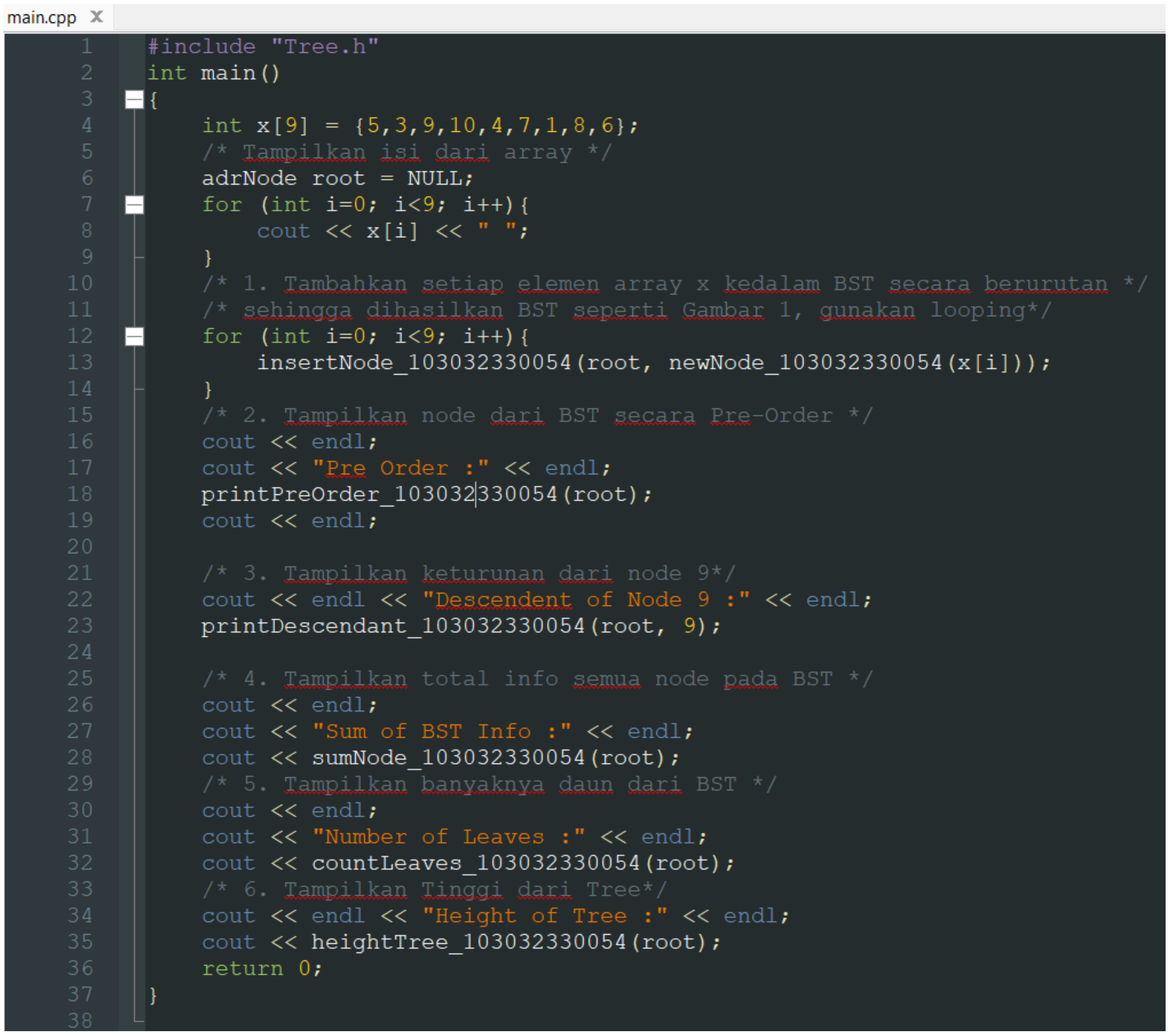
