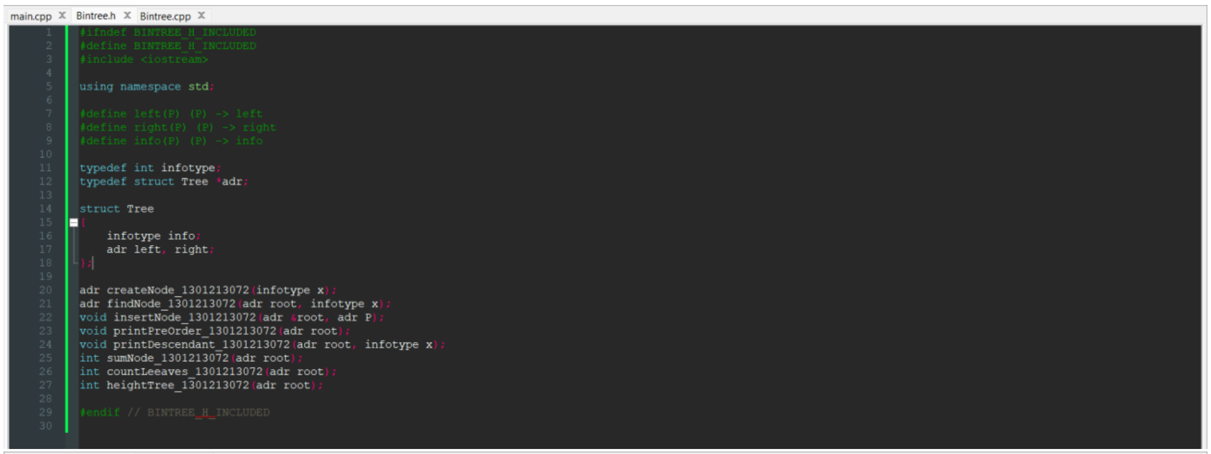
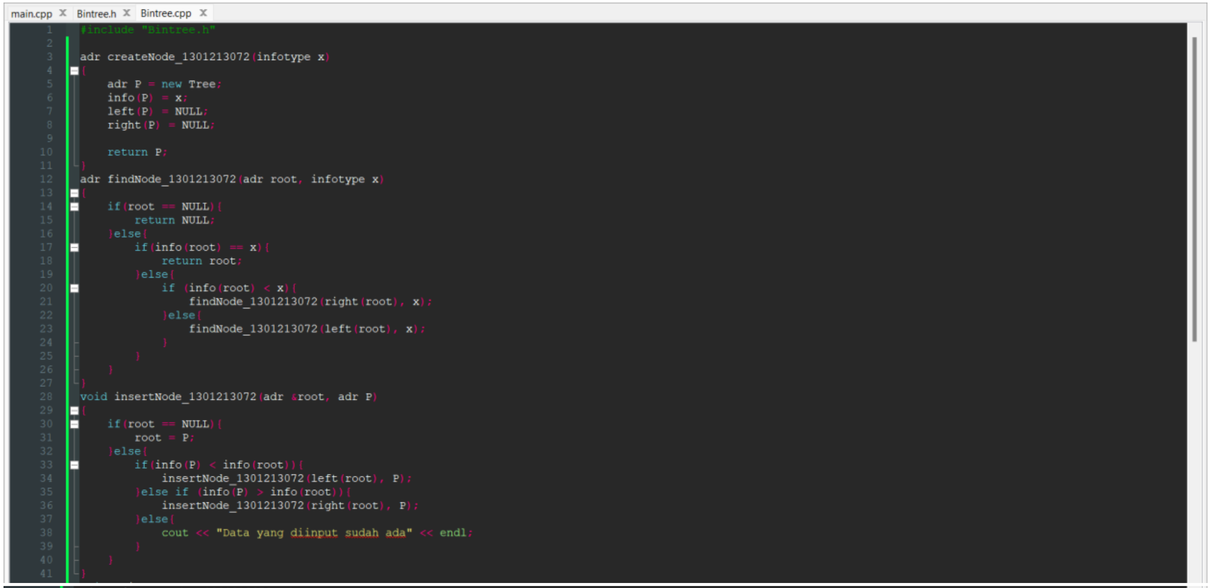
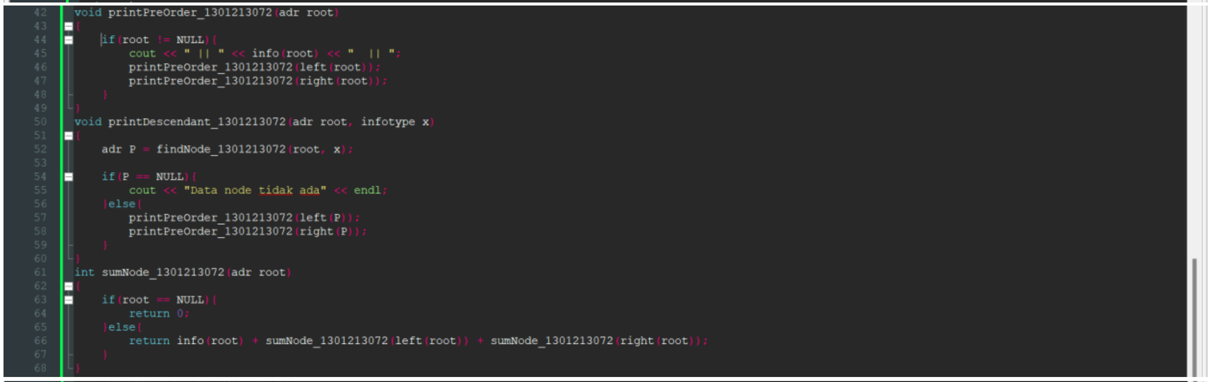


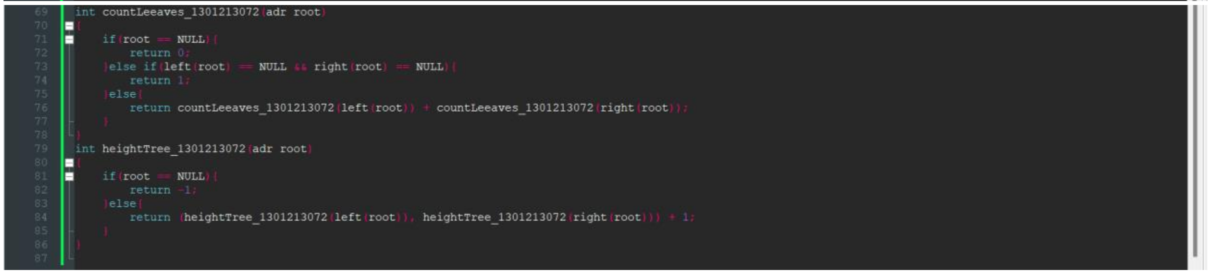
```cpp
#ifndef BINTREE_H_INCLUDED
#define BINTREE_H_INCLUDED
#include <iostream>

using namespace std;

#define left(P) (P) -> left
#define right(P) (P) -> right
#define info(P) (P) -> info

typedef int infotype;
typedef struct Tree *adr;

struct Tree
{
    infotype info;
    adr left, right;
};

adr createNode_1301213072(infotype x);
adr findNode_1301213072(adr root, infotype x);
void insertNode_1301213072(adr &root, adr P);
void printPreOrder_1301213072(adr root);
void printDescendant_1301213072(adr root, infotype x);
int sumNode_1301213072(adr root);
int countLeeaves_1301213072(adr root);
int heightTree_1301213072(adr root);

#endif // BINTREE_H_INCLUDED
```



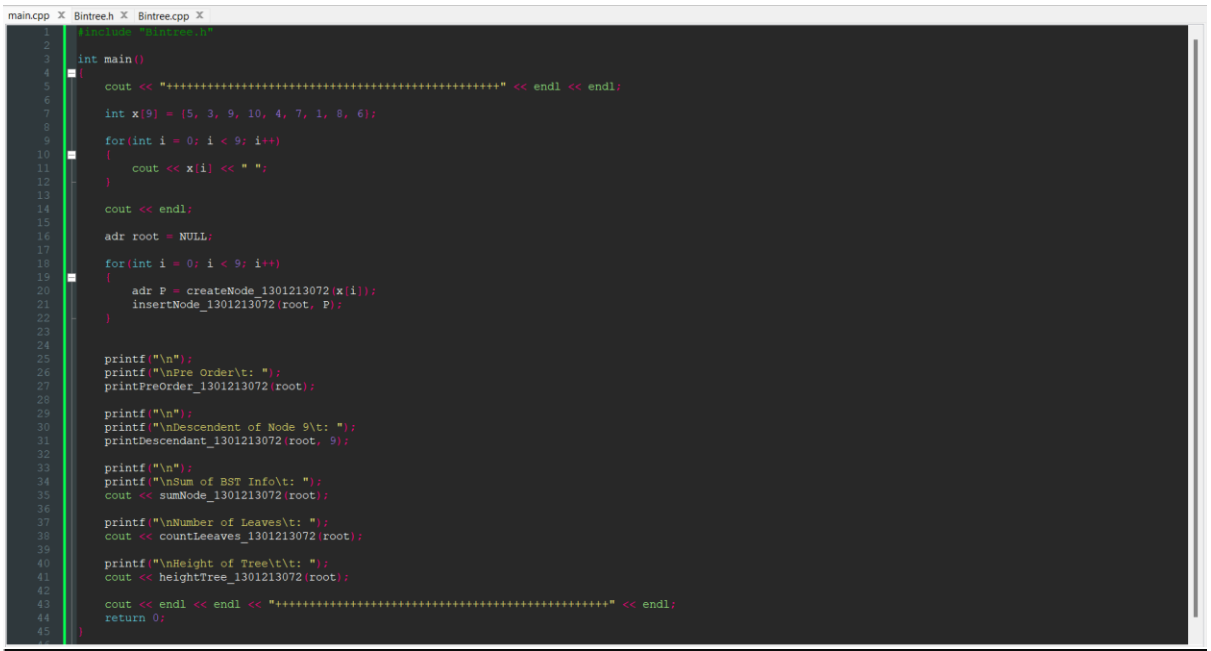
```cpp
#include "Bintree.h"

adr createNode_1301213072(infotype x)
{
    adr P = new Tree;
    info(P) = x;
    left(P) = NULL;
    right(P) = NULL;

    return P;
}
