15.10 Esercizi

* 1. Scrivere una funzione ricorsiva per visitare in ordine differito un albero binario. Si consideri la stessa struttura dei nodi vista nel primo esempio di questo capitolo.

* 2. Si consideri la seguente definizione di nodo di albero binario:

```
struct nodo {
       int inf;
       int occorrenze;
       struct nodo *alb_sin;
       struct nodo *alb_des;
```

```
};
```

Costruire la funzione `crea_nodo2` modificando `crea_nodo` del primo esempio di questo capitolo, in modo che venga calcolato il numero di occorrenze multiple dello stesso valore, eventualmente immesse dall'utente, e vengano memorizzate nel nodo stesso nel campo `occorrenze`.

* 3. Modificare la funzione che visita in forma simmetrica l'albero binario in modo che visualizzi, oltre al valore di ogni nodo, il corrispondente numero di occorrenze, memorizzate mediante la funzione `crea_nodo2` dell'esercizio precedente.

* 4. Scrivere una funzione ricorsiva per visitare in ordine differito un albero implementato mediante liste multiple. Si consideri la stessa struttura dei nodi vista negli esempi di questo capitolo:

```
struct nodo {
       char inf;
       struct nodo *figlio;
       struct nodo *p_arco;
     };
```

5. Scrivere una funzione che calcoli il numero di livelli di un albero binario memorizzato in una lista doppia.

6. Scrivere un programma che da un albero binario, memorizzato in una lista doppia, elimini (rilasciando opportunamente la memoria) il sottoalbero la cui etichetta della radice viene passata in ingresso dall'utente.

* 7. Ampliare il programma di implementazione di un grafo mediante una lista di successori esaminato nel capitolo, in modo che accetti in ingresso il valore di un'etichetta e visualizzi le etichette di tutti i nodi da esso raggiungibili. Per la scansione dei nodi connessi si utilizzi una funzione ricorsiva.
[Prestare attenzione al fatto che la scansione può portare a un ciclo infinito se nel percorso si passa per più di una volta sullo stesso nodo.]

8. Scrivere una funzione che un dato grafo, memorizzato in una matrice di adiacenze, e date in ingresso le etichette di due nodi, verifichi se esiste un arco che li collega, nel qual caso lo cancelli.

9. Verificare se, dato un grafo memorizzato in una matrice di adiacenze, e date in ingresso le etichette di due nodi, dal primo nodo è possibile arrivare al secondo attraverso un cammino di archi orientati.

10. Risolvere i due esercizi precedenti ma con il grafo memorizzato in una lista di successori.

11. Disegnare l'albero binario che si ottiene fornendo al programma del Listato 15.1 i seguenti dati:
35, 12, 91, 7, 13, 108, 64, 66, 19, 12, 8, 0.

12. Scrivere una funzione che stabilisca se due alberi binari sono uguali.

13. Dato l'albero binario costruito con il Listato 15.1 realizzare due funzioni, la prima che calcoli la somma totale delle informazioni contenute nei nodi, l'altra che determini il maggiore e il minore.

14. Scrivere una funzione che calcoli il numero di livelli di un albero memorizzato in una lista multipla.

15. Dato un albero ordinato, realizzare un programma completo per la sua gestione, dando all'utente un menu che comprenda almeno le opzioni per l'inserimento di un valore, l'eliminazione, le visite, la visualizzazione delle sole foglie.