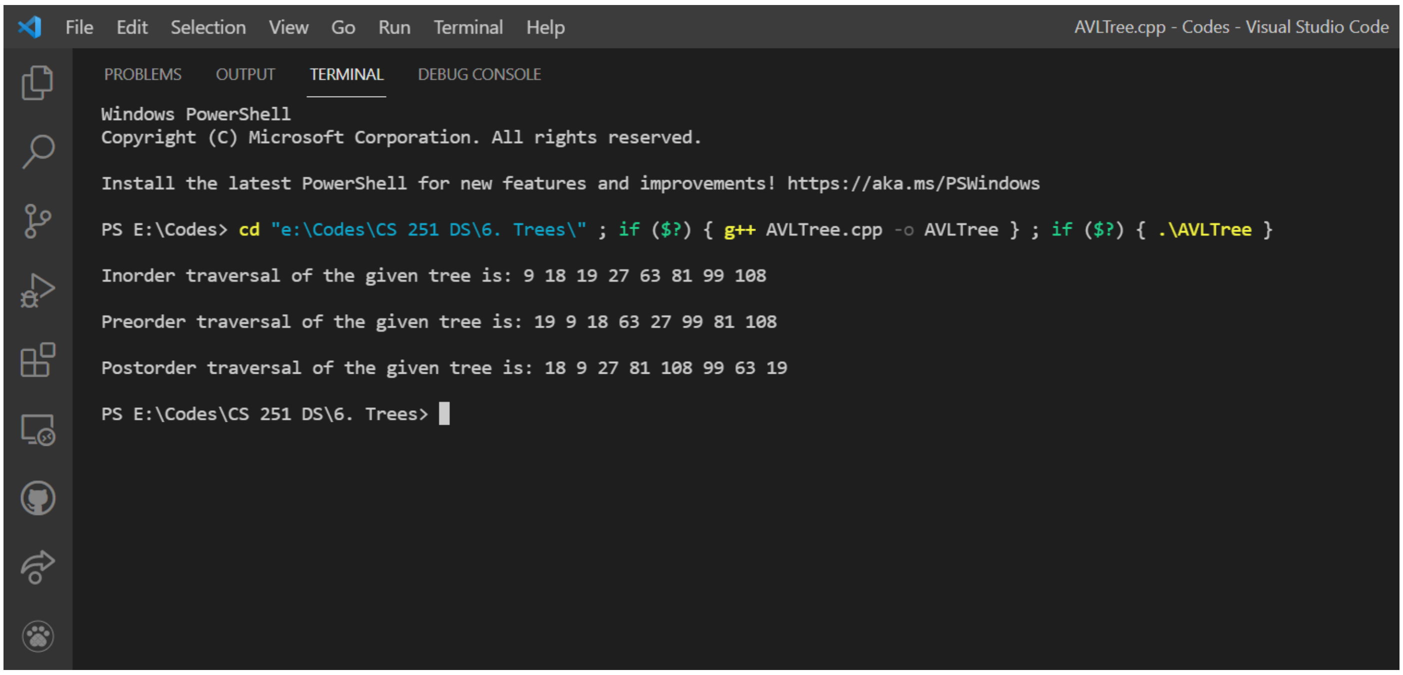
Insertion in Height Balanced AVL Tree

```cpp
// Aneesh Panchal
// 2K20/MC/21

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

class Node{
    public:
    int key;
    Node *left;
    Node *right;
    int height;
};

int height(Node *N){
    if (N == NULL)
        return 0;
    return N->height;
}

int max(int a, int b){
    return (a > b)? a : b;
}

Node* newNode(int key){
    Node* node = new Node();
    node->key = key;
    node->left = NULL;
    node->right = NULL;
    node->height = 1;
    return(node);
}

Node *rightRotate(Node *y){
    Node *x = y->left;
    Node *T2 = x->right;

    x->right = y;
    y->left = T2;

    y->height = max(height(y->left),height(y->right)) + 1;
    x->height = max(height(x->left),height(x->right)) + 1;

    return x;
}

Node *leftRotate(Node *x){
    Node *y = x->right;
    Node *T2 = y->left;
    y->left = x;
    x->right = T2;

    x->height = max(height(x->left),height(x->right)) + 1;
    y->height = max(height(y->left),height(y->right)) + 1;

    return y;
}

int getBalance(Node *N){
    if (N == NULL)
        return 0;
    return height(N->left) - height(N->right);
}

Node* insert(Node* node, int key){
    if (node == NULL)
        return(newNode(key));

    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    else
        return node;

    node->height = 1 + max(height(node->left),height(node->right));

    int balance = getBalance(node);

    if (balance > 1 && key < node->left->key)
        return rightRotate(node);

    if (balance < -1 && key > node->right->key)
        return leftRotate(node);

    if (balance > 1 && key > node->left->key){
        node->left = leftRotate(node->left);
        return rightRotate(node);
    }

    if (balance < -1 && key < node->right->key){
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }

    return node;
}

void inorder(Node *Root){
    stack<Node*> nodestack;
    Node *curr = Root;

    while (curr != NULL || nodestack.empty() == false){
```

```cpp
        while (curr !=  NULL){
            nodestack.push(curr);
            curr = curr->left;
        }

        curr = nodestack.top();
        nodestack.pop();

        cout << curr->key << " ";
        curr = curr->right;
    }
}

void preorder(Node *Root){
    if (Root == NULL)
        return;

    stack<Node*> nodeStack;
    nodeStack.push(Root);

    while (nodeStack.empty() == false) {
        Node* node = nodeStack.top();
        printf("%d ", node->key);
        nodeStack.pop();

        if (node->right!=NULL)
            nodeStack.push(node->right);
        if (node->left!=NULL)
            nodeStack.push(node->left);
    }
}

void postorder(Node *Root){
    if (Root == NULL)
        return;

    stack<Node*> nodeStack;
    nodeStack.push(Root);
    stack<int> out;

    while (!nodeStack.empty()){
        Node* curr = nodeStack.top();
        nodeStack.pop();
        out.push(curr->key);

        if (curr->left!=NULL)
            nodeStack.push(curr->left);
        if (curr->right!=NULL)
            nodeStack.push(curr->right);
    }

    while (!out.empty()){
        cout << out.top() << " ";
        out.pop();
    }
}

int main(){
    Node *root = NULL;
    root = insert(root, 63);
    root = insert(root, 9);
    root = insert(root, 19);
    root = insert(root, 27);
    root = insert(root, 18);
    root = insert(root, 108);
    root = insert(root, 99);
    root = insert(root, 81);
    cout<<endl<<"Inorder traversal of the given tree is: ";
    inorder(root);

    cout<<endl<<endl<<"Preorder traversal of the given tree is: ";
    preorder(root);

    cout<<endl<<endl<<"Postorder traversal of the given tree is: ";
    postorder(root);
    cout<<endl<<endl;
    return 0;
}
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\CS 251 DS\6. Trees\" ; if ($?) { g++ AVLTree.cpp -o AVLTree } ; if ($?) { .\AVLTree }

Inorder traversal of the given tree is: 9 18 19 27 63 81 99 108

Preorder traversal of the given tree is: 19 9 18 63 27 99 81 108

Postorder traversal of the given tree is: 18 9 27 81 108 99 63 19

PS E:\Codes\CS 251 DS\6. Trees> 
```