National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B. Tech.(CSE)-Monsoon 2024
CS2091E Data Structures Laboratory
Assignment 2

Submission deadline (on or before): 21/08/2024, 11:59 PM

Part B

3. You are given a *Parenthesis representation* (*Refer Part A of Assignment 1*) of a non-empty AVL tree T with unique keys n, where $n \in [1, 10^7]$.
   Implement a menu-driven program that performs the following operations on the AVL tree T.

   Operations

   (a) $AVL\_SeqInsert(T, [n1, n2, ..., nk])$: Insert a sequence of keys $[n1, n2, ..., nk]$ into the AVL tree T. Perform necessary rotations to maintain AVL properties after each insertion. After all insertions, print the paranthesis representation of resulting AVL tree.

   (b) $AVL\_RangeDelete(T, n1, n2)$: Delete all keys in the AVL tree T that are within the range $[n1, n2]$ (inclusive). Perform necessary rotations to maintain AVL properties after each deletion. After performing all deletions, print the total number of nodes deleted and preorder traversal of the tree T after all deletions.

   (c) $AVL\_SuccessorPath(T, n)$: For a given key n, find its inorder successor in T. Print the path from the root to the inorder successor (including the successor itself). If n does not exist or has no successor, return the height of T.

   (d) $AVL\_SubtreeSum(T, n)$: For a given key n, print the sum of all keys in the subtree rooted at n and the parenthesis representation of the corresponding subtree rooted at n. If n is not present in T, print -1.

   (e) $AVL\_FindClosest(T, n)$: Given a key n, find and print the *closest key* in T (the key with the minimum absolute difference from n). If there are multiple such keys present, print the smallest among them. If n is not found in T, or the closest key of n is not present (T has only one node and it is n), then print -1.
   Note: *closest key* of n cannot be itself.

   Input Format

   - First line of the input contains a space separated Parenthesis representation of a non-empty AVL tree T with key values, $n \in [1, 10^7]$.
   - Each subsequent line contains a character from {'a', 'b', 'c', 'd', 'e', 'g'} followed by zero or more positive integers n.
   - Character `'a'` is followed by a space-separated sequence of positive integers $[n1, n2, \ldots, nk]$. Perform $AVL\_SeqInsert(T, [n1, n2, ..., nk])$ operation.
   - Character `'b'` is followed by two positive integers, separated by a space. Perform $AVL\_RangeDelete(T, n1, n2)$ operation.
   - Character `'c'` is followed by a positive integer n, separated by a space. Perform $AVL\_SuccessorPath(T, n)$ operation.
   - Character `'d'` is followed by a positive integer n, separated by a space. Perform $AVL\_SubtreeSum(T, n)$ operation.

- Character `'e'` is followed by a positive integer n, separated by a space. Perform $AVL\_FindClosest(T, n)$ operation.
- Character `'g'` is to terminate the sequence of operations.

Output Format

- The output (if any) of each command should be printed on a separate line. However, no output is printed for `'g'`.
- For option `'a'`: Print a space-separated parenthesis representation of the resulting AVL tree after all the insertions.
- For option `'b'`: Print a space-separated sequence of integers representing the total number of nodes deleted followed by the preorder traversal of the tree T after all the deletions. Print `-1` if there are no nodes within the range in T to be deleted.
- For option `'c'`: Print the sequence of keys of nodes in the path from the root to the inorder successor of n, separated by a space. If n does not exist or has no successor in T, print the height of T.
- For option `'d'`: Print the sum of all keys in the subtree rooted at n followed by the parenthesis representation of T, separated by a space. If n is not present, print `-1`.
- For option `'e'`: Print the *closest key* of n in T. Print `-1` if n or its *closest key* is not present in T.

Sample test case 1

Input:

```
23 ( 10 ( 9 ( 7 ) ( ) ) ( 15 ( 11 ) ( 17 ) ) ) ( 45 ( 30 ) ( 67 ) )
a 4 14 21
b 30 67
c 11
e 32
d 10
c 23
d 55
e 10
g
```

Output:

```
15 ( 10 ( 7 ( 4 ) ( 9 ) ) ( 11 ( ) ( 14 ) ) ) ( 23 ( 17 ( ) ( 21 ) ) ( 45 ( 30 ) ( 67 ) ) )
3 15 10 7 4 9 11 14 21 17 23
15 10 11 14
-1
55 10 ( 7 ( 4 ) ( 9 ) ) ( 11 ( ) ( 14 ) )
3
-1
9
```