TPFI 2020/21
Hw 2: Tipi ed Efficienza

assegnato: 25 marzo 2021, consegna 11 aprile 2021

Esercizio 1 (Efficienza)

Definire una funzione Haskell che implementa l'usuale versione top-down ricorsiva di mergeSort, ma a ogni passo divide la lista in due con un'*unica* scansione (osservate che anche l'operazione `length` scansiona la lista e costa $\mathcal{O}(n)$ ed è perciò vietata!).

Esercizio 2 (Alberi binari & Funzionali)

Scrivere la funzione `foldrBT` che generalizza agli alberi binari la funzione `foldr` sulle liste.

Esercizio 3 (Alberi binari ed Efficienza) Il baobab è un albero dal tronco molto grosso. Diremo che un albero binario `(Node r Left Right)` è un *baobabBinTree* se l'etichetta r alla radice è maggiore di $max\{$(`sumLeft`), (`sumRight`)$\}$ e i sotto alberi `Left` e `Right` sono essi stessi dei baobab.

Scrivere una funzione `isBaobab::(Ord a, Num a) => BinTree a -> bool` che verifica questa proprietà.

Dare una definizione per ricorsione esplicita (che decompone gli alberi binari con pattern matching) e una che usa `foldrBT`.

Qualora non fosse questo il caso ai punti precedenti, dare anche una definizione di `isBaobab` di complessità lineare nel numero dei nodi.

Esercizio 4 (Lambda & Nat) Scrivere una funzione `toChurchNumeral` che trasforma un `Nat` (tipo dei naturali costruito a partire dai costruttori `Zero` e `Succ`) nel corrispondente numerale di Church. E la sua inversa, `fromChurchNumeral`.

Per stampare i naturali è sufficiente dichiarare la clausola `deriving Show`, mentre per verificare che il numerale di Church sia quello corretto, è sufficiente osservare che possiamo applicare a un numerale le funzioni `(+1)` e `0` e ottenere un intero.

$\star$Esercizio 5 (Lambda & Algoritmo di Hindley Milner) Definendo $\mathsf{tre} = \lambda sz.s(s(sz))$ il terzo numerale di Church, applicate l'algoritmo $\mathcal{W}$ per derivare il tipo di $\textbf{let}\ x\ =\ \mathsf{tre}\ \textbf{in}\ xx$.

A cosa riduce? Generalizzare con un generico numerale `n`.