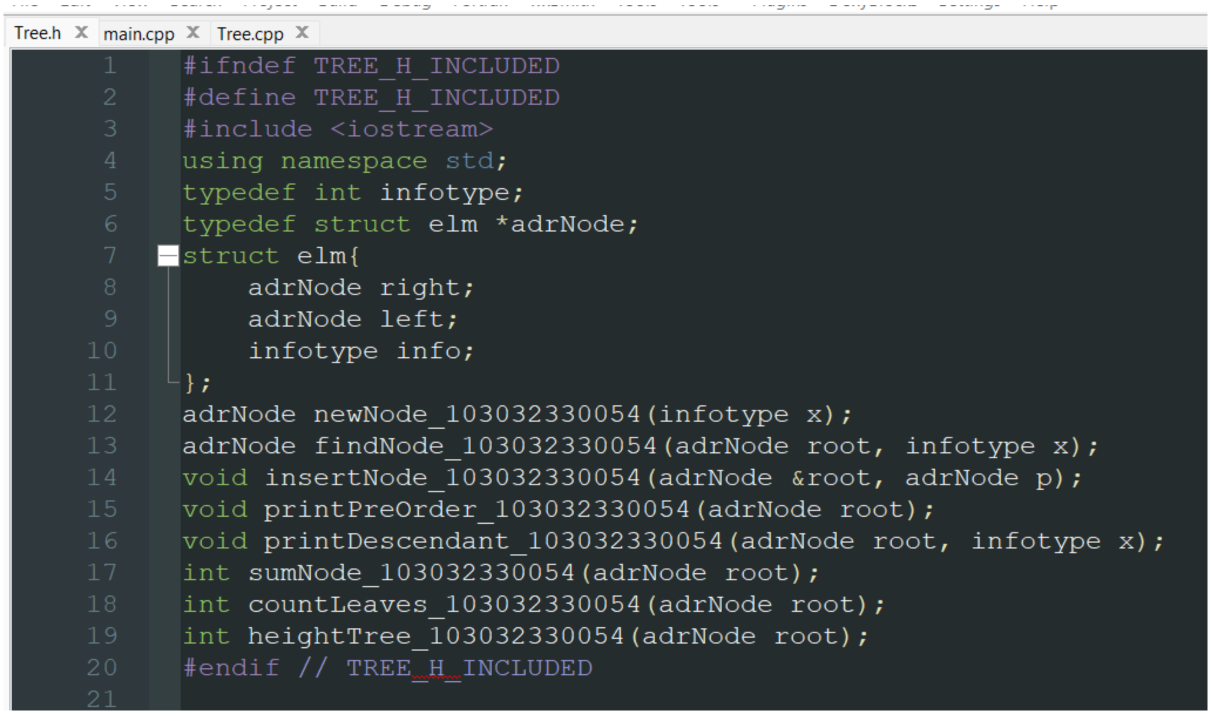
Main.cpp



