Solutions to the
Exam Functional Programming

Tuesday, May 23, 2006, 14.00–17.00

EDUC-gamma

Note: these solutions are provided "as is" and confer no rights.

1. SOLUTION: (c). ☐
2. SOLUTION: (b). ☐
3. SOLUTION: (b). ☐
4. SOLUTION: (b). ☐
5. SOLUTION: The smart constructor *node* takes a value and two subtrees and applies the appropriate Tree constructor:

```
node               :: Tree a → a → Tree a → Tree a
node Leaf v Leaf   = V v
node Leaf v r      = VR v r
node l v Leaf      = LV l v
node l v r         = LVR l v r
```

The function *maxT* selects the greatest element from a given tree:

```
maxT               :: Tree a → a
maxT Leaf          = ⊥
maxT (LVR l v r)   = maxT r
maxT (LV l v)      = v
maxT (VR v r)      = maxT r
maxT (V v)         = v.
```

The code for insertion and deletion should maintain the search-tree property as well as the invariant that either a tree is empty (i.e., *Leaf*) or it has no empty subtrees. The latter is established by employing the smart constructor *node* in the definition of the deletion function.

```
insert                          :: (Ord a) ⇒ a → Tree a → Tree a
insert Leaf v                   = V v
insert (LVR l v0 r) | v ⩽ v0    = LVR (insert v l) v0 r
                    | otherwise = LVR l v0 (insert v r)
insert (LV l v0)    | v ⩽ v0    = LV (insert v l) v0
```


1

$$
\begin{array}{lll}
 & \mid \mathit{otherwise} & = \mathit{LVR}\ l\ v_0\ (V\ v) \\
\mathit{insert}\ (\mathit{VR}\ v_0\ r) & \mid v \leqslant v_0 & = \mathit{LVR}\ (V\ v)\ v_0\ r \\
 & \mid \mathit{otherwise} & = \mathit{VR}\ v_0\ (\mathit{insert}\ v\ r) \\
\mathit{insert}\ (V\ v_0) & \mid v \leqslant v_0 & = \mathit{LV}\ (V\ v)\ v_0 \\
 & \mid \mathit{otherwise} & = \mathit{VR}\ v_0\ (V\ v)
\end{array}
$$

$$
\begin{array}{lll}
\mathit{delete} & & :: (\mathsf{Ord\ a}) \Rightarrow \mathsf{a} \to \mathsf{Tree\ a} \to \mathsf{Tree\ a} \\
\mathit{delete}\ v\ \mathit{Leaf} & & = \mathit{Leaf} \\
\mathit{delete}\ v\ (\mathit{LVR}\ l\ v_0\ r) & \mid v < v_0 & = \mathit{node}\ (\mathit{delete}\ v\ l)\ v_0\ r \\
 & \mid v \equiv v_0 & = \mathbf{let}\ v_{\max} = \mathit{maxT}\ l \\
 & & \quad \mathbf{in}\ \ \mathit{node}\ (\mathit{delete}\ v_{\max}\ l)\ v_{\max}\ r \\
 & \mid \mathit{otherwise} & = \mathit{node}\ l\ v_0\ (\mathit{delete}\ v\ r) \\
\mathit{delete}\ v\ t@(\mathit{LV}\ l\ v_0) & \mid v < v_0 & = \mathit{node}\ (\mathit{delete}\ v\ l)\ v_0\ \mathit{Leaf} \\
 & \mid v \equiv v_0 & = l \\
 & \mid \mathit{otherwise} & = t \\
\mathit{delete}\ v\ t@(\mathit{VR}\ v_0\ r) & \mid v < v_0 & = t \\
 & \mid v \equiv v_0 & = r \\
 & \mid \mathit{otherwise} & = \mathit{node}\ \mathit{Leaf}\ v_0\ (\mathit{delete}\ v\ r) \\
\mathit{delete}\ v\ t@(V\ v_0) & \mid v \equiv v_0 & = \mathit{Leaf} \\
 & \mid \mathit{otherwise} & = t
\end{array}
$$

