

# Introduction to programming

End Semester Examination

3 December, 2004

Duration: 3 hours

Marks: 40

**Note:** Answer all questions. Whenever a question asks for a Haskell function, write down the most general type of the function before the actual definition, including any dependency on type class

1. Define a Haskell function `shuffle` that takes as an input a list of integers and rearranges the list so that odd and even integers alternate. If the number of odd and even integers in the list are not equal, the extra odd/even integers should come at the end of the output list. The order of odd/even numbers in the input list should be preserved. (4 marks)
2.
  - Define a Haskell function `nondecreasing` that takes a list `xs` and returns `True` if the elements in the list are in non-decreasing order.
  - Generalise the previous function to a function `goodlist` that takes a predicate  $p :: a \rightarrow a \rightarrow \text{Bool}$  and a list `xs` over the type `a` and returns `True` if each adjacent pair of elements in `xs` satisfy `p`.
  - Use `goodlist` to define a function `alternating` that takes a list of integers `xs` and returns `True` if the elements of the list are alternately odd and even.

(6 marks)

3. Consider the following Haskell definition:

$$table = [[m^n | n < -[2..]] | m < -[1..]]$$

- Describe the contents of the table
- What would Hugs print out if you asked it to display all the values in the table.
- Write a Haskell expression to extract the finite list `[1, 16, 81, 256, 625]` from `table`.

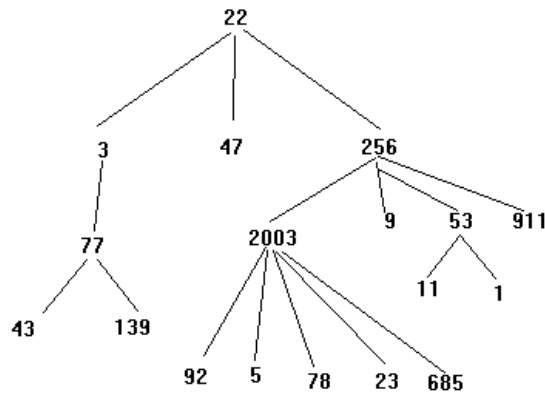
(6 marks)

4. Write an efficient Haskell implementation of the abstract datatype `Array` that supports the following operations:

- *makeArray*: Constructs an array from a list of values
- *lookup*: Given an integer  $n$  and an array  $a$ , returns the value at position  $n$  in the array. Assume the array indices start from 1.
- *update*: Given an integer  $n$ , an array  $a$  and a value  $v$ , updates the value at position  $n$  in the array to  $v$ .

(8 marks)

5. A *rose tree* is a tree with multiway branching. Each node of a rosetree stores a value. The number of children of each node is not fixed and can be as large as one wants. Here is an example:



- Write a Haskell data declaration for the abstract datatype *Rosetree*  $a$
- Write a Haskell function:

$$printlevel :: Int \rightarrow (Rosetree\ a) \rightarrow [a]$$

such that *printlevel*  $n\ t$  returns the list of values from left to right at level  $n$  of the tree. For instance, if  $t$  is the tree above, then *printlevel*  $2\ t$  should return  $[77, 2003, 9, 53, 911]$ . The root of the tree is at level 0.

(8 marks)

6. Each semester, the CMI office needs to compile the list of students who are registered for at least one course in that semester. Each student is assigned a unique roll number and the information about his course registration is stored in terms of roll numbers.

Write a Haskell function to extract the information required by the CMI office.

The input to your function consists of 2 lists.

- The first line identifies the roll number of each student. It is a list of pairs (Name, Rollnumber) where Name is a string and Rollnumber is an integer.
- The second list has information about all courses running in the current semester. It is a list of pairs (CourseName, ListOfStudents) where CourseName is a string and ListOfStudents is a list of integers. The output of your function should be a list consisting of the names of the students who are currently registered for atleast one course at CMI. This list need not be in any specific order but it should not have any duplicate entries. You may assume that no 2 students have the same name.

(8 marks)