adr findNode_1301213072(adr root, infotype x)
{
    if(root == NULL){
        return NULL;
    }else{
        if(info(root) == x){
            return root;
        }else{
            if (info(root) < x){
                findNode_1301213072(right(root), x);
            }else{
                findNode_1301213072(left(root), x);
            }
        }
    }
}
void insertNode_1301213072(adr &root, adr P)
{
    if(root == NULL){
        root = P;
    }else{
        if(info(P) < info(root)){
            insertNode_1301213072(left(root), P);
        }else if (info(P) > info(root)){
            insertNode_1301213072(right(root), P);
        }else{
            cout << "Data yang diinput sudah ada" << endl;
        }
    }
}
```



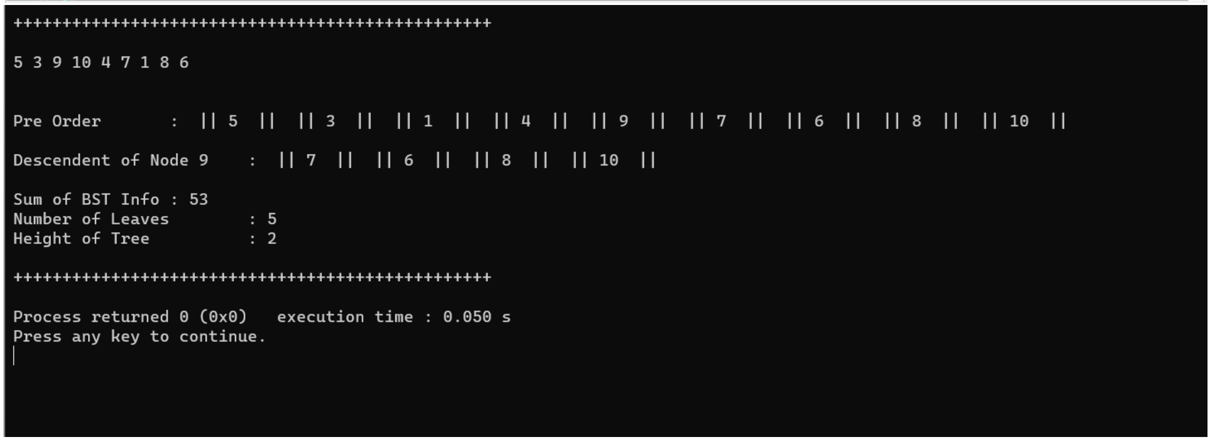
```cpp
void printPreOrder_1301213072(adr root)
{
    if(root != NULL){
        cout << " || " << info(root) << "  || ";
        printPreOrder_1301213072(left(root));
        printPreOrder_1301213072(right(root));
    }
}
void printDescendant_1301213072(adr root, infotype x)
{
