Tecniche di Programmazione

Esercitazione 12

- Scrivere dei test nel main per verificare che le funzioni implementate siano corrette.

Esercizio 12.1

Implementare la funzione

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

che, dati un albero binario `a` e un valore `v`, restituisca il livello dell'albero dove si trova `v`.
Si restituisca il livello massimo dell'albero dove si trova il nodo, -1 in caso il valore a non sia presente.

Esercizio 12.2

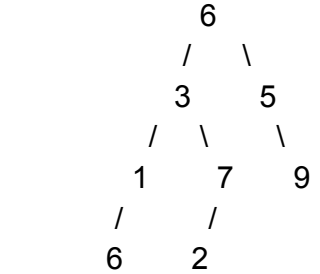
Implementare la funzione

```
int singleChildSum(TipoAlbero a);
```

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

Esempio

Dato il seguente albero src in ingresso:



`singleChildSum(src)`
dovrà restituire: 1 + 7 + 5 = 13

Esercizio 12.3

Implementare la seguente funzione

```
TipoListaSCL listaNodiFoglia(TipoAlbero a);
```

Che, dato un albero binario, restituisca una lista con i valori contenuti nelle foglie (seguendo l'ordine da sinistra a destra).

Esercizio 12.4

Implementare la funzione

```
TipoListaSCL listaPercorsoMassimo(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 12.5

Scrivere una funzione

```
void rimuoviAlberi(TipoAlbero *a, TipoInfoAlbero info);
```

che elimina e dealloca ogni sottoalbero dell'albero `a` che ha per radice un nodo con valore `info`.

Esercizi Alberi N-ari

- Ripetere gli esercizi nella prima parte considerando come input un albero N-ario con relativa libreria.

Esercizio 12.6

Implementare una funzione

```
TipoAlbero piuFigli(TipoAlbero a);
```

che restituisca il nodo dell'albero che ha il maggior numero di figli, seguendo una visita in preordine.