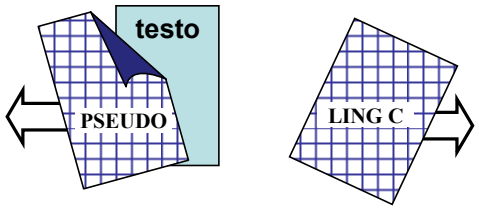
Algoritmi e Strutture di Dati – A.A. 2014-2015
Prova scritta del 3 luglio 2015 – D.M. 270
Libri e appunti chiusi
Tempo = 2:00h

☐ 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*).

Esercizio 1

Scrivi lo pseudocodice della procedura **CONNESSE-DUE-ARCHI**(A) che accetti in input un grafo non orientato A e produca in output il numero delle componenti connesse di A i cui nodi hanno al massimo due archi.

Esercizio 2

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

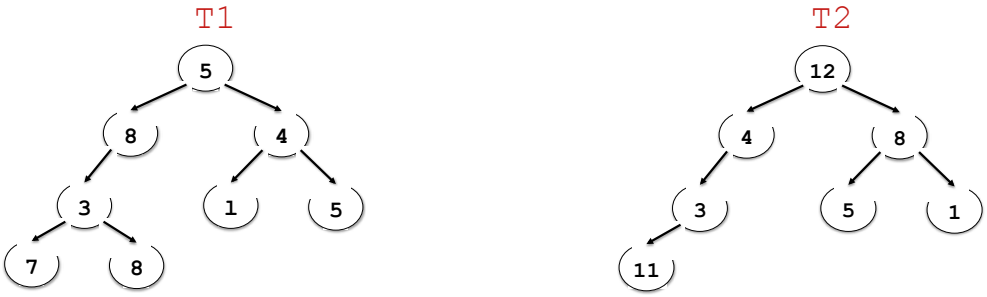
```
FUNZIONE(T) /* T è un albero di grado arbitrario di interi rappresentato
               tramite la convenzione figlio-sinistro fratello-destro */
/* L è una nuova lista (vuota) di interi */
L.head = NULL
FUNZ-RIC(T.root,L)
return L

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

Supponi che AGGIUNGI-IN-CODA faccia un numero di operazioni proporzionale alla lunghezza della lista.

LINGUAGGIO C

Si consideri un albero binario i cui nodi contengono un intero, come i due alberi qui sotto.



Si supponga di avere a disposizione una libreria `Tree.h` che implementa i seguenti tipi

```
typedef struct elem {
   int info;
   struct elem* sx, dx;
} nodo_albero;

typedef nodo_albero* albero;
```

e con le seguenti funzionalità (già implementate):

- `int* creaArray(albero T, int m)`, che dato un albero `T` e un intero `m`, restituisce un array contenente i valori interi contenuti nei nodi al livello `m` di `T`. Ad esempio `creaArray(T1, 2)` restituisce l'array `[3,1,5]`.
- `int confrontaArray(int* A1, int* A2, int m)`, che dati due array di interi `A1` e `A2` restituirà `1` se `A1` e `A2` hanno entrambi lunghezza `m` ed `m` elementi in comune, `0` altrimenti. Ad esempio se `A1 = [4,1,5]` e `A2 = [5,4,1]`, `confrontaArray(A1, A2, 3)` restituisce `1`.

Utilizzando la libreria `Tree.h`, si richiede di implementare in linguaggio C i seguenti metodi:

1) `int conta_livello(albero T, int m)`, che restituisca il numero di nodi presente al livello `m` dell'albero `T`. Se l'albero `T` è vuoto oppure non esiste il livello `m` nell'albero, la funzione restituirà il valore `0`. Ad esempio `conta_livello(T1, 2)` restituirà il valore `3`.
2) `int confronta_alberi(albero T1, albero T2)` che dati due alberi `T1` e `T2`, conta quanti livelli hanno in comune `T1` e `T2`; `T1` e `T2` hanno un livello `m` in comune se i valori interi contenuti dai nodi del livello `m` di `T1` ricorrono anche nei nodi dello stesso livello `m` in `T2`. Ad esempio `confronta_alberi(T1, T2)` restituirà il valore `2` (nell'esempio `T1` e `T2` hanno in comune il livello `1` e il livello `2`).
3) `int ModificaFile(albero T)`, che dato un albero `T` deve accedere ad un file binario di nome `"livelli.dat"` i cui record hanno due campi interi: `X` e `Y`. Il campo `X` contiene un valore intero che corrisponde al numero di un livello, mentre il campo `Y` contiene un intero che corrisponde al numero di nodi presenti al livello `X` dell'albero `T`. La funzione deve accedere al file e in caso modificarlo nel caso in cui il campo `Y` contiene un numero di nodi non corrispondente a quello effettivo al livello `X` dell'albero `T`. Ad esempio dato l'albero `T1` dell'esempio e il seguente file binario

0	2	1	2	2	2	3	2

   tale file sarà modificato in

0	1	1	2	2	3	3	2

E' possibile definire metodi di supporto e utilizzare ogni libreria conosciuta. NON è possibile modificare la libreria `Tree.h`