main.cpp

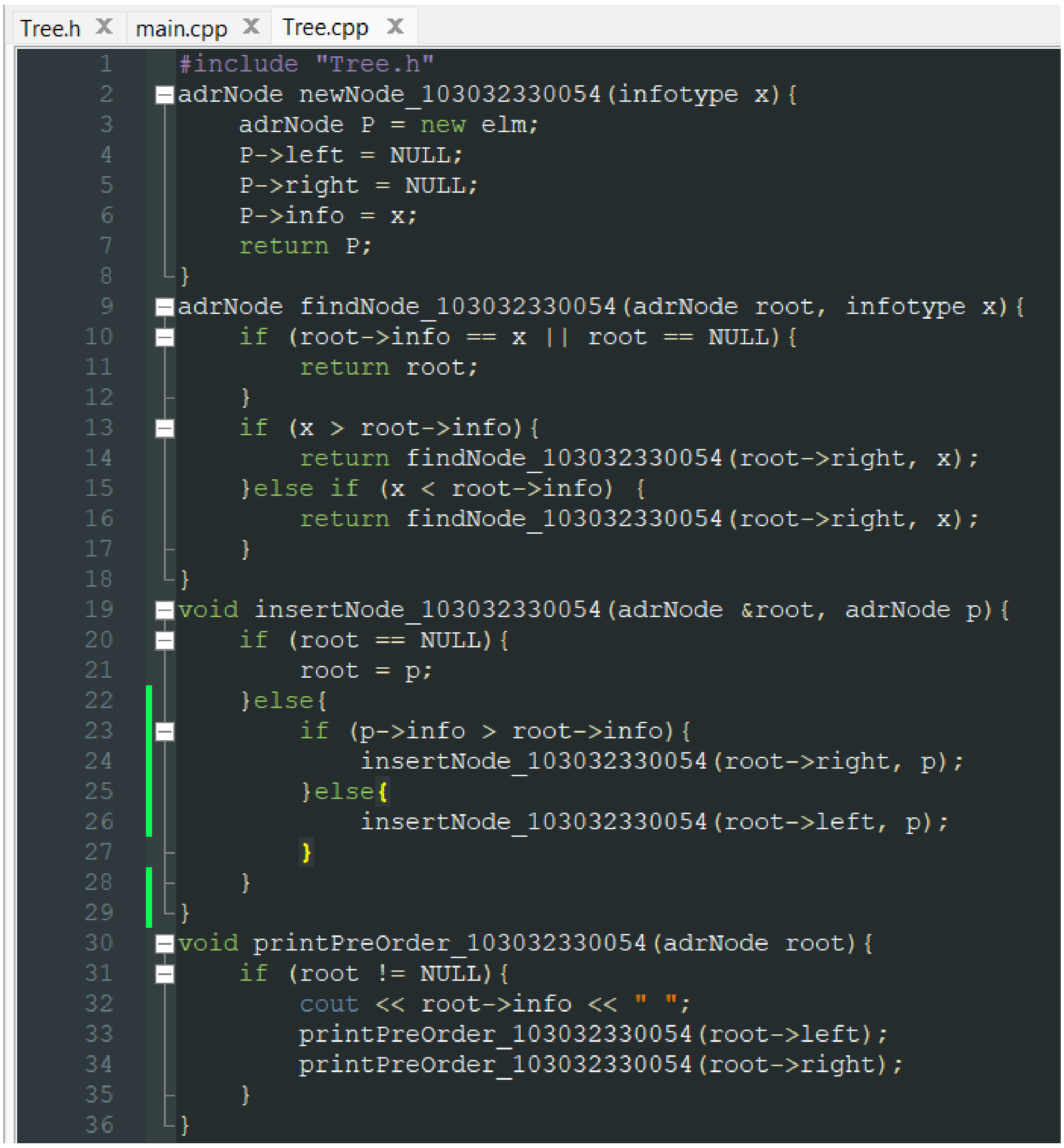
```cpp
#include "Tree.h"
int main()
{
    int x[9] = {5,3,9,10,4,7,1,8,6};
    /* Tampilkan isi dari array */
    adrNode root = NULL;
    for (int i=0; i<9; i++){
        cout << x[i] << " ";
    }
    /* 1. Tambahkan setiap elemen array x kedalam BST secara berurutan */
    /* sehingga dihasilkan BST seperti Gambar 1, gunakan looping*/
    for (int i=0; i<9; i++){
        insertNode_103032330054(root, newNode_103032330054(x[i]));
    }
    /* 2. Tampilkan node dari BST secara Pre-Order */
    cout << endl;
    cout << "Pre Order :" << endl;
    printPreOrder_103032330054(root);
    cout << endl;

    /* 3. Tampilkan keturunan dari node 9*/
    cout << endl << "Descendent of Node 9 :" << endl;
    printDescendant_103032330054(root, 9);

    /* 4. Tampilkan total info semua node pada BST */
    cout << endl;
    cout << "Sum of BST Info :" << endl;
    cout << sumNode_103032330054(root);
    /* 5. Tampilkan banyaknya daun dari BST */
    cout << endl;
    cout << "Number of Leaves :" << endl;
    cout << countLeaves_103032330054(root);
    /* 6. Tampilkan Tinggi dari Tree*/
    cout << endl << "Height of Tree :" << endl;
    cout << heightTree_103032330054(root);
    return 0;
}
```

