Lab 8: AVL tree

In this lab, we wil implement AVL tree ADT, In each node of the AVL tree, the difference between the height of the left subtree and the height of the right subtree is at most 1. We will implement *insert* function, which calls additional functions for single rotation and double rotation.

1. Input
Obtain a list of numbers from the given input file, and execute an insertion operation for each number in order. At each iteration of insertion, print the AVL tree inorder traversal. An example input file is shown below.

Input.txt

```
7 5 3 10 23 4 20 21 22 23 24 25
```

2. AVL tree ADT

(1) Data Specification for the objects

```
struct AVLNode;
typedef struct AVLNode *Position;
typedef struct AVLNode *AVLTree;

struct AVLNode
{
        ElementType Element;
        AVLTree Left;
        AVLTree Right;
        int Height;
}
```

(2) Function specification

- Position SingleRotateWithLeft(Position node)
- Position SingleRotateWithRight(Position node)
- Position DoubleRotateWithLeft(Position node)
- Position DoubleRotateWithRight(Position node)
- AVLTree Insert(ElementType X, AVLTree T)
- printInorder(AVLTree T)

3. Program description

- name : p8.c
- input : a list of operations in a file (an input file name is given as a command line argument. See the example in "1. input" on the first page)
- output : the corresponding result in the standard output

Submit to the course website (https://portal.hanyang.ac.kr) your source code and a written report. Your report should include the description of your own implementation.