Ingegneria degli Algoritmi (A.A. 2014-2015)

Corsi di Laurea in Ingegneria Informatica e Automatica, Ingegneria dei Sistemi Informatici, e Laurea Magistrale in Ingegneria Informatica

Sapienza Università di Roma

Esame di laboratorio (11/09/2015) – Durata 1h 30'

Alberi binari

Si richiede di completare le funzioni di un albero binario con elementi di tipo intero realizzato mediante strutture e puntatori in C, con i seguenti prototipi dichiarati nella header `bintree.h`:

```
bintree* bintree_new(const bintree* left,
                     int elem,
                     const bintree* right);
```

Crea un nuovo albero binario ottenuto prendendo come radice `elem`, come sottoalbero sinistro `left` e come sottoalbero destro `right`.

```
void bintree_delete(bintree* t);
```

Dealloca tutti i nodi di un albero `t`.

```
int      bintree_get_elem  (const bintree* t);
bintree* bintree_get_left  (const bintree* t);
bintree* bintree_get_right (const bintree* t);
```

Dato un albero binario `t`, restituisce l'elemento della radice (`get_elem`), il sottoalbero sinistro (`get_left`) e quello destro (`get_right`).

```
void bintree_print(const bintree* t);
```

Stampa il contenuto dell'albero `t` usando la rappresentazione parentetica.

```
bintree* bintree_clone(const bintree* t);
```

Crea un nuovo albero ottenuto come clone dell'albero `t`. I nodi del nuovo albero devono essere distinti da quelli dell'albero `t`. [**da realizzare**]

```
int bintree_equal(const bintree* t1, const bintree* t2);
```

Restituisce 1 se gli alberi `t1` e `t2` sono uguali (uguaglianza profonda), e 0 altrimenti. [**da realizzare**]

```
int bintree_get_height(const bintree* t);
```

Restituisce l'altezza dell'albero `t`, dove un albero vuoto ha altezza 0, un albero con un solo nodo ha altezza 1, ecc. [**da realizzare**]

Scrivere il codice C delle tre funzioni `bintree_clone`, `bintree_equal` e `bintree_get_height` nel file `bintree.c`. E' utile esaminare il main di prova `main.c` per vedere un esempio di operazioni su alberi.

Compilazione e test.

- *Compilazione programma di test*: dare il comando `make`, che genera il file eseguibile `bintree` contenente un programma di test.
- *Test correttezza*: eseguire il programma di test `./bintree`.

E' consigliabile modificare il main per fare ulteriori test e verificare la correttezza delle funzioni scritte, purché non si modifichino i prototipi delle funzioni dichiarate in `bintree.h`.

Nota bene: testare la correttezza dell'uso della memoria usando il tool Valgrind:

```
valgrind ./bintree
```