**DSA Lab13**

**Objectives:**

In this lab, students will practice:

1. Insert operation on AVL Trees
2. Delete operation on AVL Trees

**Question 1: For this lab, use your BST code implemented in the previous lab.In this lab, you have to create a height-balanced tree class named “AVL”. Inherit the BST class publicly in your new “AVL” class. You can add “height” variable in your existing TNode struct implementation.**

Implement the following methods for AVL class:

* 1. A default Constructor which calls the default constructor of base class (BST class).
  2. Override the insert method of base class (BST class) in your AVL class, so that the AVL tree remains height-balanced after insertion of a new node.
  3. Override the delete method of base class (BST class) in your AVL class, so that the AVL tree remains height-balanced after deletion of a node.
  4. A function “height” which returns the height of the tree. int height()const
  5. A function “search” which returns a pointer to the value of the node containing the required key b

**Question 2: Now run the following main program.**

int main()

{

AVL<int, int> tree;

for (int i = 1; i <= 100; i++)

tree.insert(i, i);

for (int i = -1; i >= -100; i--)

tree.insert(i, i);

for (int i = 150; i > 100; i--)

tree.insert(i, i);

for (int i = -150; i < -100; i++)

tree.insert(i, i);

for (int i = 150; i > 100; i--)

tree.delete(i);

tree.inorderPrintKeys();

cout << endl << endl;

cout <<"Tree Height: "<< tree.height() << endl;

int \*val = tree.search(-100);

if (val != nullptr)

{

cout <<"Key= -100 found"<< endl;

}

val = tree.search(-151);

if (val == nullptr)

{

cout <<"Key= -151 not found"<< endl;

}

system("pause");

}