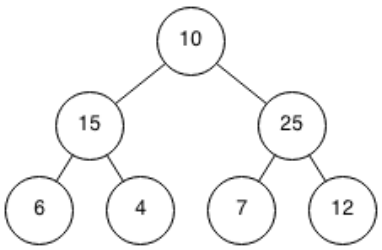
Laboratorio

Definire una funzione di ricerca di un elemento in un generico albero binario, utilizzando uno degli schemi di visita per gli alberi e la classe `AlberoB` (ed eventualmente altre classi di utilità).
La funzione prende in input due parametri: l'albero da visitare e l'oggetto da cercare nell'albero. La funzione ritorna il sottoalbero che ha come radice l'oggetto cercato nel caso in cui esso esista, o un albero vuoto nel caso in cui l'oggetto da cercare non sia presente nell'albero; si restituisca il primo sottoalbero trovato, nel caso in cui l'albero contenga più volte l'oggetto da cercare.

La funzione deve essere utilizzata in un main che legga un albero da standard input, secondo i seguenti accorgimenti:

- si legga prima la radice dell'albero generale,
- per ogni altro nodo, l'input deve specificare:
  - il valore del nuovo nodo da inserire,
  - il valode del nodo che si desidera essere il padre del nuovo elemento,
  - la direzione di inserimento: sinistra viene indicato con `s`, destra con `d`.
- la fine dello stream viene indicata con -1.



Per esempio, prendiamo l'albero mostrato. Lo stream di input che identifica il seguente albero, secondo le specifiche di cui sopra, è il seguente:

```
10
15:10 s
25:10 d
6:15 s
4:15 d
7:25 s
12:25 d
-1
```

dove la prima riga indica il valore (in questo caso 10) del nodo radice, la seconda indica che un nodo con valore 15 viene inserito come figlio sinistro del nodo con valore 10, etc.

Il main deve leggere l'input secondo tale formato, quindi richiamare la funzione di ricerca definita precedentemente. Essa restituirà il srotolare su cui richiamare la funzione `insFiglio()` per inserire il nuovo elemento.

Dopo aver costruito l'albero, stampare in output i nodi dell'albero, seguendo l'ordine determinato dalla *visita per livelli* indicando, per ogni nodo, la somma dei propri figli (diretti e indiretti).

Esempio

Utilizzando l'input mostrato sopra per creare l'albero in figura, l'output del programma sarà:

```
10 -> 69
15 -> 10
25 -> 19
6 -> 0
4 -> 0
7 -> 0
12 -> 0
```