Complementi di Programmazione

Esercitazione 11

- Si considerino alberi binari con valori di tipo intero, **positivi**.
- Scrivere dei test nel main per verificare che le funzioni scritte siano corrette.
- Svolgere gli esercizi sia accedendo solo tramite le funzione del tipo astratto (ove possibile, vedere lista.h, albero.h), sia accedendo alla rappresentazione interna.

Esercizio 11.1

Implementare la funzione

```
TipoInfoAlbero trova_massimo(TipoAlbero a);
```

che, dato un albero binario, restituisca il valore massimo contenuto nei nodi dell'albero. Se l'albero è vuoto, si ritorni -1.

Esercizio 11.2

Implementare la funzione:

```
TipoInfoAlbero somma_foglie(TipoAlbero a);
```

che, dato un albero binario, restituisca la somma di tutti i valori contenuti nei nodi foglia.

Esercizio 11.3

Implementare la funzione

```
int cerca_livello(TipoAlbero a, TipoInfoAlbero v);
```

che, dati un albero binario `a` e un valore `v`, restituisca il livello dell'albero dove si trova `v`. Se `v` non e' presente all'interno dell'albero ritornare -1. Se sono presenti più nodi con lo stesso valore, si restituisca il livello del nodo più a sinistra.

Esercizio 11.4

Implementare la funzione:

```
int conta_dispari(TipoAlbero a);
```

Che, dato un albero binario, restituisca il numero di nodi che soddisfano la seguente condizione: la somma tra il valore del nodo e i suoi figli è dispari.

Esercizio 11.5

Implementare la seguente funzione

```
int somma_singoli(TipoAlbero a);
```

che, dato in input un albero binario, restituisca la somma dei valori dei nodi che hanno un solo successore.

Esercizio 11.6

Implementare la seguente funzione:

```
TipoLista albero_lista(TipoAlbero a);
```

Che, dato un albero binario di ricerca, restituisca la lista dei valori ordinati, in modo decrescente.

Esercizio 11.7

Implementare la seguente funzione

```
TipoLista percorso_lungo(TipoAlbero a);
```

Che, dato un albero binario, restituisca la lista dei nodi contenuti nel percorso più lungo dalla radice a una delle foglie. Se esistono diversi percorsi di dimensione massima, si restituisca quello più a sinistra.

Esercizio 11.8

Implementare la funzione:

```
void somma_sottoalbero(TipoAlbero a);
```

che modifica il valore di ogni nodo dell'albero `a`, scrivendo come valore la somma dei valori dei sottoalberi di quel nodo.

Esercizio 11.9

Implementare la funzione:

```
void scambia_foglie(TipoAlbero a);
```

che scambia il contenuto di tutte le coppie di foglie che hanno lo stesso padre.

Esercizio 11.10

Implementare la funzione:

```
void max_figli_livello(TipoAlbero a, int livello);
```

che modifica il valore di ogni nodo dell'albero `a` al livello `livello`, scrivendo come valore il massimo tra i valori dei figli di quel nodo.

Esercizio 11.11

Implementare la funzione:

```
void scambia_foglie_livello(TipoAlbero a, int livello);
```

che scambia il contenuto di tutte le coppie di foglie che hanno lo stesso padre e si trovano al livello `livello`.

Esercizio 11.12

Implementare la seguente funzione

```
void set_albbinric(TipoAlbero a, TipoLista l);
```

che dato un albero `a` e una lista `l` di dimensioni pari al numero di nodi dell'albero contenente valori ordinati, modifichi il contenuto dell'albero `a` in modo che esso diventi un albero binario di ricerca con i valori contenuti in `l`.