Tree.h

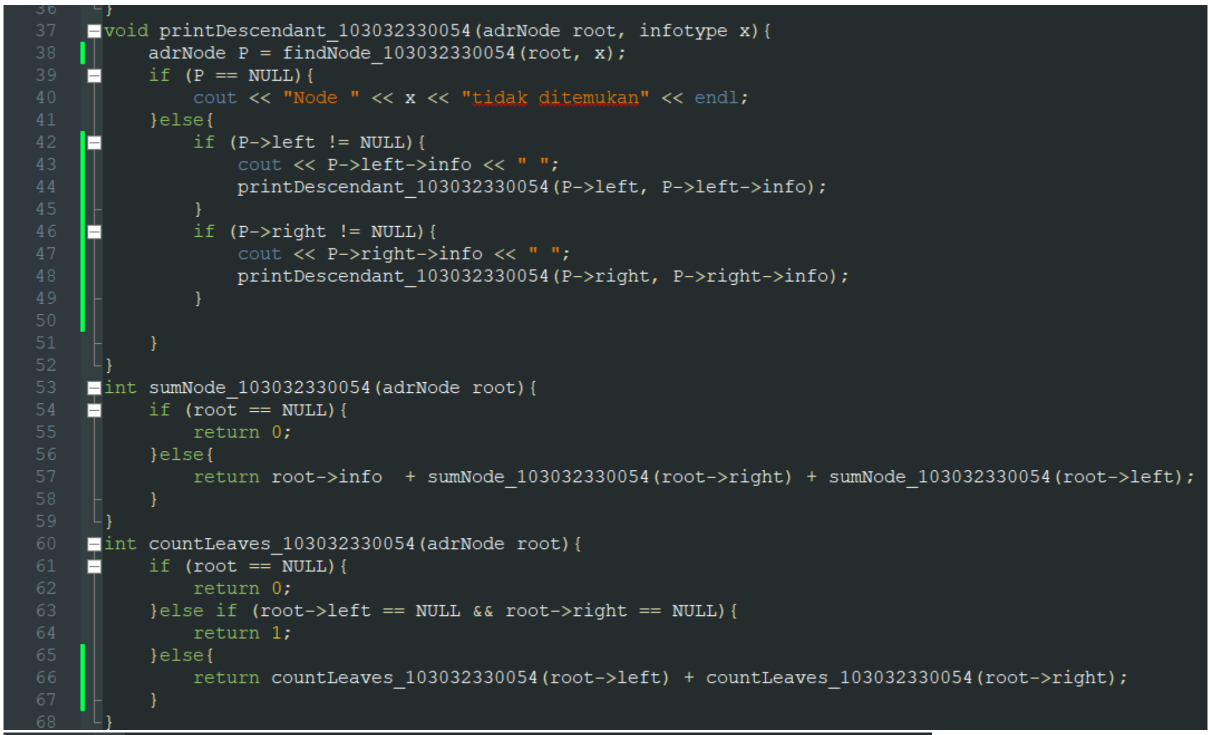


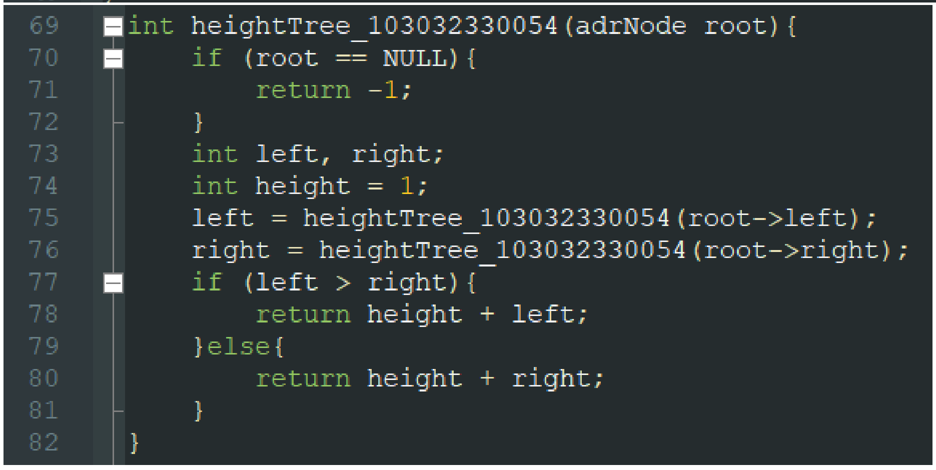
```cpp
#ifndef TREE_H_INCLUDED
#define TREE_H_INCLUDED
#include <iostream>
using namespace std;
typedef int infotype;
typedef struct elm *adrNode;
struct elm{
    adrNode right;
    adrNode left;
    infotype info;
};
adrNode newNode_103032330054(infotype x);
adrNode findNode_103032330054(adrNode root, infotype x);
void insertNode_103032330054(adrNode &root, adrNode p);
void printPreOrder_103032330054(adrNode root);
void printDescendant_103032330054(adrNode root, infotype x);
int sumNode_103032330054(adrNode root);
int countLeaves_103032330054(adrNode root);
int heightTree_103032330054(adrNode root);
#endif // TREE_H_INCLUDED
```

