Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Programming and Data Structures
Ph.D. Coursework: First year, First Semester (Session: 2024-25)
Assignment #08

Full Marks: 200 | Instructor: Dr. Laltu Sardar
Clarification Deadline: **2024-Nov-28** | Submission Deadline: **2024-Dec-01**

Instructions:

1. Keep all functions in "`avlTree.h`" file, and create a separate test file named "`avlTree_test.c`".
2. The program should be as fault-tolerant as possible, handling potential input errors gracefully.
3. The test file should include a menu, and input matrices must be provided via files only.
4. You must use previously defined BST-related functions.

Problem #0801: AVL Tree

This assignment focuses on implementing AVL tree operations in C to manage a self-balancing binary search tree efficiently. Implement each AVL tree operation in C as described below.

1. `Node* insertAVL(Node *root, int key);` Insertion into an AVL tree is similar to a binary search tree, with additional rotations to maintain balance after each insertion.
   - `root`: Pointer to the root node of the AVL tree.
   - `key`: Integer value to insert.
   - Returns a pointer to the new root node after insertion, adjusting for balance if needed.
2. `Node* deleteAVL(Node *root, int key);` Deletion in an AVL tree removes a node while maintaining balance through rotations.
   - `root`: Pointer to the root node of the AVL tree.
   - `key`: Integer value to delete.
   - Returns a pointer to the new root node after deletion, with AVL balance maintained.
3. `Node* searchAVL(Node *root, int key);` Searching for a key in an AVL tree is a binary search operation, with complexity $O(\log n)$ due to the balanced property.
   - `root`: Pointer to the root node of the AVL tree.
   - `key`: Integer value to search for.
   - Returns a pointer to the node containing the key if found, otherwise returns NULL.
4. `void inOrderTraversal(Node *root);` Implement for traversing an AVL tree. Consider in-order traversal only.
   - `root`: Pointer to the root node of the AVL tree.
   - Output: Prints the elements of the AVL tree in the specified order.

Testing

Write a test function containing menu. The menu should take input from files only. The format of the input is as follows.

- 1st line contains number of test cases.
- 2nd line onward, each line is of the form `op key` where `op` = `i`, `d`, or `s`, indicating insert, delete and search respectively.
- After every insert and delete operations, display the tree (in-order).

Example input file:

```
7
i 10
i 20
i 30
s 20
d 10
i 25
d 30
```