Dit tentamen is in elektronische vorm beschikbaar gemaakt door de $\mathcal{BC}$ van A–Eskwadraat. A–Eskwadraat kan niet aansprakelijk worden gesteld voor de gevolgen van eventuele fouten in dit tentamen.

Exam Functional Programming

Tuesday, May 23, 2006, 14.00–17.00

EDUC-gamma

The exam consists of four multiple-choice questions (1 point each) and three open questions (2 points each). At the multiple-choice questions, only one choice corresponds to the correct answer. Not answering a multiple-choice question earns you $\frac{1}{4}$ point. Hand in the solution sheets (pages i–iv), with choices circled and open questions answered; fill in your name and student number in the appropriate boxes.

Problems

1. Problem [1 pt]: Which of the following is a correct type for $\mathit{concat} \circ \mathit{concat}$?

   a. $[[a]] \to [[a]] \to [[a]]$

   b. $[[a]] \to [[a]] \to [a]$

   c. $[[[a]]] \to [a]$

   d. $[a] \to [[a]] \to [a]$ □

2. Problem [1 pt]: Which of the following functions counts the number of subsets of a given set of non-negative integers that sum up to a specific value? You may assume that the list argument is indeed a set—i.e., that each value appears at most once as an element of the list—and that all elements are indeed non-negative.

   a.
   $$\begin{array}{ll} \mathit{count}\ [\,]\ 0 & = 1 \\ \mathit{count}\ [\,]\ \_ & = 0 \\ \mathit{count}\ (x:\mathit{xs})\ v & = \mathit{count}\ \mathit{xs}\ (v - x) \end{array}$$

   b.
   $$\begin{array}{lll} \mathit{count}\ [\,]\ 0 & & = 1 \\ \mathit{count}\ \mathit{xs}\ v & \mid v < 0 & = 0 \\ & \mid \mathit{xs} \equiv [\,] & = 0 \\ & \mid \mathit{otherwise} & = \mathit{count}\ (\mathit{tail}\ \mathit{xs})\ (v - \mathit{head}\ \mathit{xs}) + \mathit{count}\ (\mathit{tail}\ \mathit{xs})\ v \end{array}$$

   c.
   $$\begin{array}{ll} \mathit{count}\ \_\ 0 & = 1 \\ \mathit{count}\ \mathit{xs}\ v & = \mathbf{if}\ v \leqslant 0\ \mathbf{then}\ 0\ \mathbf{else}\ \mathit{sum}\ [r \mid x \leftarrow \mathit{xs}, r \leftarrow \mathit{count}\ \mathit{xs}\ (v - x)] \end{array}$$

   d. $\mathit{count}\ \mathit{xs}\ v = \mathit{sum} \circ \mathit{map}\ (\mathit{const}\ 1) \circ \mathit{filter}\ (v\ \equiv)\ \$\ \mathit{segs}\ \mathit{xs}$ □

1

3. PROBLEM [1 PT]: Which of the following expressions is equivalent to the list comprehension $[x + y \mid x \leftarrow [1\,..\,10], \mathit{even}\ x, y \leftarrow [1\,..\,10]]$?

   a. $\mathit{map}\ (+) \circ \mathit{filter}\ (\mathit{even} \circ \mathit{fst})\ \$\ [(x, y) \mid x \leftarrow [1\,..\,10], y \leftarrow [1\,..\,10]]$

   b. $\mathit{concat} \circ \mathit{map}\ ((\mathit{flip}\ \mathit{map}\ [1\,..\,10]) \circ (+)) \circ \mathit{filter}\ \mathit{even}\ \$\ [1\,..\,10]$

   c. $\mathit{map}\ (\lambda x \rightarrow \mathit{map}\ (x+)\ [1\,..\,10]) \circ \mathit{concat} \circ \mathit{filter}\ \mathit{even}\ \$\ [1\,..\,10]$

   d. $\mathit{concat}\ (\mathit{zipWith}\ (+)\ [2, 4\,..\,10]\ [1\,..\,10])$

   Note: $\mathit{flip}\ f\ x\ y = f\ y\ x$. $\square$

4. PROBLEM [1 PT]: Which of the following claims holds?

   a. The function *return* is idempotent—i.e., in all contexts, $\mathit{return}\ (\mathit{return}\ x)$ can safely be replaced by $\mathit{return}\ x$;

   b. there exist expressions of type IO (IO Int);

   c. if you define an instance of the class Eq, you have to at least specify the operator $(\equiv)$;

   d. the class Enum has a fixed number of instances. $\square$

5. PROBLEM [2 PTS]: One of the disadvantages of the search trees as discussed in the lectures is that they are a bit wasteful. For instance, a singleton value v is represented by *Node Leaf v Leaf*. A more efficient data type encodes the emptyness of left and right subtrees in a constructor. For example:

```
data Tree a = Leaf
            | LVR (Tree a) a (Tree a)   -- like Node l v r
            | LV (Tree a) a             -- representing Node l v Leaf
            | VR a (Tree a)             -- representing Node Leaf v r
            | V a                       -- representing Node Leaf v Leaf.
```

   Define the functions for insertion and deletion for this type of search trees. Hint: use "smart constructors":

```
node Leaf a Leaf = V a
node Leaf a r    = VR a r
  -- etc.
```

   $\square$

6. PROBLEM [2 PTS]: Consider the data type Prop,

```
data Prop = And     Prop Prop
          | Or      Prop Prop
          | Implies Prop Prop
          | Cnst    Bool
          | Var     String.
```

   (1) Give the type signature and definition of the corresponding fold function, *foldProp*.

2

(2) Use the function *foldProp* to define a function $evalProp :: \mathsf{Prop} \rightarrow \mathsf{Env} \rightarrow \mathsf{Bool}$ which computes the value of a given proposition in an environment of type Env,

$$\textbf{type}\ \mathsf{Env} = \mathsf{String} \rightarrow \mathsf{Bool}.$$

If you had no clue at part (1), then define *evalProp* directly. $\square$

7. PROBLEM [2 PTS]: Prove by induction on lists that $foldr\ f\ e\ (reverse\ xs) = foldl\ (flip\ f)\ e\ xs$. You may use the lemma $foldr\ f\ e\ (as +\!\!+ [b]) = foldr\ f\ (f\ b\ e)\ as$. $\square$