Esercitazione di laboratorio n. 6

(Caricamento sul portale entro le 23.59 del 21/11/2016 di tutti gli esercizi)

Esercizio n. 1: Operazioni su alberi
Si scriva un programma in linguaggio C che esegua le seguenti operazioni su alberi binari:

- conteggio del numero di nodi con 2 figli tra due livelli L_1 e L_2 dell'albero
- conteggio del numero di archi che separano due nodi dell'albero**.**

Per svolgere questo esercizio, completare il file `albero.c` in cui sono già state inserite le funzioni di lettura, i prototipi delle funzioni da realizzare nonché definite le strutture dati necessarie a rappresentare gli alberi stessi.

Il formato dei file di ingresso accettato dalla libreria è composto da una serie di terne, in ragione di una per riga, a rappresentare la chiave (stringa di al massimo 10 caratteri) di ogni nodo e una coppia di valori binari a indicare la presenza (1) o assenza (0) dei figli destro e sinistro, rispettivamente.

Si assuma che la chiave di un nodo sia unica e che esista un criterio di ordinamento per l'albero stesso: tutti i figli sinistri (destri) di un nodo hanno chiave alfabeticamente inferiore (superiore) al nodo stesso.

Per maggiore leggibilità, i due file di esempio sono caratterizzati da un'indentazione crescente all'aumentare della profondità a cui un dato nodo si trovi.
L'esempio proposto nel file di ingresso `alb1.txt` è riprodotto a seguire.

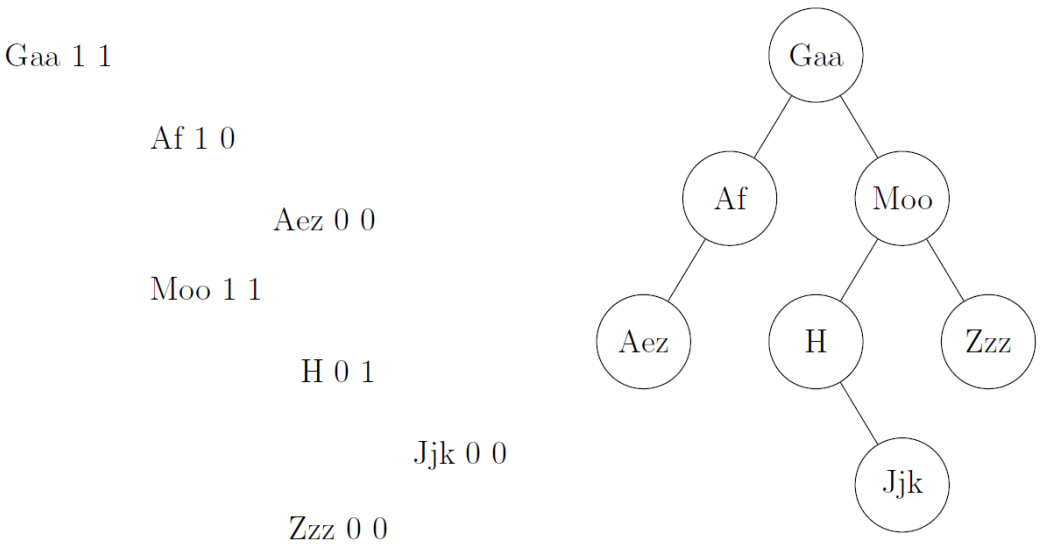

Figura 1: Un esempio di albero nella sua rappresentazione testuale (sinistra) e grafica (destra).

Esercizio n. 2: Split di stringhe
Si scriva una funzione il cui prototipo sia:

```
nodo *splitStringa(char *str, char sep)
```

Tale funzione riceve come parametri una stringa `str` e un carattere `sep`, con funzione di separatore.
Lo scopo della funzione è suddividere la stringa ricevuta come parametro in una sequenza di sottostringhe, sulla base del separatore ricevuto in input. La sequenza di sottostringhe deve essere memorizzata in una lista di stringhe allocate dinamicamente. La testa della lista sia resa al chiamante mediante il valore di ritorno della funzione. Al termine si percorra la lista stampando le sottostringhe.

Si consideri ad esempio la stringa `str = "aaaa.bbb.cc.d"` e il separatore `sep = '.'`
La lista ritornata dalla funzione avrà una forma del tipo:

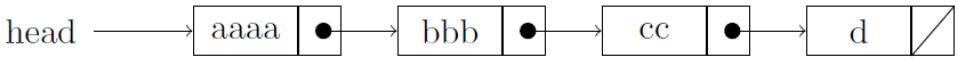

Figura 2: Risultato della funzione `splitStringa`

Esercizio n. 3: Lista di punti del piano
Si definisca una `struct` adatta a memorizzare coordinate del piano cartesiano. Si scriva una funzione in grado di ricevere in input i valori `x` e `y` di un punto del piano ed aggiungere tale punto a una lista ordinata di punti del piano, inizialmente vuota.
L'inserimento sia ordinato per distanze crescenti dall'origine del piano. Si ricorda che la distanza di un punto dall'origine può essere calcolata come:

$$\boldsymbol{d(P,O) = \sqrt{x_p^2 + y_p^2}}$$

Al termine dell'inserimento si stampi la lista ordinata.