Algorytmy i Struktury danych

Lista zadań 2 (drzewa)

1. Napisz implementację usuwania węzła z drzewa binarnego wg następującego schematu:
   (a) jeśli usuwany węzeł nie ma dzieci, to go usuwamy a odpowiedni wskaźnik zmieniamy na NULL.
   (b) jeśli ma jedno dziecko, to go usuwamy, a odpowiedni wskaźnik w węźle rodzica zastępujemy wskaźnikiem na to dziecko.
   (c) jeśli ma dwoje dzieci, to nie usuwamy tego węzła, lecz najmniejszy element w jego prawym poddrzewie, a dane i klucz tego elementu wpisujemy do węzła, który miał być usunięty.

2. Napisz operacje dla drzewa BST (`find`, `insert`, `remove`) bez użycia rekurencji.

3. Zakładając, że w każdym węźle drzewa BST jest również wskaźnik na ojca, napisz klasę `iterator` oraz funkcje `iterator begin(node *t)` oraz `iterator end(node *t)`, które pozwolą wypisać wszystkie klucze z drzewa `t` za pomocą instrukcji:

   ```
   for(iterator begin(t); i!=end(t); i++)
       cout<< *i <<endl;
   ```

   Jedyną składową (w cześci prywatnej) powinien być wskaźnik na bieżący węzeł.

4. Jak należy zmienić operacje drzewa BST (`find`, `insert`, `remove`), by prawidłowo uwzględniać pole `parent` - wskaźnik na ojca.

5. (Zadanie trudne) Jak należy zmodyfikować `iterator` drzewa BST, jeśli nie ma ono pola `parent`. Wskazówka: do części prywatnej można dodać stos elementów typu `node*` zawierający węzły, powyżej bieżącego.