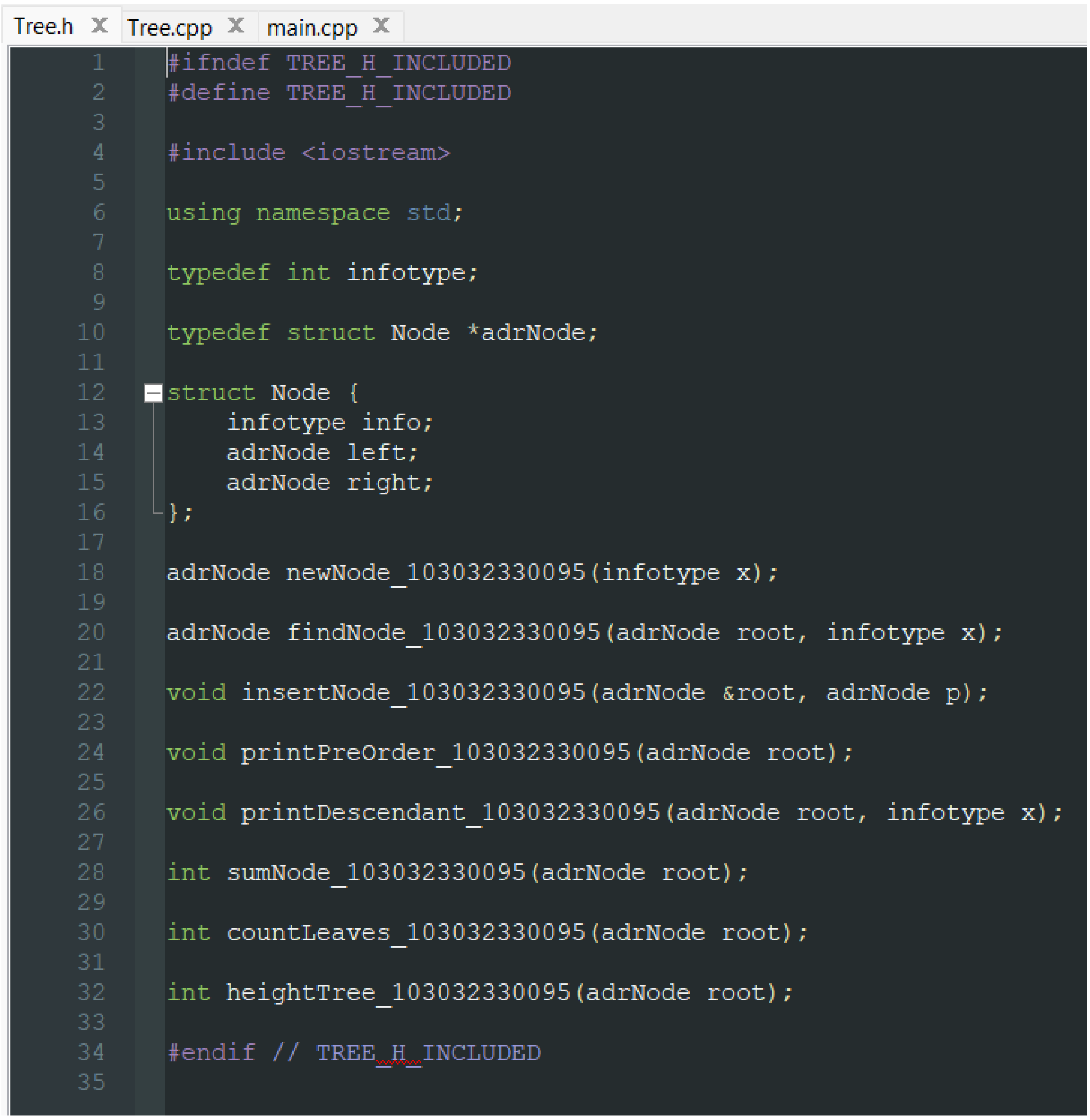
Gena Darma || 103032330095 || IT-47-KHS

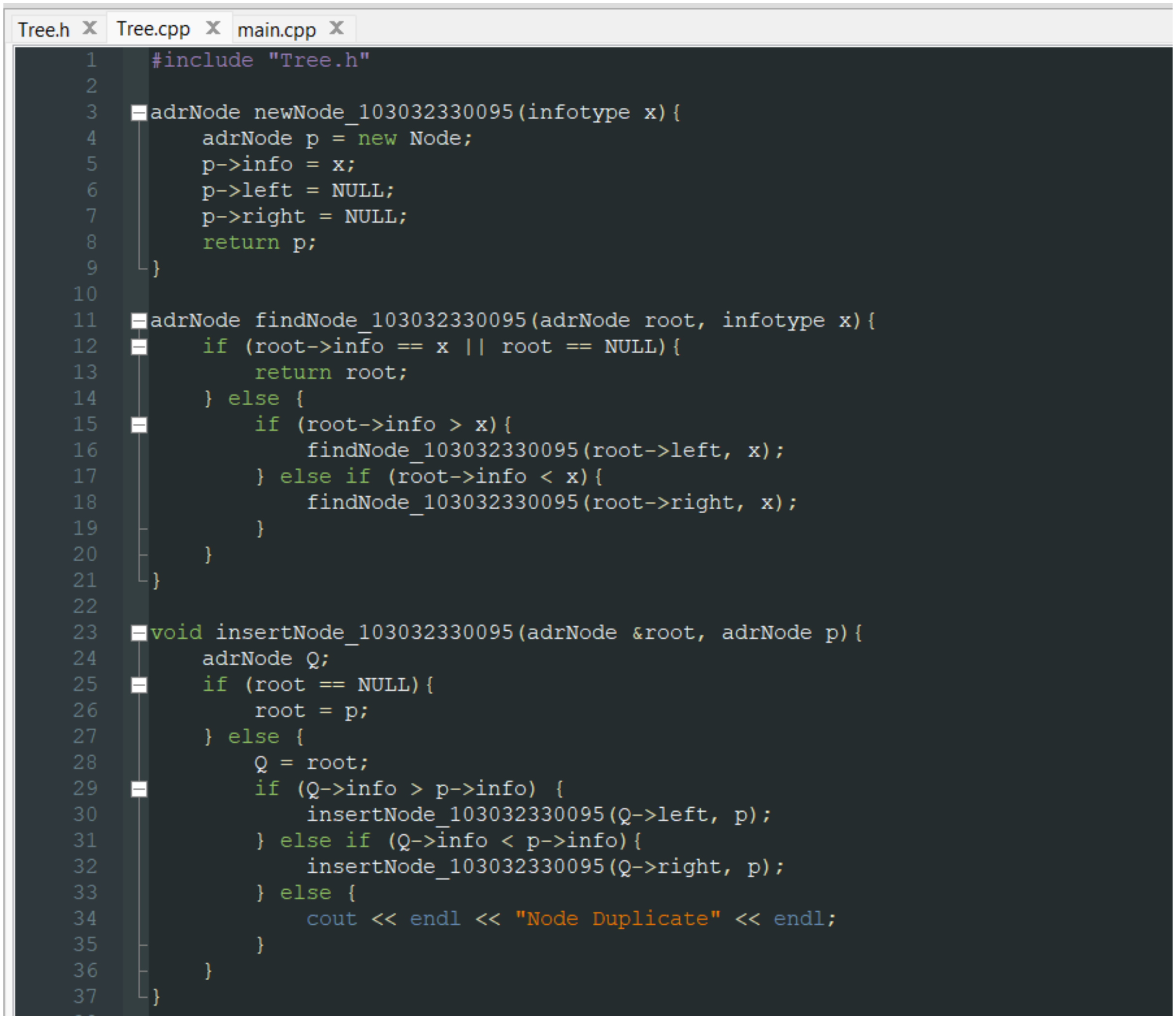
TP_MOD_13_103032330095_GENA DARMA

1. Tree.h

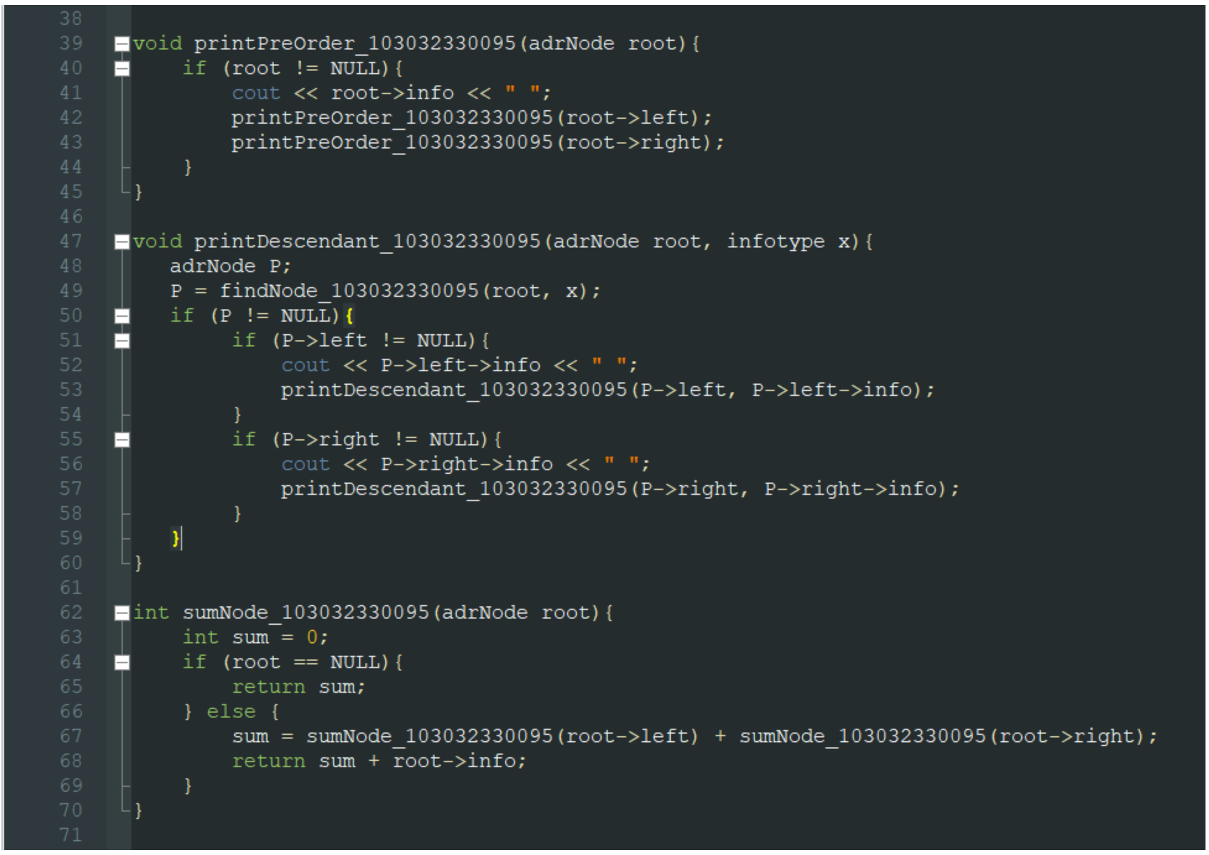


```
#ifndef TREE_H_INCLUDED
#define TREE_H_INCLUDED

#include <iostream>

using namespace std;

typedef int infotype;

typedef struct Node *adrNode;

struct Node {
    infotype info;
    adrNode left;
    adrNode right;
};

