UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

MASTER OF SCIENCE IN COMPUTER SCIENCE

APPLICATIVE PROGRAMMING AND SPECIFICATION

Tuesday 24 April 2001

14:00 to 15:30

Examiners:

I. Watson (External)

R.N. Ibbett (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. Consider binary trees defined as follows:

```
datatype 'a tree = empty | tip of 'a | node of 'a tree * 'a tree
```

The *deepest* tips are the ones that are furthest from the root of the tree. So in

```
node(node(node(tip 2, empty),
          tip 1),
     node(empty,
          node(tip 3, tip 2)))
```

the deepest tips are the ones underlined.

(a) We want a function `deepest : 'a tree -> 'a list` that returns a list containing the labels of the deepest tips in a tree, in the order that they appear. In the example above, the result would be `[2,3,2]`.

One algorithm for `deepest` works by computing the depth of the deepest tip and then using this result in a function that returns the list of all tip labels at a given depth. Implement this. [*8 marks*]

(b) Another algorithm for `deepest` works by traversing the tree, deciding at each node which recursive applications of `deepest` are needed in the result on the basis of the depth of that subtree. Implement this. [*8 marks*]

(c) Implement `shallowest : 'a tree -> 'a list`, which returns a list containing the labels of the *shallowest* tips in a tree. In the example above, the result would be `[1]`. (Note: `shallowest` t should be `[]` only if there are no tips in t.) [*9 marks*]

2. A *priority queue* (pqueue) is a data structure that allows elements to be added in arbitrary order, but the smallest element is always removed first.

   (a) Define a functor in Standard ML that provides priority queues over an arbitrary type of items with respect to an ordering on items. The following functions should be included:

`empty : pqueue`	The empty pqueue.
`isempty : pqueue -> bool`	Test for the empty pqueue.
`insert : item * pqueue -> pqueue`	Add an item to a pqueue.
`smallest : pqueue -> item`	Smallest item in a pqueue.
`rest : pqueue -> pqueue`	Remove the smallest item from a pqueue.

   Both `largest(empty)` and `rest(empty)` should raise an exception. [*13 marks*]

   (b) Using the functor defined in (a), write a functor that provides a function to sort a list of items into ascending order by inserting all the items in the list into a pqueue and then removing them in order. [*12 marks*]

3. One strategy for fixing Y2K problems is to use *windowing*. The idea is to continue using two digits to represent years, but re-interpret the meaning of the two digits as follows. Each window has a given start year or *pivot*, and all numbers less than the pivot are interpreted as years in the 21st century, while numbers greater than or equal to the pivot are interpreted as belonging to the 20th century. For example, if the pivot is 70 then 41 means 2041 while 75 means 1975. (The usual 20th century way of using two digits to represent years is equivalent to using a pivot of 0.)

   (a) Use Extended ML to specify a functor `Window` that implements a window containing a type `year` and a function `precedes : year * year -> bool`, with respect to a pivot that is provided as a functor parameter. You need not hide the fact that the pivot and years are represented as integers between 0 and 99. [*8 marks*]

   (b) Use Extended ML to specify a functor that takes two windows as parameters and implements a function `precedes` that compares years from the first window with years from the second. Do not assume that the two windows have the same pivots. [*8 marks*]

   (c) The following axioms specify a function that checks to see if a list of years is ordered with respect to `precedes`:

```
ordered []
forall a => ordered [a]
forall (a,b,l) =>
   ordered(a::b::l) iff precedes(a,b) andalso ordered(b::l)
```

   Use this to prove

```
forall (a,l,b) => ordered([a]@l@[b]) implies precedes(a,b)
```

   by structural induction on `l`. You may assume basic properties of `@` and `proceeds`. (Hint: In the "step" case, you will probably need a case analysis on `l`, but you shouldn't need a nested induction.) [*9 marks*]