G6021: Comparative Programming

Exercise Sheet 5

1 Types

1. What are the types of the following Haskell expressions. Try to think what they might be before checking with the Haskell interpreter.

   (a) `(*)`

   (b) `(&&) True`

   (c) `\x -> \f -> f (f x)`

   (d) `tail [1,2,3]`

   (e) `error`

   (f) `\(x,y) -> x&&True`

   Answer:

   Each of these can be tested in Haskell by entering: `:t` followed by the expression at the prompt. The answers below are not exactly the same as what Haskell gives. What is the difference?

   (a) `(*) :: Int -> Int -> Int`

   (b) `(&&) True :: Bool -> Bool`

   (c) `\x -> \f -> f (f x) :: t -> (t -> t) -> t`

   (d) `tail [1,2,3] :: [Int]`

   (e) `error :: [Char] -> a`

   (f) `\(x,y) -> x&&True :: (Bool, t) -> Bool`

2 Lists and Pattern Matching

1. Write a function `equal` in Haskell syntax that takes two lists of elements (where each element has a type that is an instance of the `Eq` class) and checks whether they are equal (i.e., returns `True` if they have exactly the same elements in the same order, `False` otherwise). Give the most general (polymorphic) type for `equal`.

   Answer:

```
equal :: Eq a => [a] -> [a]  -> Bool

equal []     []  = True
equal (x:s)  []  = False
equal []  (y:p)  = False
equal (x:s)(y:p) = (x == y) && (equal s p)
```

2. Write a Haskell function to reverse a list. For example: `rev [1,2,3]` should give `[3,2,1]`.

   Answer: We will look at better reverse functions later, this one will do for now:

```
rev [] = []
rev (h:t) = rev t ++ [h]
```

3. Using `equal` and `rev` write a function `palindrome` that checks whether a list is a palindrome. A list is a palindrome if the list is the same in reverse. The lists [1,0,0,1], [True, False, True] and [0,1,2,3,3,2,1,0] are examples of palindromes.

   Answer:

   ```
   palindrome :: [a] ->  Bool
   palindrome l = equal l (rev l)
   ```

3 Data types

1. Using the definition of binary tree from Exercise sheet 3, write a function `mapTree` that will apply a function to all the node elements of the tree.

   Answer:

   ```
   mapTree f EmptyTree = EmptyTree
   mapTree f (Node v l r) = Node (f v) (mapTree f l) (mapTree f r)
   ```


4 If you have time

Take a look at the extra questions that you can find on:

`http://users.sussex.ac.uk/~im74/G6021/`.