    adr P = findNode_1301213072(root, x);

    if(P == NULL){
        cout << "Data node tidak ada" << endl;
    }else{
        printPreOrder_1301213072(left(P));
        printPreOrder_1301213072(right(P));
    }
}
int sumNode_1301213072(adr root)
{
    if(root == NULL){
        return 0;
    }else{
        return info(root) + sumNode_1301213072(left(root)) + sumNode_1301213072(right(root));
    }
}
```



```cpp
int countLeeaves_1301213072(adr root)
{
    if(root == NULL){
        return 0;
    }else if(left(root) == NULL && right(root) == NULL){
        return 1;
    }else{
        return countLeeaves_1301213072(left(root)) + countLeeaves_1301213072(right(root));
    }
}
int heightTree_1301213072(adr root)
{
    if(root == NULL){
        return -1;
    }else{
        return (heightTree_1301213072(left(root)), heightTree_1301213072(right(root))) + 1;
    }
}
```


main.cpp Bintree.h Bintree.cpp

```cpp
#include "Bintree.h"

int main()
{
    cout << "++++++++++++++++++++++++++++++++++++++++++++++++++" << endl << endl;

    int x[9] = {5, 3, 9, 10, 4, 7, 1, 8, 6};

    for(int i = 0; i < 9; i++)
    {
        cout << x[i] << " ";
    }

    cout << endl;

    adr root = NULL;

    for(int i = 0; i < 9; i++)
    {
        adr P = createNode_1301213072(x[i]);
        insertNode_1301213072(root, P);
    }


    printf("\n");
    printf("\nPre Order\t: ");
    printPreOrder_1301213072(root);

    printf("\n");
    printf("\nDescendent of Node 9\t: ");
    printDescendant_1301213072(root, 9);

    printf("\n");
    printf("\nSum of BST Info\t: ");
    cout << sumNode_1301213072(root);

    printf("\nNumber of Leaves\t: ");
    cout << countLeeaves_1301213072(root);

    printf("\nHeight of Tree\t\t: ");
    cout << heightTree_1301213072(root);

    cout << endl << endl << "++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;
    return 0;
}
```



```
+++++++++++++++++++++++++++++++++++++++++++++++++

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          : 2

+++++++++++++++++++++++++++++++++++++++++++++++++

Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.
```