* ESERCIZI DA ESAMI

Domanda B (7 punti) Scrivere una funzione `toTree(A)` che dato un array `A` organizzato a max-heap (dimensione A.heapSize), lo trasforma in un albero binario realizzato con strutture linked, ancora organizzato a max-heap e ritorna la radice di tale albero. Il nuovo albero è costituito da nodi `x` con i campi `x.p` (parent), `x.k` (chiave), `x.l` e `x.r` (figlio sinistro e figlio destro). Per allocare un nuovo nodo si assuma di avere a disposizione un costruttore `node()`. Valutare la complessità.

Domanda B (5 punti) Dare la definizione di albero binario di ricerca. Specificare l'albero ottenuto inserendo, con la procedura vista a lezione, a partire da un albero vuoto, i nodi aventi le seguenti chiavi: 5, 4, 8, 6, 12, 7. Si supponga che dall'albero così ottenuto si cancelli il nodo con chiave 5 e si indichi l'albero ottenuto. Sia per gli inserimenti che per la cancellazione, motivare sinteticamente il risultato ottenuto.

Domanda C (5 punti) Scrivere una funzione `diff(T)` che dato in input un albero binario di ricerca `T` determina la massima differenza di lunghezza tra due cammini che vanno dalla radice ad un sottoalbero vuoto. Ad esempio sull'albero ottenuto inserendo 1, 2 e 3 produce 2, su quello ottenuto inserendo 2, 1, 3 produce 0. Valutarne la complessità.

Esercizio 1 (7 punti) Sia T un albero binario i cui nodi x hanno i campi *x.left*, *x.right*, *x.key*. L'albero si dice *k-bounded*, per un certo valore k, se per ogni nodo x la somma delle chiavi lungo ciascun cammino da x ad una foglia è minore o uguale a k.

Scrivere una funzione `Bound(T,k)` che dato in input un albero T e un valore k verifica se `T` è k-bounded e ritorna un corrispondente valore booleano. Valutarne la complessità.

Esercizio 1 (10 punti) Realizzare una funzione `union(A1,A2,n)` che dati due array di interi `A1` e `A2`, organizzati a max-heap, con capacità n, restituisce un nuovo array `A`, ancora organizzato a max-heap con capacità $2n$, che contiene l'unione insiemistica dei valori contenuti in `A1` e `A2`. Si assuma che `A1` e `A2` non contengano duplicati e si faccia in modo anche anche l'array ottenuto come unione non contenga duplicati. Ad es. se `A1` contiene i valori `3,1,2` e `A2` contiene i valori `5,2` allora l'unione A conterrà i valori `5,3,1,2`, possibilmente non in questo ordine, ovvero l'elemento `2` non è duplicato. Valutare la complessità della funzione definita.

Qualora il risultato `A` potesse contenere duplicati ci sarebbero soluzioni più efficienti?