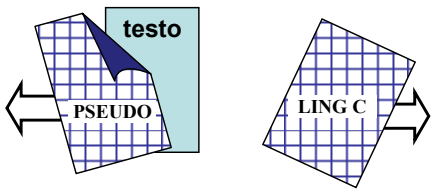
Algoritmi e Strutture di Dati – A.A. 2015-2016
Prova scritta del 22 giugno 2016 – D.M. 270
Libri e appunti chiusi
Tempo = 2:00h

☐ NOTE (Es.: Ho bisogno di una correzione veloce in quanto…) _____

_____

Cognome: _____ Nome: _____Matricola:_____



- **CONSEGNA PSEUDOCODIFICA E LINGUAGGIO C SU DUE FOGLI PROTOCOLLO SEPARATI**
- **METTI IL TESTO DENTRO LA PARTE DI PSEUDOCODIFICA**
- **PUOI SCRIVERE (E CONSEGNARE) A MATITA**

PSEUDOCODIFICA

Negli esercizi seguenti un grafo non orientato è rappresentato con un array A in cui ogni elemento A[*u*] è un riferimento al primo elemento della lista di adiacenza doppiamente concatenata del nodo *u* (con i campi `prev`, `info` e `next`). Essendo il grafo non orientato esiste un arco (*u*,*v*) per ogni arco (*v*,*u*). Un albero binario `T1` è un oggetto che ha il solo campo `T1.root` che è un riferimento al nodo radice dell'albero, dove ogni nodo ha i campi `parent` (genitore), `left` (figlio sinistro), `right` (figlio destro) e `info`.

Esercizio 1

Scrivi lo pseudocodice della procedura **FOGLIE-COMPONENTE**(T,A) che accetti in input un albero binario di interi e grafo non orientato A e verifichi che gli indici delle foglie dell'albero corrispondano nel grafo a dei nodi che appartengono tutti alla stessa componente connessa. Puoi assumere che ogni nodo dell'albero sia anche nel grafo e viceversa.

Esercizio 2

Discuti la complessità computazionale della seguente procedura nel caso peggiore fornendo O-grande, Omega e Theta in funzione del numero n di elementi dell'albero.

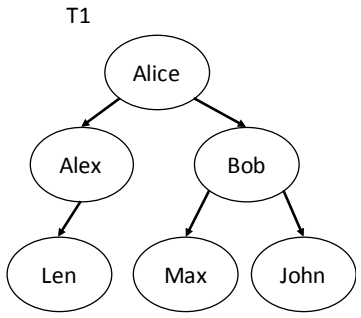
```
FUNZIONE(T) /* T è un albero binario di interi */
L.head = NULL   /* L è una nuova lista (vuota) di interi */
FUNZ-RIC(T.root,L)
return L

FUNZ-RIC(v,L)
if(v==NULL) return
if(TEST(v))
      AGGIUNGI-IN-CODA(L,v.info)
else
      FUNZ-RIC(v.left,L)
      FUNZ-RIC(v.right,L)
```

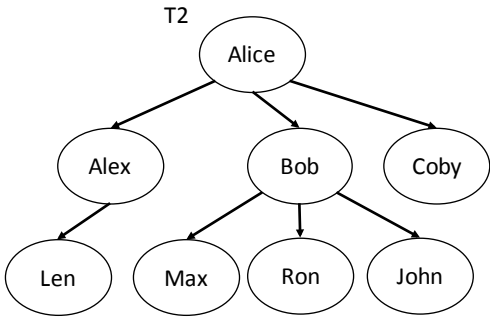
Supponi che TEST(v) abbia complessità Theta(1) e che AGGIUNGI-IN-CODA abbia complessità Theta(x) dove x è la lunghezza della lista.

LINGUAGGIO C

Si consideri la libreria **genealogia.h** che implementa quanto segue:

```
typedef char stringa[20];

typedef struct elem {
                stringa nome;
                struct elem* sx;
                struct elem* dx;
  } nodo_albero;

typedef nodo_albero* BTree;

typedef struct elem2 {
                stringa nome;
                struct elem2* primofiglio;
                struct elem2* fratello;
  } nodo_alberoN;

typedef nodo_alberoN* Ntree;
```





Tale libreria implementa il tipo **stringa** (stringhe di lunghezza massima 20 caratteri), il tipo **BTree** (alberi binari i cui nodi contengono stringhe) e il tipo **NTree** (alberi di grado arbitrario i cui nodi contengono stringhe). Sia il tipo **Btree** che il tipo **Ntree** implementano un albero genealogico (l'arco da un nodo n1 ad un nodo n2 rappresenta la relazione di discendenza di n2 da n1).

Estendere tale libreria implementando in linguaggio C i seguenti metodi:

- `int` **`verificaDiscendenza`**`(Btree T, stringa S1, stringa S2)` che dato un albero binario T, e due stringhe S1 e S2 ritorni 1 se il nodo contenente S2 discende dal nodo contenente S1, 0 altrimenti. Nel caso in cui l'albero T sia vuoto, la stringa S1 non esista in T o la stringa S1 non esista in T (o entrambe), la funzione deve ritornare 0. Ad esempio **`verificaDiscendenza`**`(T1, "Alice", "John")` ritornerà 1, mentre **`verificaDiscendenza`**`(T1, "Max", "John")` ritornerà 0.

- `int` **`verificaAntenato`**`(Btree T, stringa A, stringa S1, stringa S2)` che dato un albero binario T e tre stringhe A, S1 e S2, ritornerà 1 se i nodi contenenti S1 e S2 discendono entrambi dal nodo contenente A, 0 altrimenti. Nel caso in cui l'albero T sia vuoto, la stringa S1 o la stringa S1 o la stringa A non esista in T (o tutte e tre), la funzione deve ritornare 0. Ad esempio **`verificaAntenato`**`(T1, "Alice", "John", "Len")` ritornerà 1, mentre **`verificaAntenato`**`(T1, "Alex", "Max", "John")` ritornerà 0.

- `int` **`contaFratelli`**`(Ntree T, stringa S)` che dato un albero di grado arbitrario T e una stringa S, conti quanti fratelli ha il nodo contenente S. Nel caso in cui T sia vuoto oppure non esiste un nodo contenente S, la funzione ritornerà 0. Ad esempio `int` **`contaFratelli`**`(T2, "Ron")` ritornerà 2.

E' possibile implementare altri metodi di supporto ma non è ammesso modificare la struttura dati fornita.