CS29003 Algorithms Laboratory
Assignment 6: Tree

General instruction to be followed strictly

1. Do not use any global or static variable unless you are explicitly instructed so.
2. Do not use Standard Template Library (STL) of C++.
3. Use proper indentation in your code and comment.
4. **Name your file as `<roll_no>_<assignment_no>`. For example, if your roll number is 14CS10001 and you are submitting assignment 3, then name your file as `14CS10001_3.c` or `14CS10001_3.cpp` as applicable.**
5. **Write your name, roll number, and assignment number at the beginning of your program.**
6. Make your program as efficient as possible. Follow best practices of programming.
7. Submit your program on Moodle before deadline. Submissions by email or any other means will NOT be considered for evaluation.

Create an interactive interface for the user to (i) insert (ii) delete a key from a "normal" binary search tree, and (iii) make the tree balanced. By "normal," I mean that the binary search tree is not self-balancing like AVL tree, etc. To delete a node which has two children, "use/exchange" its in-order predecessor instead of in-order successor. After every tree operation, print the pre-order traversal of the binary search tree.

For making the binary search tree balanced, implement the following algorithm. Initialize an empty AVL tree. Iteratively delete the root of the binary search tree and insert it in the AVL tree until the binary search tree becomes empty.

The approach above takes $\mathcal{O}(n \log n)$ time to make a binary search tree with n nodes balanced. Now design an $\mathcal{O}(n)$ time algorithm for making a binary search tree balanced. Hint: use divide-and-conquer technique.

Assume the keys are all distinct.

For this assignment, you are allowed to store parent pointer also in the tree node if you want.

Sample Output

```
1. insert
2. delete
3. make tree balanced (AVL tree based method).
4. make tree balanced faster
5. exit

1
Write key to be inserted: 10
Pre-order traversal: 10
```

```
1
Write key to be inserted: 5
Pre-order traversal: 10, 5
1
Write key to be inserted: 20
Pre-order traversal: 10, 5, 20
1
Write key to be inserted: 7
Pre-order traversal: 10, 5, 7, 20
1
Write key to be inserted: 1
Pre-order traversal: 10, 5, 1, 7, 20
1
Write key to be inserted: 9
Pre-order traversal: 10, 5, 1, 7, 9, 20
1
Write key to be inserted: 30
Pre-order traversal: 10, 5, 1, 7, 9, 20, 30
2
Write key to be deleted: 10
Pre-order traversal: 9, 5, 1, 7, 20, 30
2
Write key to be deleted: 9
Pre-order traversal: 7, 5, 1, 20, 30
2
Write key to be deleted: 7
Pre-order traversal: 5, 1, 20, 30
1
Write key to be inserted: 40
Pre-order traversal: 5, 1, 20, 30, 40
3
Pre-order traversal of tree after balancing with AVL tree based method: 5, 1, 30, 20, 40
1
Write key to be inserted: 50
Pre-order traversal: 5, 1, 30, 20, 40, 50
4
Pre-order traversal of tree after balancing with faster method: 20, 1, 5, 40, 30, 50
5
Program exits
```