Esercitazione 6: primo esercizio

Si vuole realizzare una coda a max-priorità con un max-heap ternario. Si consideri quindi il package PQ composto da:
-PQ.java: interfaccia che specifica le operazioni della coda a priorità
-PQNode.java: classe che memorizza una coppia (key,elem)
-PQHeap.java: classe che realizza le code a priorità con un heap ternario.

Si richiede di completare l'implementazione della classe PQHeap in modo efficiente.

Esercitazione 6: secondo esercizio

Def: un albero è k-ario se è vuoto oppure se tutti i suoi nodi hanno al più k figli. Un albero k-ario è completo se ogni suo nodo interno ha esattamente k figli e tutte le foglie si trovano allo stesso livello.

Dato il package Trees si vuole aggiungere alla classe GenTree.java il seguente metodo:

```
// pre: k >= 0
// post: ritorna true sse l'albero e' k-ario completo
public boolean kcompleto(int k) {…}
```

Si richiede di implementare il metodo utilizzando la ricorsione e definendo, se necessario, ulteriori metodi privati di supporto.

Esercitazione 6: terzo esercizio

Dato il package Trees si vuole aggiungere alla classe GenTree.java il seguente metodo:

```
// pre: k >= 0
// post: ritorna true sse tutti i suoi nodi interni
//       hanno esattamente k figli.
public boolean kfigli(int k) {…}
```

Si richiede di implementare il metodo utilizzando la ricorsione e definendo, se necessario, ulteriori metodi privati di supporto.

Esercitazione 6: consegna

Primo esercizio: consegnare **OBBLIGATORIO**
- la classe PQHeap.java

Secondo esercizio: consegnare
-l'intera classe GenTree.java
-dire qual è la complessità dell'algoritmo proposto

Terzo esercizio: consegnare
-l'intera classe GenTree.java
-dire qual è la complessità dell'algoritmo proposto

ATTENZIONE: Il primo esercizio è obbligatorio. Potete invece scegliere se consegnare il secondo OPPURE il terzo. Chi vuole può consegnarli entrambi.

Consegnare entro Lunedì 8 Dicembre alle 23.55