Esercizi sui TDA Iterator, Tree e Binary Tree

NB: Gli esercizi 9-13 devono effettuare una visita ricorsiva dell'albero e NON devono invocare i metodi iterator() e positions() di Tree né altri metodi con analoghe funzionalità.

1. Scrivere la classe ElementIterator per IndexList e Sequence

2. Scrivere la classe VectorIterator che implementa Iterator mediante un vettore (Array List)

3. Scrivere un programma che stampa gli elementi di una lista mediante un iteratore

4. Scrivere la funzione eltsOfOddRank che prende in input una lista e restituisce un iteratore di tutti gli elementi di rango dispari della lista.

5. Aggiungere a **LinkedTree** i metodi
   a. depth(v) che restituisce la profondità del nodo
   b. height() che restituisce l'altezza dell'albero

6. Aggiungere alla classe **LinkedTree** il metodo
   public int distanza(TreePosition<E>u, TreePosition<E> v)
che restituisce la distanza tra i nodi u e v (numeri di rami tra u e v).

I seguenti metodi non devono invocare funzioni che restituiscono collezioni/iteratori di tutti i nodi o di tutti gli elementi dell'albero.

7. Aggiungere alla classe Linked Tree il metodo **arity**() che calcola l'arità dell'albero, ovvero il massimo delle arita` dei suoi nodi

8. Scrivere la funzione
   public static <E>Iterable<E> selectLeaves(Tree<E> T, int k)
che restituisce una collezione iterabile delle fogli dell'albero a profondità k.

9. Scrivere la funzione
   public static <E>boolean isContained(Tree<E> T, E x)
che restituisce true se e solo se x contenuto in qualche nodo dell'albero.

10. Scrivere la funzione
   p**ublic static int sumAllNodes(Tree<Integer> T)**
che restituisce la somma di tutti gli elementi dell'albero.

11. Aggiungere alla classe LinkedBinaryTree il metodo

public void attachLeaves()

che trasforma l'albero in un albero binario proprio (in cui ogni nodo interno ha esattamente 2 figli) attaccando a ciascun nodo con meno di 2 figli una o due foglie che contengono null come elemento.

13. Svolgere gli esercizi 6, 9, 10, 11 sostituendo Tree con BinaryTree e LinkedTree con LinkedBinaryTree.