FP 2010-2011, Herkansing
Jan 5, 2011

Do not forget necessary parentheses! For the multiple choice hand in the exam sheet with the chosen answers encircled.

1 Mulitple Choice 1

Which of the items is true for the following definition:

class $Eq\ a$ **where**
$(\equiv), (\not\equiv) :: a \to a \to Bool$
$x \not\equiv y = not\ (x \equiv y)$
$x \equiv y = not\ (x \not\equiv y)$

1. In a class definition it is not allowed to define functions in terms of each other.
2. This is exactly the definition of the class *Eq* from the Haskell report.
3. Because *Eq* is built-in into Haskell it can also be used to compare functions.
4. The function definitions are not allowed here, since they belong to the **instance** declarations and not to the class declaration.

2 Mulitple Choice 2

Using GHCi the Haskell expression: $2 + True$ results in the error message:

```
No instance for (Num Bool)
  arising from use of ‘+’ at <interactive>:1:1
Probable fix: add an instance declaration for (Num Bool)
```

If we follow the hint of the system we have amongst others to:

1. Define a function *fromInteger* that maps *True* to some integer value.
2. Define a function for $(+)$ with type $Integer \to Bool \to Int$.
3. Define a function for *fromInteger* that has the type $Integer \to Bool$.
4. Both b and c

3 Multiple Choice 3

Which is the type of *map foldr*?

1. $(a \to b \to b) \to [a \to b] \to [b]$
2. $[a \to b \to b] \to [b \to [a] \to b]$
3. $[a \to a \to b] \to [[a] \to b]$
4. $[a \to b \to b] \to [[a] \to b \to b]$

4 Multiple Choice 4

The function *intString* converts an *Int*-value to its *String* representation. We have the following variants:

$$\begin{array}{l}
\mathit{intString1} = \mathit{reverse} \quad . \ \mathit{map}\ (\mathit{digitChar} \ . \ (`\mathit{rem}`10)) \quad . \ \mathit{takeWhile}\ (\not\equiv 0) \ . \ \mathit{iterate}\ (/10) \\
\mathit{intString2} = \mathit{map}\ (\mathit{digitChar} \ . \ (`\mathit{rem}`10)) \quad . \ \mathit{reverse} \quad . \ \mathit{takeWhile}\ (\not\equiv 0) \ . \ \mathit{iterate}\ (/10) \\
\mathit{intString3} = \mathit{map}\ \mathit{digitChar} \ . \ \mathit{reverse} \quad . \ \mathit{map}\ (`\mathit{rem}`10) \ . \ \mathit{takeWhile}\ (\not\equiv 0) \ . \ \mathit{iterate}\ (/10)
\end{array}$$

Which of the following statements is true?

1. Only *intString1* is a solution.
2. Only *intString2* is a solution.
3. Only *intString3* is a solution.
4. All definitions are equivalent.

5 Multiple Choice 5

In the Haskell prelude the list constructor [] has been made an instance of the class *Monad*:

$$\begin{array}{l}
\textbf{instance}\ \mathit{Monad}\ [\,]\ \textbf{where} \\
\quad \mathit{ma} \gg\!= \mathit{a2mb} = \mathit{concat}\ (\mathit{map}\ \mathit{a2mb}\ \mathit{ma}) \\
\quad \mathit{return}\ a \qquad = [\,a\,]
\end{array}$$

Which of the following equals $[\,f\ x\ y \mid x \leftarrow \mathit{expr1}, y \leftarrow \mathit{expr2}\,]$?

1. $$\begin{array}{l}
\textbf{do}\ \mathit{return}\ (f\ x\ y) \\
\quad \textbf{where do}\ x \leftarrow \mathit{expr1} \\
\qquad\qquad\quad\ \ y \leftarrow \mathit{expr2}
\end{array}$$

2. $$\begin{array}{l}
\textbf{do}\ x \leftarrow \mathit{expr1} \\
\quad y \leftarrow \mathit{expr2} \\
\quad f\ x\ y
\end{array}$$

3. $$\begin{array}{l}
\textbf{do}\ x \leftarrow \mathit{expr1} \\
\quad y \leftarrow \mathit{expr2} \\
\quad \mathit{return}\ (f\ x\ y)
\end{array}$$

4. $$\begin{array}{l}
\textbf{do}\ y \leftarrow \mathit{expr2} \\
\quad x \leftarrow \mathit{expr1} \\
\quad \mathit{return}\ (f\ x\ y)
\end{array}$$

6 Multiple Choice 6

Which of the following is true?

