

Princípios de Programação

Exercícios

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2019/2020

Módulos

1. Escreva um módulo que implemente um novo tipo `Set`, que represente um conjunto ordenado de elementos que sejam **Ord**. Como representação do conjunto utilize listas ordenadas sem elementos repetidos. Considere as seguintes operações:
 - `empty`, criação de um conjunto vazio,
 - **`null`**, o dado conjunto é vazio?
 - `singleton`, criação de de um conjunto singular,
 - `member`, está um elemento em um conjunto?
 - **`insert`**, inserção de um elemento num conjunto,
 - `fromList`, um conjunto com os elementos constantes numa lista,
 - **`filter`**, filtrar os elementos que satisfazem o predicado,
 - `remove`, remoção de um elemento num conjunto,
 - **`union`**, união de dois conjuntos,
 - `intersection`, intersecção de dois conjuntos,
 - `difference`, o conjunto diferença,
 - `size`, o número de elementos no conjunto,
 - **`partition`**, dividir o conjunto em dois, aqueles que satisfazem o predicado e aqueles que não.

Até que ponto as seguintes funções seriam úteis na definição de muitas destas operações?

```
liftS :: ([a] -> [b]) -> Set a -> Set b
liftS f (S xs) = S (f xs)
```

```
liftS2 :: ([a] -> [b] -> [c]) -> Set a -> Set b -> Set c
liftS2 f (S xs) (S ys) = S (f xs ys)
```

Que outras funcionalidades podem elas permitir? Se exportássemos também estas funções, temos garantia que o seu uso pelo cliente iria sempre garantir as invariantes da representação (ie, lista ordenada sem repetições)?

2. Escreva um módulo que represente um mapa: uma estrutura de dados que mantém associações entre chaves e valores. Defina um tipo de dados para mapas com chaves do tipo `k` e valores do tipo `a`.

```
data Map k a = M [(k, a)]
```

Como representação utilize *listas de associação*, isto é, listas de pares chave-valor onde as chaves não devem aparecer repetidas. Certifique-se que as todas as operações do módulo mantêm as seguintes invariantes:

- (a) As chaves não aparecem repetidas,
- (b) A lista está ordenada pelas chaves.

Prepare operações para as seguintes operações:

- `empty :: Map k a`, o mapa vazio,
- `singleton :: k -> a -> Map k a`, construir um mapa com um único elemento,
- `insert :: Ord k => k -> a -> Map k a -> Map k a`, juntar uma entrada (chave, valor) ao mapa, substituindo o valor caso a chave já esteja no mapa,
- `null :: Map k a -> Bool`, está o mapa vazio?
- `size :: Map k a -> Int`, o número de elementos no mapa,
- `member :: Ord k => k -> Map k a -> Bool`, está a chave no mapa?
- `lookup :: Ord k => k -> Map k a -> Maybe a`, procurar uma chave no mapa, obtendo o valor associado (`Just` valor), ou `Nothing`, caso contrário,
- `delete :: Ord k => k -> Map k a -> Map k a`, apagar uma chave e o seu valor de um mapa,
- `unionWith :: Ord k => (a -> a -> a) -> Map k a -> Map k a -> Map k a`, união de dois mapas, utilizando uma função para combinar os valores de chaves duplicadas,

- `fromList :: Ord k => [(k, a)] -> Map k a`, para criar um mapa com os pares dados,
- `toList :: Map k a -> [(k, a)]`, para obter uma lista com os pares constantes num mapa.

Para mais detalhes e exemplos sobre estas funções consulte as funções com o mesmo nome no módulo `Data.Map.Strict`.