Data la seguente struttura BST:

```
struct btree {
    int valore;
    struct btree *leftPtr;
    struct btree *rightPtr;
};

typedef struct btree BST;
```

Completa i seguenti esercizi:

1. **Verifica BST Bilanciato**
   Implementa una funzione `int is_balanced(BST *root)` che restituisce 1 se il BST è bilanciato, e 0 altrimenti. Un BST bilanciato è un albero in cui le altezze dei due sottoalberi figli di ogni nodo differiscono al massimo di uno.
2. **Antenato Comune più Basso**
   Scrivi una funzione `BST* find_lca(BST *root, int n1, int n2)` che trova l'Antenato Comune più Basso (Lowest Common Ancestor, LCA) di due nodi con valori n1 e n2 nel BST. La funzione dovrebbe restituire il nodo che è l'LCA.
3. **K-esimo Elemento più Piccolo**
   Implementa una funzione `int kth_smallest(BST *root, int k)` che trova il k-esimo elemento più piccolo nel BST. Se k è maggiore del numero di nodi nel BST, restituisci -1.
4. **Somma in Intervallo**
   Implementa una funzione `int range_sum(BST *root, int low, int high)` che calcola la somma di tutti i valori dei nodi nel BST che cadono nell'intervallo dato [low, high], estremi inclusi.