Fondamenti di Programmazione con Laboratorio

Prova di Laboratorio del 1 Febbraio 2019

(Tempo a disposizione: 2 ore e 30 minuti)

Esercizio

Dato lo scheletro di codice qui riportato (e contenuto nel file `bozza.c`), lo si completi aggiungendo l'implementazione della funzione e della procedura richieste.

```c
#include <stdio.h>
#include <stdlib.h>

struct nodoAlberoBinario {
   int info;
   struct nodoAlberoBinario *left;
   struct nodoAlberoBinario *right;
};
typedef struct nodoAlberoBinario NodoAlbero;
typedef NodoAlbero *AlberoBinario;

/* DICHIARAZIONE DELLE FUNZIONI/PROCEDURE DA DEFINIRE */
void insert(AlberoBinario *radice, int val);
int kth(AlberoBinario radice, int k);

/* PROCEDURA PER RILASCIARE LA MEMORIA */
void free_tree(AlberoBinario radice)
{
   if (radice != NULL) {
      free_tree(radice->left);
      free_tree(radice->right);
      free(radice);
   }
}

/* MAIN */
int main(void)
{
   int i, n, k, num;
   AlberoBinario radice = NULL;
   scanf("%d", &n);
   for (i = 0; i < n; ++i) {
      scanf("%d", &num);
      insert(&radice, num);
   }
   scanf("%d",&k);
   printf("%d\n", kth(radice,k));
   free_tree(radice);
   return 0;
}
```

La funzione `main` del programma contenuto in `bozza.c` legge una sequenza di $n \geq 2$ interi ed li inserisce in un albero binario di ricerca *non bilanciato* con la procedura `insert`. Il programma legge poi un valore $k \leq n$ e stampa l'elemento *k-esimo* tra quelli inseriti nell'albero (ovvero tale che esattamente $k-1$ dei valori inseriti nell'albero sono inferiori ad esso), utilizzando la funzione `kth`.

Al fine di consentire una corretta esecuzione della funzione `main`, le procedure da implementare devono rispettare la seguente specifica:

`insert`: La procedura `insert` deve prendere in ingresso un albero binario di ricerca e un valore intero `val`. Se `val` non è contenuto nell'albero, `insert` deve creare un nuovo nodo con `val` come valore, inserirlo nell'albero (mantenendo l'ordine dell'albero binario di ricerca). Altrimenti, `insert` non deve effettuare alcun inserimento nell'albero ricevuto in ingresso.

`kth`: La funzione `kth` deve prendere in input la `radice` di un albero binario di ricerca ed un valore `k`. La funzione `kth` deve restituire il k-esimo valore tra quelli memorizzati nell'albero di ricerca la cui radice è `radice`. *Si assuma che il numero n di interi memorizzati nell'albero sia non inferiore a due e che $k \leq n$.*

ATTENZIONE Il codice già contenuto nel file `bozza.c` non puó essere modificato in alcun modo. L'implementazione di quanto richiesto deve essere aggiunte *dopo* la funzione `main` contenuta nel file `bozza.c`.

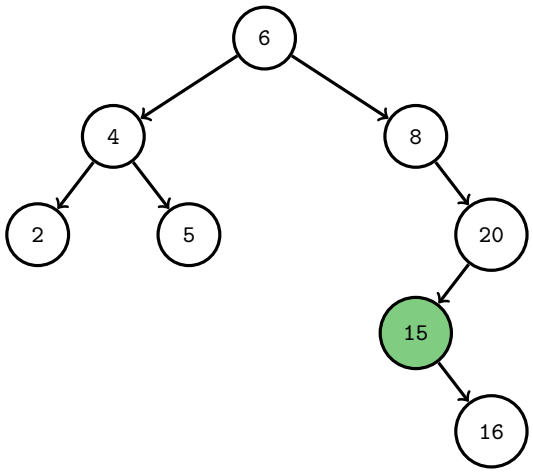
Esempi

NB: Il testo in verde è da intendersi come *commento* e dunque non fa parte dell'input. Inoltre, nell'albero corrispondente il k-esimo valore è evidenziato in verde.

Esempio 1

Input

```
8 # N
6 # Primo valore
4 # Secondo valore
2 # etc.
5
8
20
15
16
6 # k
```



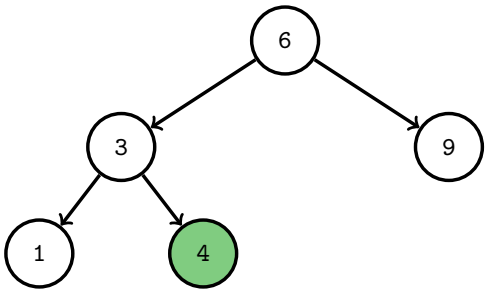
Input

```
15 # Sesto valore
```

Esempio 2

Input

```
6 # N
6 # Primo valore
9 # Secondo valore
3 # etc.
4
1
9
3 # k
```



Output

```
4 # Terzo valore
```

Suggerimenti

Attenzione a input/output La correzione della soluzione prodotta avverrà in maniera automatica, eseguendo dei test e confrontando l'output prodotto dalla soluzione prodotta con l'output atteso. Si presti quindi particolare attenzione alla formattazione dell'input e dell'output.

Si ricorda inoltre che è possibile verificare la correttezza della soluzione prodotta su un sottoinsieme dei input/output utilizzati. I file di input e output per i test sono nominati secondo lo schema:

```
input0.txt output0.txt
input1.txt output1.txt
...
```

Per effettuare una prova, utilizzare il comando per la redirezione dell'input. Ad esempio:

```
./compilato < input0.txt
```

effettua il test della soluzione prodotta sui dati contenuti nel primo file di input, assumendo che `compilato` contenga la compilazione della soluzione prodotta e che si trovi nella home directory. Si deve verificare che l'output coincida con quello contenuto nel file `output0.txt`.

Abilitare i messaggi di diagnostica del compilatore Compilare il codice usando le opzioni `-g -Wall` di `gcc`:

```
gcc -Wall -g soluzione.c -o soluzione
```

e risolvere gli eventuali *warning* restituiti dal compilatore, in particolar modo quelli relativi alle funzioni che non restituiscono un valore e ad assegnamenti tra puntatori di tipo diverso.

Provare la propria soluzione in locale Valutare la correttezza della soluzione sulla propria macchina accertandosi che rispetti gli input/output definiti dal `TestSet`. In particolare, si consiglia di provare tutti gli input/output previsti nel `TestSet` usando le istruzioni nella pagina precedente.

Consegna della soluzione Una volta consegnata, la vostra soluzione verrà valutata dal server compilando ed eseguendo su alcuni file di test. Si ricorda di avvisare i docenti una volta che il server ha accettato una soluzione come corretta, poiché la sola approvazione del server non garantisce il superamento della prova.