1. If we want to *show* a value of type $[\,a\,]$ we have always to make sure that *show* is also defined for values of type a.
2. We can call *show* on values of type $[\,a\,]$, without having defined *show* for a, as long as a itself is also a list type.
3. If we define *show* for $[\,a\,]$, then *show* for values of type a is automatically constructed.
4. We cannot define *show* for the polymorphic type $[\,a\,]$ since we cannot make this work for all possible types a at the same time.

7 Multiple Choice 7

Which of the following is a correct definition of *transpose*?

1. $\mathit{foldr}\ (\mathit{zipWith}\ (:))\ (\mathit{repeat}\ [\,])$
2. $\mathit{foldr}\ (\mathit{zipWith}\ (+\!\!+))\ (\mathit{repeat}\ [\,])$
3. $\mathit{foldr}\ (\mathit{zipWith}\ (:))\ (\mathit{repeat}\ [[\,]])$
4. $\mathit{foldl}\ (\mathit{zipWith}\ (+\!\!+))\ (\mathit{repeat}\ [\,])$

8 Multiple Choice 8

Which of the following is true?

1. Expressions of type $\mathit{IO}\ a$ cannot occur inside other expressions.
2. Expressions of type $\mathit{IO}\ a$ can only occur as a subexpression of expressions of type $\mathit{IO}\ b$ for a suitable b.
3. The expression $\mathit{return}\ (\mathit{return}\ a)$ can be replaced by $\mathit{return}\ a$.
4. The expressions **do** $\mathit{return}\ a$ is the same as $\mathit{return}\ a$.

9 Programming Exercise 1

The function $\mathit{numberOfCombinations} : [\mathit{Int}] \to \mathit{Int} \to \mathit{Int}$ computes the number of combinations in which we can select elements from the first list that sum up to the second argument. You may assume that all the elements in the first list are different and positive. Elements may be selected more than once. So is $\mathit{numberofCombinations}\ [5, 10, 20]\ 40$ gelijk equal to 9, because that is the length of the list $[[5,5,5,5,5,5,5,5], [5,5,5,5,5,5,10], [5,5,5,5,10,10], [5,5,5,5,20], [5,5,10,10,10], [5,5,10,20], [10,10,10,10], [10,10,20], [20,20]]$.

1. Define the function $\mathit{numberOfCombinations}$
2. Redefine the function in such a way that it uses arrays to remember common calls, and the overall complexity becomes linear in the second argument, assuming that indexing takes constant time.

10 Programming Exercise 2

1. Give the data type definition for a Huffman tree, i.e. $\mathit{Huff}\ a$ which encodes for values of some type a.
2. Write a function $\mathit{mkHuff} :: \mathit{Ord}\ a \Rightarrow [(a, \mathit{Weight})] \to \mathit{Huff}\ a$ which constructs the Huffman tree that is used in the encoding for the argument of mkHuff. Note that **type** $\mathit{Weight} = \mathit{Int}$.

11 Programming Exercise 3

In the context of the code:

```
data RoseTree = Node [RoseTree]
leaf = Node []
tt = Node [Node [  leaf,
                   Node [leaf,
                         leaf]],
           Node [  leaf,
```

```
                    Node [leaf,
                          Node [leaf,
                                leaf]]]]

printRoseTree = putStrLn . concat . intersperse "\n" . show'

map' f g []       = []
map' f g [e]      = [g e]
map' f g (x : xs) = f x : map' f g xs

show' :: RoseTree -> [String]
show' (Node []) =   ??
show' (Node ts) =   ??
                  . intersperse ??
                  . map' ?? ??
                  . map ??
                  $ ts
```

the command *printRoseTree tt* gives:

```
+-+-Leaf
| |
| +-+-Leaf
|   |
|   +-Leaf
|
+-+-Leaf
  |
  +-+-Leaf
    |
    +-+-Leaf
      |
      +-Leaf
```

Complete the function *show'* by filling in the ?? positions.

Name: **Student nr:** **Program: Inf/CKI/HBO/...**

1	2	3	4	5	6

If needed place code in two columns.

QUESTION 1: Type inference:

1. The type of *concat* . *map concat*? :
2. The type of *map concat* . *concat*:

QUESTION 2: The function *reachableFromRoot*:

QUESTION 3.1: The function *insertInHeap*:

QUESTION 3.2: The function *enumHeap*:

QUESTION 4.1: **instance** *Composable* $(\rightarrow)$ **where**

QUESTION 4.2: **instance** *Monad* $m \Rightarrow$ *Composable* $(M\ m)$ **where**

QUESTION 5: The function *fill*

QUESTION 6.1: **data** *Proposition* =

Question 6.2: *pChainl*...

Question 6.3: *pProp env* =