**TAD Data Structures**

|  |  |  |
| --- | --- | --- |
| **TAD Binary Search Tree** | | |
| BSTree= {Node<T> root} | | |
| {inv: right subtree < root and the left subtree root} | | |
| Primitive Operations | | |
| * BSTree: * Insert: * Search: * Delete: | Key  Root, Key  Key | →Binary Search Tree  →Binary Search Tree  →Key  →Root |

|  |
| --- |
| **BSTree():** |
| “Create a new empty binary search tree” |
| {pre: TRUE} |
| {post: new empty binary search tree} |

|  |
| --- |
| **Insert(K key)** |
| “Add a new node to the Binary Search Tree” |
| {pre: Binary Search Tree has to be created } |
| {post: new node added to the Binary Search Tree} |

|  |
| --- |
| **Search(Node<T> root, T key):** |
| “Search if the node is inside of the Binary Search Tree” |
| {pre: Binary Search Tree has to be created and Binary Serach Tree != empty} |
| {post: new node added to the Binary Search Tree} |

|  |
| --- |
| **Delete():** |
| “Delete a particular node in the Binary Search Tree according to the key” |
| {pre: Binary Search has to be created and k is in the Binary Search Tree} |
| {post: key doesn’t exist in the Binary Search Tree and slot is null} |

|  |  |  |
| --- | --- | --- |
| **TAD AVL Tree** | | |
| AVLTree= {Node<T> root} | | |
| {inv: right subtree < root and the left subtree root} | | |
| Primitive Operations | | |
| * BSTree: * Add: * Insert: * Search: * Delete: | Key  Root, Key  Key | →Binary Search Tree  →Binary Search Tree  →Key  →Root |

|  |
| --- |
| **AVLTree():** |
| “Create a new empty AVL tree” |
| {pre: TRUE} |
| {post: new empty AVL tree} |

|  |
| --- |
| **Add(T data):** |
| “Add a new element in the AVL Tree ” |
| {pre: AVL Tree has to be created} |
| {post: new element in the AVL Tree} |

|  |
| --- |
| **Insert(T data):** |
| “Add a new element in the AVL Tree ” |
| {pre: AVL Tree has to be created} |
| {post: new element in the AVL Tree} |

|  |
| --- |
| **Search(Node x):** |
| “Search if the node is inside of the AVL Tree” |
| {pre: AVL Tree has to be created and AVL Tree != empty} |
| {post: value that was stored in the node identified with the input key} |

|  |
| --- |
| **DeleteNode():** |
| “Delete a particular node in the Binary Search Tree according to the key” |
| {pre: AVL Tree has to be created and k is in the AVL Tree} |
| {post: key doesn’t exist in the AVL Tree and slot is null} |

|  |
| --- |
| **RotateLeft(Node x):** |
| “Rotate nodes to the right if the roll of node x is greater than 1” |
| {pre: node x with the factor balance greather than 1} |
| {post: new factor balance in the node x} |

|  |
| --- |
| **RotateRigth(Node x):** |
| “Rotate nodes to the left if the roll of node x is less than -1” |
| {pre: node x with the factor balance less than -1} |
| {post: new factor balance in the node x} |