on these datastructures that are used by Prim's algorithm. (Give the type of each operation, and briefly describe its effect.)

(b) Consider a graph, $G = (V, E)$, whose vertices are integers.

i. *(2 marks)*
What is the complexity of the priority queue operations if the queue is implemented as a heap?

ii. *(3 marks)*
Give the complexities of the set operations for each of the following implementations of the vertex set:

A. ordered list

B. balanced search-tree

C. array of booleans

iii. *(2 marks)*
For each of these implementations of the set, assuming the priority queue is implemented as a heap, derive an expression for the complexity of Prim's algorithm in terms of the sizes of the sets (V, E).

The End