Akshay S Bharadwaj
1BM18CS011

ADS Lab-4 Writeup (AVL Trees)

```
class node {    // AVL Tree Node class
  public:
    int data;
    node * left;
    node * right;
    int height;
};
```

```
int height (node *p) {
    if (p == NULL)
        return 0;
    return p -> height;
} // Height is returned
```

```
node * getnode(int data) {  // Function to create node
    node *p = new node();
    p -> data = data;
    p -> left = NULL;
    p -> right = NULL;
    p -> height = 1;
    return p;
}
```

```
node *rotateright (node *b) {  // Rotate right function
    node *a = b -> left;
    node *t = b -> right;
    a -> right = b;
    b -> left = t;
    b -> height = max (height (b -> left),
                       height (b -> right)) + 1;
    a -> height = max (height (a -> left),
                       height (a -> right)) + 1;
    return a;
}
```

Akshay S Bharadwaj
1BM18CS011

```
node *rotateleft (node *a){// Rotate left function
    node *b = a->right;
    node *t = a->left;
    b->left = a;
    a->right = t;
a->height = max(height(a->right),
                height(a->left)) + 1;
b->height = max(height(b->left),
                height(b->right)) + 1;
    return b;
}
int balance (node *p) { // Balance factor of node
    if (p == NULL)
        return 0;
    return height(p->left) - height(p->right);
}
node *insertion (node *root, int data) { // Insert function
    if (root == NULL) {
        return getnode(data);
    if (data < root->data)
        root->left = insertion(root->left, data);
    else if (data > root->data)
        root->right = insertion(root->right, data);
    else
        return root;
root->height = max(height(root->left),
                   height(root->right)) + 1;
```

Akshay S Bharadwaj
1BM18CS011

```
int bl = balance(root);
  if (bl > 1 && data < root->left->data)
        return rotateright(root);
  else if (bl < -1 && data > root->right->data)
        return rotateleft(root);
  if (bl > 1 && data > root->left->data) {
        root->left = rotate left(root->left);
            return rotate right(root);
  }
  if (bl < -1 && data < root->left->data){
        root->right = rotateright(root->right);
        return rotate left(root);
  }
  return root;
}
node *deletion(node* root, int item) { // Deletion fun
  if (root == NULL)
        return root;
  if (item < root->data)
      root->left = deletion(root->left, item);
else if (item > root->data)
      root->right = deletion(root->right, item);
  else {
    if (root->left == NULL || root->right == NULL) {
        node *t = root->left ? root->left : root->right;
```

Akshay S Bharadwaj
1BM18CS011

```
        if (t == NULL) {
            t = root;
            root = NULL;
        }
        else
            *root = *temp;
        free(temp);
    }
    else {
        node *t = minvalue(root -> right);
        root -> data = t -> data;
        root -> right = deletion(root -> right, t -> data);
    }
}
if (root == NULL)
    return root;
int bl = balance(root);
root -> height = max(height(root -> left), height(root -> right)) + 1;
return
if (bl > 1 && balance(root -> left) >= 0)
    return rotateright(root);
if (bl < -1 && balance(root -> right) <= 0)
    return rotateleft(root);
if (bl > 1 && balance(root -> left) < 0) {
    root -> left = rotateleft(root -> left);
    return rotateright(root);
}
if (bl < -1 && balance(root -> right) > 0) {
    root -> right = rotateright(root -> right);
    return rotateleft(root);
}
return root;
}
```

AS