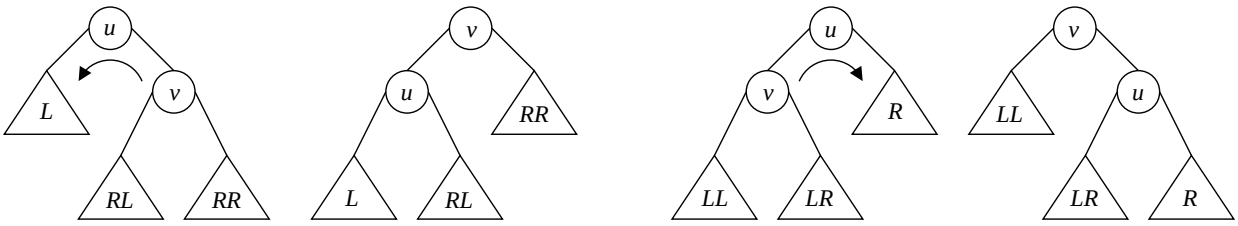
CS29003 Algorithms Laboratory

Assignment No: 6

Last date of submission: 31–August–2017

A node storing a key value and two pointers can be a building block of (1) a doubly linked list where the pointers are interpreted as the previous and next pointers, and (2) a binary tree where the pointers play the role of child pointers. In this assignment, you build a BST (with integer keys) based on user inputs, convert that tree to a doubly linked list, and convert back the list to a balanced tree. In a (size-)balanced tree, the two subtrees at every node differ in size by at most one. Size balance implies height balance (but not conversely).

Part 1: Write the basic functions for binary search trees: (1) Printing in the format shown in the sample output, (2) Insertion, (3) Left rotation, and (4) Right rotation.



(a) Left rotation (b) Right rotation

Also write a function that, given a doubly linked list, first prints the list in the forward direction and then in the backward direction.

Part 2: Write a function *tree2list* that takes a binary search tree as input and adjusts the pointers so that the result is a sorted doubly linked list on the same nodes. A header of the list (pointer to the smallest node) is subsequently returned. The function works as follows.

Let us see what happens in the left subtree of the input BST T. The right subtree can be symmetrically handled. Keep on making left rotations until the left subtree becomes a tree connected only by the left pointers. See the figure on the next page for an illustration. Since the right subtree of Node 40 is not NULL, we do a left rotation there, bringing 747 as the new child of the root. Moreover, 40 gets the subtree rooted at 682 as its new right subtree. Node 747 does not possess a right child, so we move down the tree, and again reach Node 40. Another left rotation at this node brings 682 in the correct position, and so on.

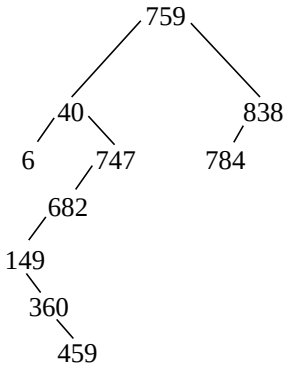
When the two subtrees are completely flattened (see Part (b) of the figure on the next page), all the right pointers in the left subtree and all the left pointers in the right subtree are NULL. Reorient these pointers so that they act as parent pointers. After this, the desired doubly linked list is ready (see Part (c) of the figure).

Part 3: Write a function *list2tree* to convert a sorted doubly linked list to a balanced binary search tree. Let n be the number of nodes in the input list. Locate the position of the root such that its left subtree will contain $\lceil \frac{n-1}{2} \rceil$ nodes, and the right subtree $\lfloor \frac{n-1}{2} \rfloor$ nodes. Recursively build the two subtrees. See Part (d) of the figure on the next page.

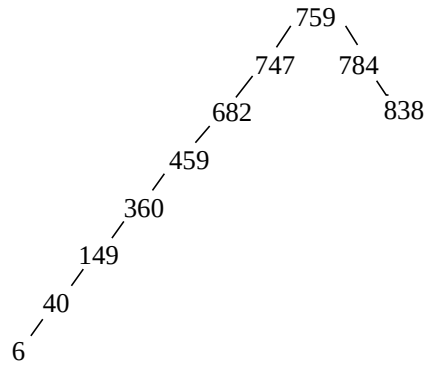
The *main*() function

- Read n from the user. Initialize T to an empty BST.
- The user then supplies n keys. Insert them in T. The resulting tree has $\leqslant n$ nodes. Print T.
- Call *tree2list* to get a sorted doubly linked list L of the key values stored in the tree. The function, after flattening the two trees, prints the tree. It then makes the pointer rearrangements to convert T to L. Print L using the list-printing function.
- Call *list2tree* to convert L to a balanced tree T. Print T.

Submit a single C/C++ source file. Do not use global/static variables.



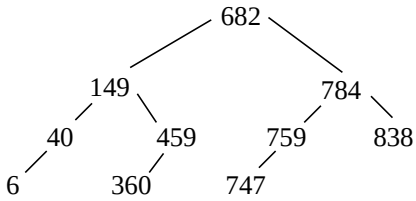
(a)



(b)

6 — 40 — 149 — 360 — 459 — 682 — 747 — 759 — 784 — 838

(c)



(d)

Sample output

```
    10
    759  40 747 682 149 360   6 838 784 459
+++ Initial tree
              6
          40
                    149
                        360
                            459
                682
            747
    759
            784
        838
+++ Flattened tree
                                    6
                                40
                            149
                        360
                    459
                682
            747
        759
            784
                838
+++ Linked list
      6  40 149 360 459 682 747 759 784 838
    838 784 759 747 682 459 360 149  40   6
+++ Balanced tree
                    6
                40
            149
                    360
                459
    682
                    747
                759
            784
                838
```