Laboratorio di Linguaggi di Algoritmi e Strutture Dati
Prova di laboratorio - 18 settembre 2002 - turno 1

Dato il tipo `tree` degli alberi ad apertura illimitata con ordine tra i figli, senza albero vuoto ed etichettati da interi, come definito nel file `lib.h`:

```
typedef struct cell *tree;

struct cell{
  int info;
  tree child,sibling;
};
```

1. **(punti: -2 ; 0)** Definire la funzione `bool member(tree t,int i)` che restituisce `TRUE` se `i` è un'etichetta di `t`, `FALSE` altrimenti.

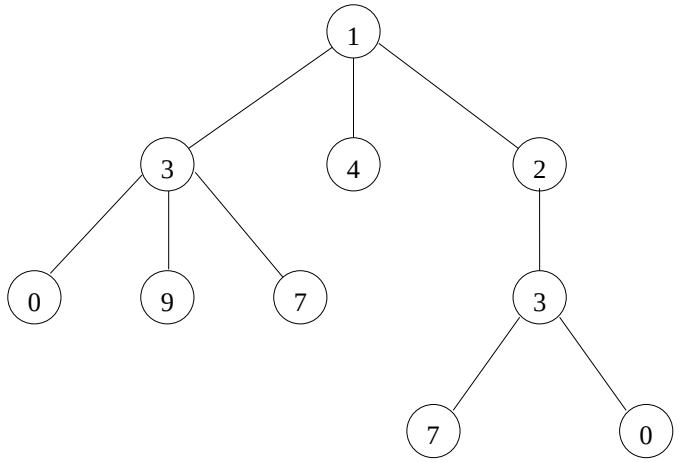
2. **(punti: 0 ; 1)** Data la seguente definizione induttiva di uguaglianza tra alberi:
   t è uguale a t' se e solo se

   - l'etichetta della radice di t è uguale all'etichetta della radice di t';
   - t e t' hanno lo stesso numero di figli;
   - se $t = (n, <t_1, \ldots, t_n>)$ e $t' = (n', <t'_1, \ldots, t'_n>)$, allora per ogni $i \in \{1, \ldots, n\}$ t_i deve essere uguale a t'_i.

   Definire la funzione `bool equal(tree t1,tree t2)` che implementa tale uguaglianza.

3. **(punti: 0 ; 2)** Utilizzando il tipo `list` delle liste di interi e le primitive `empty_list`, `make_list` e `cat` definite in `lib.o` (si veda anche il corrispondente header `lib.h`), definire la funzione `list frontier(tree t)` che restituisce la frontiera di `t`, ossia la lista delle etichette di tutte le foglie di `t` in ordine da sinistra a destra (con la solita convenzione che il primo figlio di `t` è quello che "viene disegnato più a sinistra").

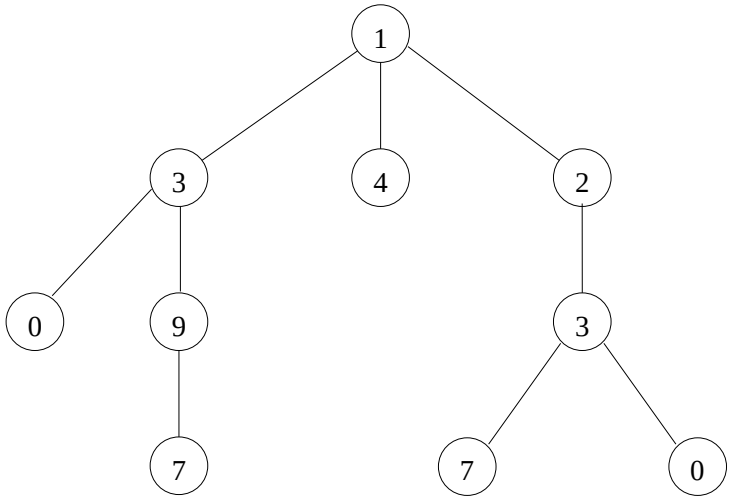
   Ad esempio, se `t` contiene il seguente albero:



   Allora `frontier(t)` restituisce la lista `[0, 9, 7, 4, 7, 0]`.

4. **(punti: 0 ; 2)** Definire la funzione `void print_leaves(tree t,int i)` che stampa in ordine da sinistra a destra le etichette di tutte le foglie di `t` che si trovano a livello `i` (dove 0 è il livello della radice, 1 quello dei suoi figli e così via).

   Ad esempio, se `t` contiene il seguente albero:



Allora `print_leaves(t,3)` stampa i valori `7 7 0`.

Per provare i programmi

In `lib.o` (si veda anche il corrispondente header `lib.h`) sono definite tre funzioni utili per effettuare delle prove sui programmi.

- `tree get_tree(int h,int w)`: restituisce un albero generato casualmente con etichette nell'insieme $\{0, \ldots, 9\}$, la cui altezza è al più `h` e la cui apertura massima è al più `w` (quindi ogni nodo non può avere più di `w` figli).
- `void print_tree(tree t)`: stampa su video le etichette di `t` per livelli, in ordine dalla radice alle foglie e da sinstra a destra in modo che ogni riga dell'output corrisponda a un livello. Le etichette di nodi"fratelli" sono racchiuse tra coppie di parentesi `< >`. Le parentesi che non racchiudono alcuna etichetta corrispondono a nodi foglia.

  Ad esempio, l'albero dell'esempio dell'esercizio 4 viene stampato nel seguente modo:

  ```
  < 1 >
  < 3 4 2 >
  < 0 9 > < > < 3 >
  < > < 7 > < 7 0 >
  < > < > < >
  ```

- `void print_list(list l)`: stampa su video gli elementi di `l` in ordine a partire dalla testa.