SOURCE CODE

```
#include<iostream>
using namespace std;

class node{
public:
    int data;
    node* left;
    node* right;
    node(int val){
        data=val;
        left=NULL;
        right=NULL;
    }
};
// search the element in Avl tree
int searchEle(node* root,int ele){
    if(root==NULL){
        return -1;
    }
    if(root->data==ele){
        return 1;
    }
    int left=searchEle(root->left,ele);
    if(left!=-1){
        return left;
    }
    return searchEle(root->right,ele);
}
```

```
//  Calculate height
int height(node* root){
    if(root==NULL){
        return -1;
    }
    int lh=height(root->left);
    int rh=height(root->right);
    return max(lh,rh)+1;
}
```

```
// Get balance factor
int get_balfactor(node* root){
    if(root==NULL){
        return -1;
    }
    int lh=height(root->left)+1;
    int rh=height(root->right)+1;
    return (lh-rh);
}
// left rotate
```

```
node* leftRotate(node* &root){
    node* temp1=root->right;
    node* temp2=temp1->left;
    temp1->left=root;
    root->right=temp2;
    return temp1;
}
//rightRotate
node* rightRotate(node* &root){
    node* temp1=root->left;
    node* temp2=temp1->right;
    temp1->right=root;
    root->left=temp2;
    return temp1;
}
```

```
// Insertion of a node in AVL tree
node* Insertion(node* &root,int val){
    if(root==NULL){
        return new node(val);
    }
    if(root->data>val){
        root->left=Insertion(root->left,val);
    }
    else if(root->data<val){
        root->right=Insertion(root->right,val);
    }
    else{
        cout<<"Duplicate is not Allowed"<<endl;
        return root;
    }
    int balf=get_balfactor(root);
    if(balf>1 and val < root->left->data){ //ll
        return rightRotate(root);
    }
    if(balf>1 and val > root->left->data){ //lr
        root->left=leftRotate(root->left);
        return rightRotate(root);
    }
    if(balf<-1 and val > root->right->data){ //rr
        return leftRotate(root);
    }
    if(balf<-1 and val < root->right->data){ //rl
        root->right=rightRotate(root->right);
        return leftRotate(root);
    }
    return root;
}
```

```cpp
//print Inorder
void printInorder(node* root){
    if(root==NULL){
        return;
    }
    printInorder(root->left);
    cout<<root->data<<" ";
    printInorder(root->right);
}
```

```cpp
int main(){
node *root=NULL;
int n,val;
cout<<"Enter the Number of element you want to insert: ";
cin>>n;
cout<<"Enter the element: ";
cin>>val;
root=Insertion(root,val);
for(int i=0;i<n-1;i++){
    cout<<"Enter the element: ";
    cin>>val;
    root=Insertion(root,val);
}
printInorder(root);
cout<<endl;

return 0;
}
```

OUTPUT

```
PS C:\Users\anil kumar\Documents\anil\.vscode\DataSructure_in_nsut> cd "c:\Users\anil ku
cture_in_nsut\" ; if ($?) { g++ -std=c++17 18_AvlTree_Insertion.cpp -o 18_AvlTree_Inser
ertion }
Enter the Number of element you want to insert: 9
Enter the element: 5
Enter the element: 3
Enter the element: 4
Enter the element: 7
Enter the element: 6
Enter the element: 8
Enter the element: 9
Enter the element: 2
Enter the element: 1
1 2 3 4 5 6 7 8 9
PS C:\Users\anil kumar\Documents\anil\.vscode\DataSructure_in_nsut>
```