**Data Structures and Algorithms**

Logo, company name

Description automatically generated

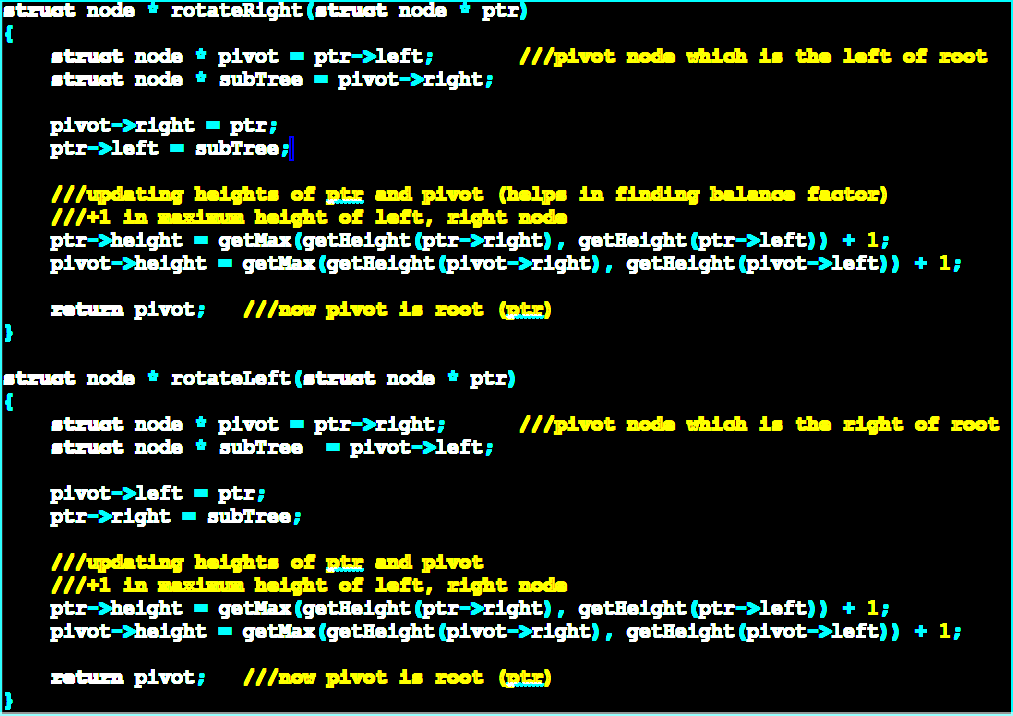
**Lab report: 11**

|  |  |
| --- | --- |
| **Name:** | **Ali Salman** |
| **Reg no:** | **FA22-BCE-005** |
| **Class:** | **BCE-3A** |
| **Lab Instructor:** | **Dr. Ali Mustafa** |

