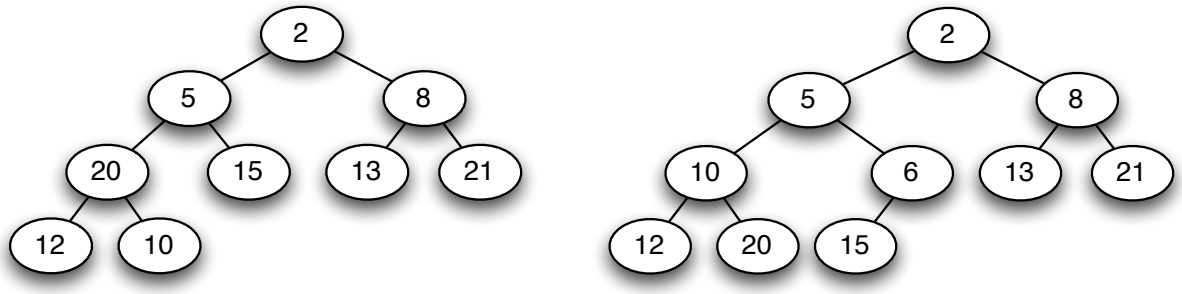
2º Teste

Programação Funcional – 1º Ano, LEI / LCC / MIEF
19 de Fevereiro de 2015

Duração: 90 min

Uma *heap* é uma árvore binária em que o elemento que está na raíz é menor do que todos os outros. Para além disso as subárvores também verificam essa propriedade.

Das árvores que se apresentam abaixo a da direita é uma heap enquanto que a da esquerda não é (note que a subárvore de altura 2 mais à esquerda não é uma heap).



Considere por isso o seguinte tipo para representar *heaps*.

```
data Heap a = Vazia
            | Nodo a (Heap a) (Heap a)
```

Defina as seguintes funções sobre *heaps*:

1. `quantos :: Heap a -> Int` que conta o número de elementos de uma *heap*.
2. `existe :: Ord a => a -> Heap a -> Bool` que testa se um dado elemento aparece numa *heap*.
3. `removeMin :: Heap a -> (a,Heap a)` que remove o menor elemento de uma *heap* não vazia (a raíz), devolvendo esse elemento bem como a `heap` resultante.
4. Defina `Heap a` como uma instância da classe `Eq` de tal forma que duas *heaps* são iguais se e só se tiverem exactamente os mesmos elementos. (Sugestão: comece por determinar a lista ordenada dos elementos de cada uma das *heaps*).
5. `randomHeap :: [a] -> IO (Heap a)` que calcula uma *heap* aleatória contendo todos os elementos da lista argumento. Use a função `randomRIO :: Random t => (t,t) -> IO t` para calcular um valor aleatório numa dada gama.

   Note que para uma lista não vazia, a escolha de qual o elemento a usar para a raíz é única. O que deve variar é quais os elementos que serão usados para construir as sub-árvores.