Strutture Dati / Laboratorio di Algoritmi e Strutture Dati
Anno Accademico 2009/2010
20/7/10

Cognome e Nome:
Numero di Matricola:

Spazio riservato alla correzione

1	2	3	totale
/18	17	/25	/60

1. [18 punti] Scrivere la funzione

   public static <E> void removeDuplicates(IndexList<Integer>V, int k)

nella classe **ExArrayList_20_7** fornita dalla docente.

Istruzioni per lo svolgimento dell'esercizio:

- La funzione prende in input un vettore **V** di interi compresi tra **0** e **k-1** e per ciascun elemento del vettore, sostituisce con **null** tutte le occorrenze dell'elemento ad eccezione della prima occorrenza. Ad esempio se inizialmente **V** = <10 5 12 6 5 7 5 11 12 8 10 6 >, dopo aver invocato la funzione si ha **V**= <10 5 12 6 null 7 null 11 null 8 null nul >.
- La funzione sarà valutata fino ad un massimo di **18 punti** se il suo tempo di esecuzione risulterà essere lineare nel caso pessimo e fino ad un massimo di **13 punti** se il suo tempo di esecuzione risulterà essere quadratico nel caso pessimo.

La classe di test **ExArrayList_20_7** deve essere inserita nel pacchetto in cui si trova l'interfaccia **IndexList**.

2. [17 punti] Scrivere la funzione

   public static <V,E> boolean isAnEdgeCover(Graph<V,E> G,Iterable <Edge<E>> EC)

nella classe **ExGraph_22_7** fornita dalla docente.

Istruzioni per lo svolgimento dell'esercizio:

- La funzione prende in input un grafo **G** ed una collezione iterabile EC di archi appartenenti a **G** e restituisce **true** se e solo se per ogni vertice **v** di **G** c'è un arco in **EC** che incide su **v**.
- Si assuma che l' insieme dei vertici di **G** non sia vuoto.

La classe di test **ExGraph_22_7** deve essere inserita nel pacchetto in cui si trova l'interfaccia **Graph**.

.3. [25 punti] Scrivere la funzione

   public static <E> Position<E>findElt(Tree<E> T, E x)

nella classe **ExTree_20_7** fornita sul dischetto.

Istruzioni per lo svolgimento dell'esercizio:

- La funzione deve effettuare una visita ricorsiva **preorder** dell'albero **T** e restituire in output la position del primo nodo incontrato contenente **x**. Se nessun nodo dell'albero contiene l'elemento **x** allora la funzione restituisce **null**.
- **La funzione deve soddisfare i seguenti requisiti:**
  a. **deve arrestare la visita non appena incontra un nodo contenente x**
  b. **ad eccezione di children(), NON deve invocare funzioni che restituiscono o utilizzano collezioni/iteratori di uno o più nodi dell'albero.**

  Nel caso in cui non venga soddisfatto il requisito **a** la funzione sarà valutata al massimo **18 punti**. Nel caso in cui non vengano soddisfatti entrambi i requisiti **a** e **b** la funzione sarà valutata al massimo **12 punti**.
- Se l'albero è vuoto la funzione deve lanciare l'eccezione **EmptyTreeException**.

La classe di test **ExTree_20_7** deve essere inserita nel pacchetto in cui si trova l'interfaccia **Tree**.