Tree.cpp

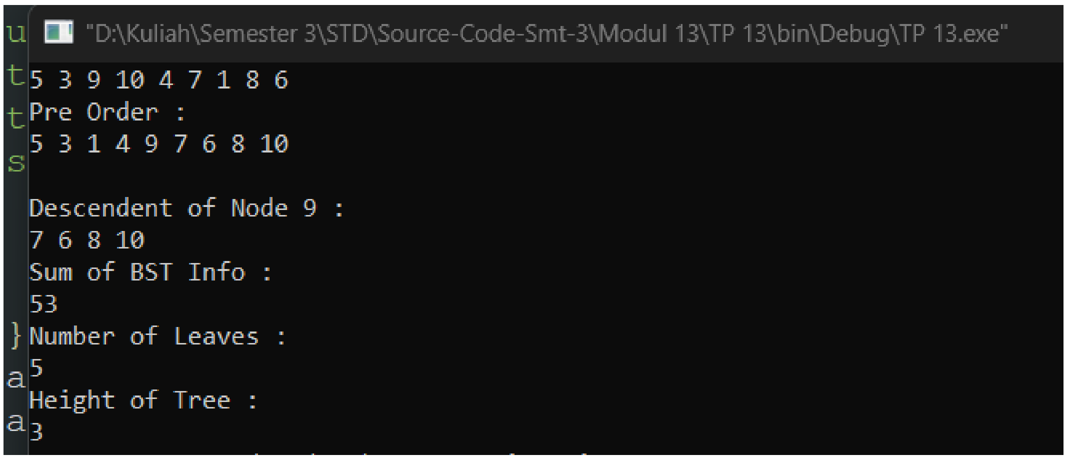


```cpp
#include "Tree.h"
adrNode newNode_103032330054(infotype x){
    adrNode P = new elm;
    P->left = NULL;
    P->right = NULL;
    P->info = x;
    return P;
}
adrNode findNode_103032330054(adrNode root, infotype x){
    if (root->info == x || root == NULL){
        return root;
    }
    if (x > root->info){
        return findNode_103032330054(root->right, x);
    }else if (x < root->info) {
        return findNode_103032330054(root->right, x);
    }
}
void insertNode_103032330054(adrNode &root, adrNode p){
    if (root == NULL){
        root = p;
    }else{
        if (p->info > root->info){
            insertNode_103032330054(root->right, p);
        }else{
            insertNode_103032330054(root->left, p);
        }
    }
}
void printPreOrder_103032330054(adrNode root){
    if (root != NULL){
        cout << root->info << " ";
        printPreOrder_103032330054(root->left);
        printPreOrder_103032330054(root->right);
    }
}
```


```
}
void printDescendant_103032330054(adrNode root, infotype x){
    adrNode P = findNode_103032330054(root, x);
    if (P == NULL){
        cout << "Node " << x << "tidak ditemukan" << endl;
    }else{
        if (P->left != NULL){
            cout << P->left->info << " ";
            printDescendant_103032330054(P->left, P->left->info);
        }
        if (P->right != NULL){
            cout << P->right->info << " ";
            printDescendant_103032330054(P->right, P->right->info);
        }

    }
}
int sumNode_103032330054(adrNode root){
    if (root == NULL){
        return 0;
    }else{
        return root->info  + sumNode_103032330054(root->right) + sumNode_103032330054(root->left);
    }
}
int countLeaves_103032330054(adrNode root){
    if (root == NULL){
        return 0;
    }else if (root->left == NULL && root->right == NULL){
        return 1;
    }else{
        return countLeaves_103032330054(root->left) + countLeaves_103032330054(root->right);
    }
}
```


```
int heightTree_103032330054(adrNode root){
    if (root == NULL){
        return -1;
    }
    int left, right;
    int height = 1;
    left = heightTree_103032330054(root->left);
    right = heightTree_103032330054(root->right);
    if (left > right){
        return height + left;
    }else{
        return height + right;
    }
}
```

Output


"D:\Kuliah\Semester 3\STD\Source-Code-Smt-3\Modul 13\TP 13\bin\Debug\TP 13.exe"

```
5 3 9 10 4 7 1 8 6
Pre Order :
5 3 1 4 9 7 6 8 10

Descendent of Node 9 :
7 6 8 10
Sum of BST Info :
53
Number of Leaves :
5
Height of Tree :
3
```