National Undergraduate Programme in Mathematical Sciences
Introduction to Programming
National Graduate Programme in Computer Science
Programming Laboratory
Mid-Semester Examination, I Semester, 2008–2009

Date : 22 September, 2008 — Marks : 30
Duration : Three hours — Weightage : 20%

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

1. Define the following functions in Haskell.

   (a) `applyEach`, that takes a list of functions $[f_1, f_2, \ldots, f_n]$ and applies each of them to a given value v, returning the list of values $[f_1\ v, f_2\ v, \ldots, f_n\ v]$. For instance (assuming that `factorial` has already been defined):

   `applyEach [(+3), factorial] 5` $\Longrightarrow$ `[8,120]`.

   (b) `applyAll`, that takes a list of functions $[f_1, f_2, \ldots, f_n]$ and a value v and returns the value $f_1\ (f_2\ (\cdots (f_n\ v) \cdots))$. For instance:

   `applyAll [(+3), factorial] 5` $\Longrightarrow$ `123`.

   (4 marks)

2. Define a Haskell function `findunique` that reads as input two lists of the same type and returns all values that occur in one list or the other, but not both. For instance, `findunique [4,3,5] [3,1,6,4]` should return `[5,1,6]` because `3` and `4` occur in both lists. You may assume that no value repeats in either input list. The order in which the elements appear in the output is not important.

   Your function should work in time $O(n \log n)$ for a suitable class of inputs, where n is the sum of the lengths of the input lists. Be sure to specify the class of input types for which your function is defined. *(3 marks)*

3. Consider the following function that computes x^n.

```
power :: Float -> Int -> Float
power x 0 = 1.0
power x n = x * (power x (n-1))
```

   For inputs x and n, this function calls itself $O(n)$ times. To reduce the number of recursive calls, observe that x^{2k} can be rewritten as $(x \cdot x)^k$ and x^{2k+1} can be rewritten as $x \cdot (x \cdot x)^k$.

   (a) Write an improved function `fastpower` to compute x^n using these observations.

   (b) How many times does `fastpower` call itself, in terms of x and n? *(4 marks)*

4. In set theory, one way of representing integers is as follows.

   - 0 is denoted by the emptyset, $\emptyset$.
   - $n+1$ is denoted by the set $\{\bar{n}, \{\bar{n}\}\}$, where $\bar{n}$ inductively denotes the representation of n.

Thus, $\bar{1}$, the representation of 1, is $\{\bar{0}, \{\bar{0}\}\}$, or $\{\emptyset, \{\emptyset\}\}$, since $\bar{0} = \emptyset$. Likewise, $\bar{2} = \{\bar{1}, \{\bar{1}\}\}$ $= \{ \{\emptyset, \{\emptyset\}\}, \{\{\emptyset, \{\emptyset\}\}\} \}$, and so on.

We mimic this in Haskell as follows: denote 0 by the emptylist `[]`, 1 by the list `[0,[0]]` = `[[],[[]]]`, 2 by the list `[1,[1]]` = `[[[],[[]]],[[[],[[]]]]]`, etc.

Is it possible to define a function `encode` in Haskell that takes as input an integer and returns its encoding using this notation? Explain your answer. *(3 marks)*

5. One way to generate a sequence of pseudorandom numbers is to start with an integer seed x_0 and repeatedly compute $x_{i+1} = (ax_i + b) \bmod c$, where a, b, c are fixed positive integers, with $0 \leq x_0 < c$.

   For instance, for $a = 3$, $b = 2$, $c = 7$, if we start with $x_0 = 5$, we get $x_1 = (3 \cdot 5 + 2) \bmod 7 = 3$, $x_2 = (3 \cdot 3 + 2) \bmod 7 = 4$, $x_3 = (3 \cdot 4 + 2) \bmod 7 = 0$, $x_4 = (3 \cdot 0 + 2) \bmod 7 = 2$, $x_5 = (3 \cdot 2 + 2) \bmod 7 = 1$, $x_6 = (3 \cdot 1 + 2) \bmod 7 = 5$, after which the sequence repeats.

   If, instead, we had taken $a = 2$, $b = 3$, $c = 7$, starting with $x_0 = 5$ we would have generated $x_1 = (2 \cdot 5 + 3) \bmod 7 = 6$, $x_2 = (2 \cdot 6 + 3) \bmod 7 = 1$, $x_3 = (2 \cdot 1 + 3) \bmod 7 = 5$, after which the sequence repeats.

   Write a function `pseudorandom` that takes as inputs the values a, b, c and x_0 and computes the sequence $[x_0, x_1, \ldots, x_n]$ of pseudorandom numbers generated by these parameters such that x_n is the first number that repeats in the sequence—that is, each $x_i \in [x_0, x_1, \ldots, x_{n-1}]$ is distinct and $x_n = x_j$ for some $0 \leq j < n$.

   (4 marks)

6. Consider the Haskell definition

   ```
   table = [[m*n | n <- [2..]] | m <- [1..]]
   ```

   (a) Describe the contents of `table`.

   (b) What would `hugs/ghci` print out if you asked it to display all the values in `table`?

   (c) Write a Haskell expression to extract the finite list `[5,10,15,20]` from `table`.

   (5 marks)

7. Our aim is to define a list datatype that can hold values of two different types, in any order. For instance, this would allow lists such as `[1,3,1,'a',2,'b']` and `[False,1,1,True]`, which are not otherwise permitted by Haskell's built in lists.

   - Define a new Haskell datatype `Duallist a b` with this property.
   - Write a function `census` that counts the number of elements of each type in a given list of type `Duallist a b` and returns a pair of integers, where the first component of the output is the number of elements of type `a` and the second component is the number of elements of type `b`.
   - Describe how to make this new datatype a member of the type class `Eq` where two lists are deemed to be equal if the sublists of type `a` and type `b` are equal. For instance, `[1,1,True,False]` should be equal to `[True,1,False,1]` because in both lists, the sublist of `Int` is `[1,1]` and the sublist of `Bool` is `[True,False]`. On the other hand `[1,1,True,False]` should not be equal to `[False,1,True,1]` because the sublists `[True,False]` and `[False,True]` are not equal.

   (7 marks)