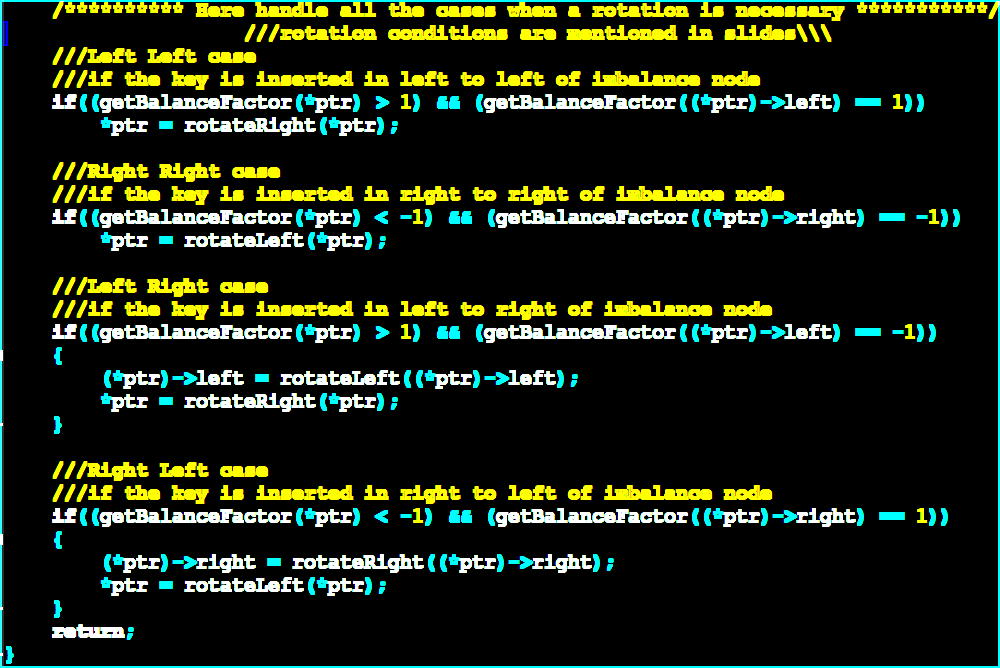
**Lab 11: AVL Trees Implementation**

**In-Lab Tasks:**

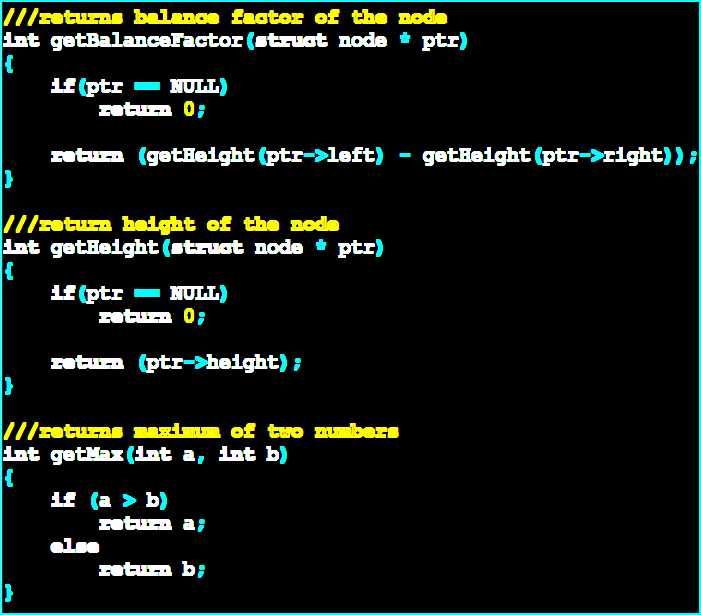
**AVL Rotation Functions**

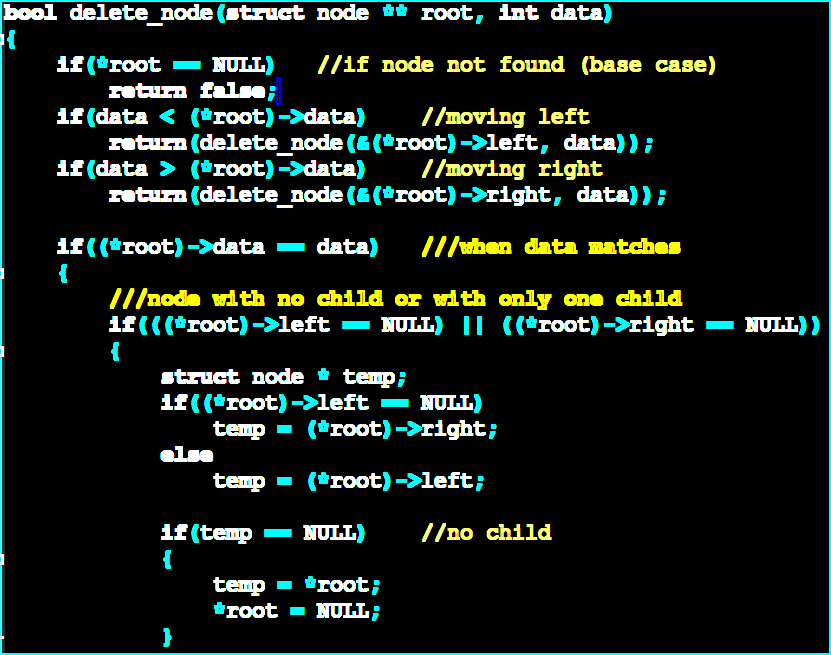


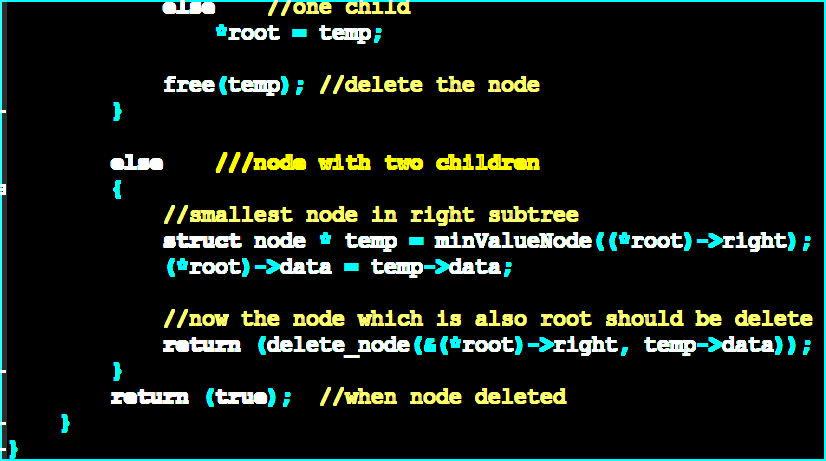
**Rotation cases for AVL tree:**



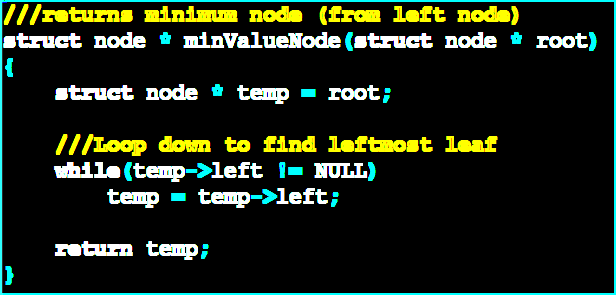
**Helper functions**

**Post Lab Task:**





**Helper function:**



**Critical Analysis:**

The AVL Trees Implementation lab offered us a deep dive into the intricate world of AVL trees, focusing on their properties and the crucial aspect of maintaining balance.

During the in-lab session, students were handed a skeleton code facilitating the creation of a binary search tree with 10 initial nodes, where functions for node insertion and in-order tree traversal were pre-implemented. In order to check the balancing factor, we need a helper function to check the balancing factor of each node. The balancing factor is checked after every insertion. The crux of the task lay in modifying the insert function to incorporate AVL insertion, a task guided by a programming example from a specified source. Further, we were tasked with implementing the rotateLeft() and rotateRight() functions, along with tackling the four cases of balancing the tree.

The post-lab tasks extended the complexity, demanding the addition of delete node functionality to the BST. This lab not only delved into the theoretical aspects of AVL trees but also tested students' programming prowess by challenging them to apply these concepts in a hands-on manner.