Esercizio 1

Considerate l'implementazione dell'albero binario ordinato di caratteri vista a lezione. Modificate `cerca` in modo che restituisca l'albero trovato per riferimento, anziché per valore, i.e.: `tree & cerca (tree & t,char elem)`
Definite le seguenti funzioni:

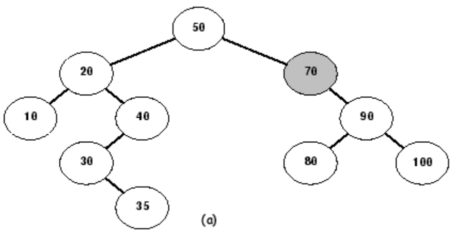
1. `int size(tree t);`
   La funzione deve restituire il numero di elementi contenuti nell'albero.

2. `tree &min(tree &t);`
   La funzione deve restituire un puntatore al nodo con l'elemento minore.

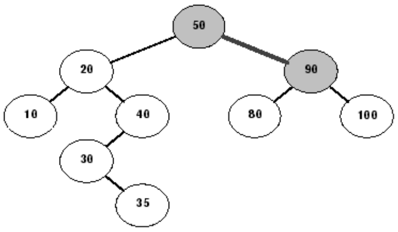
3. `tree &max(tree &t);`
   La funzione deve restituire un puntatore al nodo con l'elemento maggiore.

4. `retval remove(tree& t, char c);`
   La funzione `remove` deve cercare il nodo `z` che contiene il carattere `c`, ed elimininare `z` in modo che l'albero resti ordinato. L'algoritmo da seguire è il seguente:

   - Se `z` è una foglia, è sufficiente sostituirlo con l'albero vuoto (esempio: rimozione dei nodi 10, 35, 80 nella figura sotto).
   - Se `z` ha un solo figlio, è sufficiente sostituire `z` con il figlio di `z` (esempio: la figura (b) sotto mostra l'albero ottenuto cancellando il nodo 70 da (a)).



(a)



(b)

   - Se `z` ha entrambi i figli (esempio: il nodo 20 della figura sotto (a)), allora i casi sono due:

– se il sottoalbero sinistro è più grande del sottoalbero destro, si elimina dall'albero il nodo `z'` con l'elemento maggiore del sottoalbero sinistro: siccome `z'` può avere solo un figlio, basta sostituire `z'` con il figlio di `z'`.
– se il sottoalbero sinistro è più piccolo del sottoalbero destro, si elimina dall'albero il nodo `z'` con l'elemento minore del sottoalbero destro: siccome `z'` può avere solo un figlio, basta sostituire `z'` con il figlio di `z'`.

Nell'esempio: se `z` è il nodo 20, `z'` è il nodo 30; copio il contenuto di `z'` in `z` ed elimino `z'` (vedi passaggi da (a) a (b) e da (b) a (c)).