□

6. SOLUTION:

(1) $\mathit{foldProp} :: (\mathsf{a} \to \mathsf{a} \to \mathsf{a}) \to (\mathsf{a} \to \mathsf{a} \to \mathsf{a}) \to (\mathsf{a} \to \mathsf{a} \to \mathsf{a}) \to (\mathsf{Bool} \to \mathsf{a})$
$\quad \to (\mathsf{String} \to \mathsf{a}) \to \mathsf{Prop} \to \mathsf{a}$

$$
\begin{array}{l}
\mathit{foldProp}\ f_{\text{and}}\ f_{\text{or}}\ f_{\text{implies}}\ f_{\text{cnst}}\ f_{\text{var}} = \mathit{fold} \\
\quad \mathbf{where} \\
\qquad \begin{array}{ll}
\mathit{fold}\ (\mathit{And}\ p\ q) & = f_{\text{and}}\ (\mathit{fold}\ p)\ (\mathit{fold}\ q) \\
\mathit{fold}\ (\mathit{Or}\ p\ q) & = f_{\text{or}}\ (\mathit{fold}\ p)\ (\mathit{fold}\ q) \\
\mathit{fold}\ (\mathit{Implies}\ p\ q) & = f_{\text{implies}}\ (\mathit{fold}\ p)\ (\mathit{fold}\ q) \\
\mathit{fold}\ (\mathit{Cnst}\ b) & = f_{\text{cnst}}\ b \\
\mathit{fold}\ (\mathit{Var}\ x) & = f_{\text{var}}\ x
\end{array}
\end{array}
$$

(2) $\mathit{evalProp} \quad :: \mathsf{Prop} \to \mathsf{Env} \to \mathsf{Bool}$
$\quad \mathit{evalProp}\ p\ \mathit{env} = \mathit{foldProp}\ (\wedge)\ (\vee)\ ((\vee) \circ \neg)\ \mathit{id}\ \mathit{env}\ p$

□

7. SOLUTION: Proceed by induction on the structure of *xs*.

Case $xs = [\,]$*:*

Proceed by equational reasoning.

$$
\begin{array}{ll}
\quad \mathit{foldr}\ f\ e\ (\mathit{reverse}\ [\,]) & \\
= & \text{(definition of } \mathit{reverse}\text{)} \\
\quad \mathit{foldr}\ f\ e\ [\,] & \\
= & \text{(definition of } \mathit{foldr}\text{)} \\
\quad e & \\
= & \text{(definition of } \mathit{foldl}\text{)} \\
\quad \mathit{foldl}\ (\mathit{flip}\ f)\ e\ [\,] &
\end{array}
$$

Case $xs = x : xs'$*:* $\qquad \mathit{foldr}\ f\ e\ (\mathit{reverse}\ xs') = \mathit{foldl}\ (\mathit{flip}\ f)\ e\ xs'$

Proceed by equational reasoning.

$$
\begin{array}{ll}
\quad \mathit{foldr}\ f\ e\ (\mathit{reverse}\ (x : xs')) & \\
= & \text{(defintion of } \mathit{reverse}\text{)} \\
\quad \mathit{foldr}\ f\ e\ (\mathit{reverse}\ xs' +\!\!+ [x]) & \\
= & \text{(lemma)} \\
\quad \mathit{foldr}\ f\ (f\ x\ e)\ (\mathit{reverse}\ xs') & \\
= & \text{(induction hypothesis)} \\
\quad \mathit{foldl}\ (\mathit{flip}\ f)\ (f\ x\ e)\ xs' & \\
= & \text{(definition of } \mathit{flip}\text{)} \\
\quad \mathit{foldl}\ (\mathit{flip}\ f)\ (\mathit{flip}\ f\ e\ x)\ xs' & \\
= & \text{(definition of } \mathit{foldl}\text{)} \\
\quad \mathit{foldl}\ (\mathit{flip}\ f)\ e\ (x : xs') & \qquad \square
\end{array}
$$