COMP50001: Algorithm Design & Analysis

Sheet 2 (Week 3)

Exercise 2.1

Find a binary operation $(\diamond) :: (a \to a) \to (a \to a) \to (a \to a)$ and an element $\epsilon :: a \to a$ such that the set of functions of type $a \to a$ with $\diamond$ and ϵ forms a monoid.

Exercise 2.2

Given any two monoids $(M_1, \diamond_1, \epsilon_1)$ and $(M_2, \diamond_2, \epsilon_2)$, a *monoid homomorphism* from M_1 to M_2 is a function $h :: M_1 \to M_2$ such that

$$\begin{aligned} h\ (x \diamond_1 y) &= (h\ x) \diamond_2 (h\ y) \\ h\ \epsilon_1 &= \epsilon_2 \end{aligned}$$

Give three monoid homomorphisms from $([Int], +\!\!+, [\,])$ to $(Int, +, 0)$.

Exercise 2.3

Calculate the asymptotic time complexity of *concatl xs* below in terms of n and m where xs contains n lists, each containing m elements.

```
concatl :: [[a]] → [a]
concatl = foldl (++) []
```

Exercise 2.4

The *List* type class is shown in Figure 2.4. Complete the specification of the *List* type class by providing a default implementation for all the operations other than *fromList* and *toList*.

```
class List list where
  fromList :: [a] → list a
  toList :: list a → [a]
  normalize :: list a → list a
  empty :: list a
  single :: a → list a
  cons :: a → list a → list a
  snoc :: list a → a → list a
  head :: list a → a
  tail :: list a → list a
  init :: list a → list a
  last :: list a → a
  isEmpty :: list a → Bool
  isSingle :: list a → Bool
  length :: list a → Int
  (++) :: list a → list a → list a
  (!!) :: list a → Int → a
```

Figure 1: List class definition

Exercise 2.5

Implement an instance of *List* using standard lists $[a]$ without using functions from the *Prelude* other than the list constructors, and give the time complexities of each operation.

Exercise 2.6

Implement an instance of *List* using the following *Tree* type:

```
data Tree a = Tip | Leaf a | Fork (Tree a) (Tree a)
```

Ensure that the worst case complexity of $(+\!\!+)$ is $O(1)$. What is the worst case complexity of *head*?

Exercise 2.7

Define an instance of *List* using *DList* below, and give the complexities of all operations in terms of the length of the input list (assume all *DList* arguments to functions are built using the operations in *List*).

$$\textbf{newtype}\ \mathit{DList}\ a = \mathit{DList}\ ([a] \to [a])$$

Hint: $\mathit{fromList}\ \mathit{xs}\ =\ \mathit{DList}\ (\mathit{xs}{+\!\!+})$. Consider carefully whether the time complexity is affected by strict or lazy evaluation.

Exercise 2.8

Explain why the following implementation of fromList is undesirable in the last exercise:

$$\mathit{fromList}\ \mathit{xs} = \mathit{DList}\ ({+\!\!+}\mathit{xs})$$

Exercise 2.9

Prove or disprove the following assertions for the *DList* instance of *List* from Exercise (2.7).

1. $\mathit{fromList}\ (\mathit{toList}\ \mathit{dxs}) = \mathit{dxs}$ for any $\mathit{dxs} :: \mathit{DList}\ a$.
2. $\mathit{toList}\ (\mathit{fromList}\ \mathit{xs}) = \mathit{xs}$ for any $\mathit{xs} :: [a]$.