adrNode newNode_103032330095(infotype x);

adrNode findNode_103032330095(adrNode root, infotype x);

void insertNode_103032330095(adrNode &root, adrNode p);

void printPreOrder_103032330095(adrNode root);

void printDescendant_103032330095(adrNode root, infotype x);

int sumNode_103032330095(adrNode root);

int countLeaves_103032330095(adrNode root);

int heightTree_103032330095(adrNode root);

#endif // TREE_H_INCLUDED

```

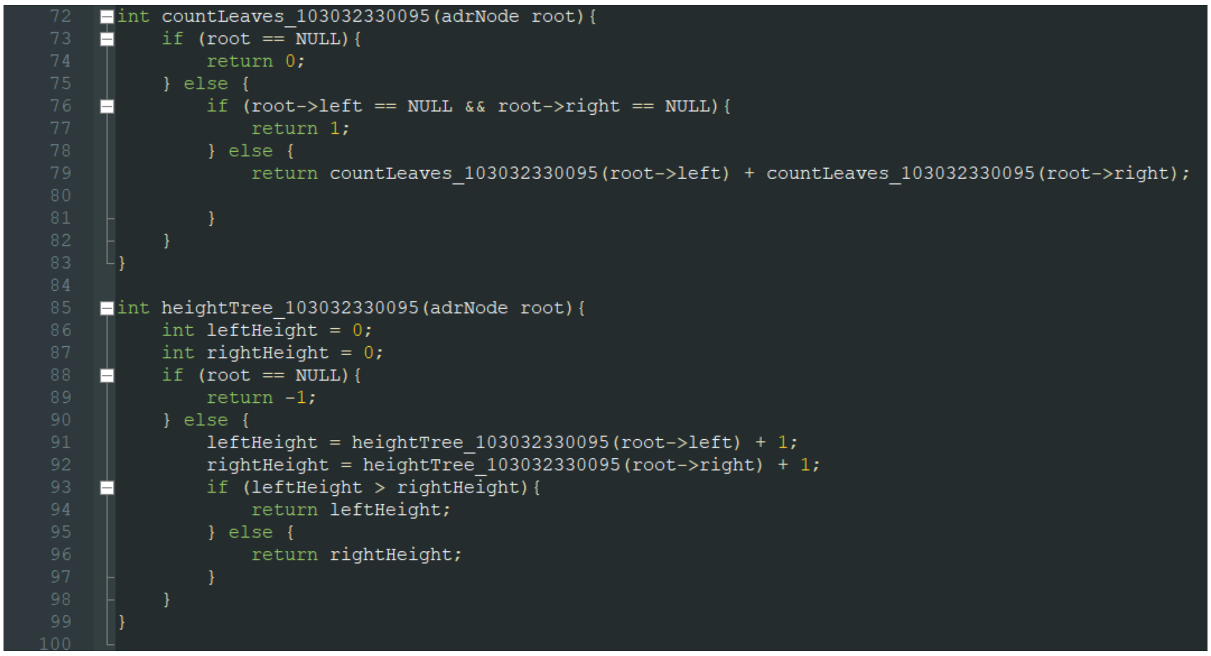
2. Tree.cpp



```cpp
#include "Tree.h"

adrNode newNode_103032330095(infotype x){
    adrNode p = new Node;
    p->info = x;
    p->left = NULL;
    p->right = NULL;
    return p;
}

adrNode findNode_103032330095(adrNode root, infotype x){
    if (root->info == x || root == NULL){
        return root;
    } else {
        if (root->info > x){
            findNode_103032330095(root->left, x);
        } else if (root->info < x){
            findNode_103032330095(root->right, x);
        }
    }
}

void insertNode_103032330095(adrNode &root, adrNode p){
    adrNode Q;
    if (root == NULL){
        root = p;
    } else {
        Q = root;
        if (Q->info > p->info)  {
            insertNode_103032330095(Q->left, p);
        } else if (Q->info < p->info){
            insertNode_103032330095(Q->right, p);
        } else {
            cout << endl << "Node Duplicate" << endl;
        }
    }
}
```

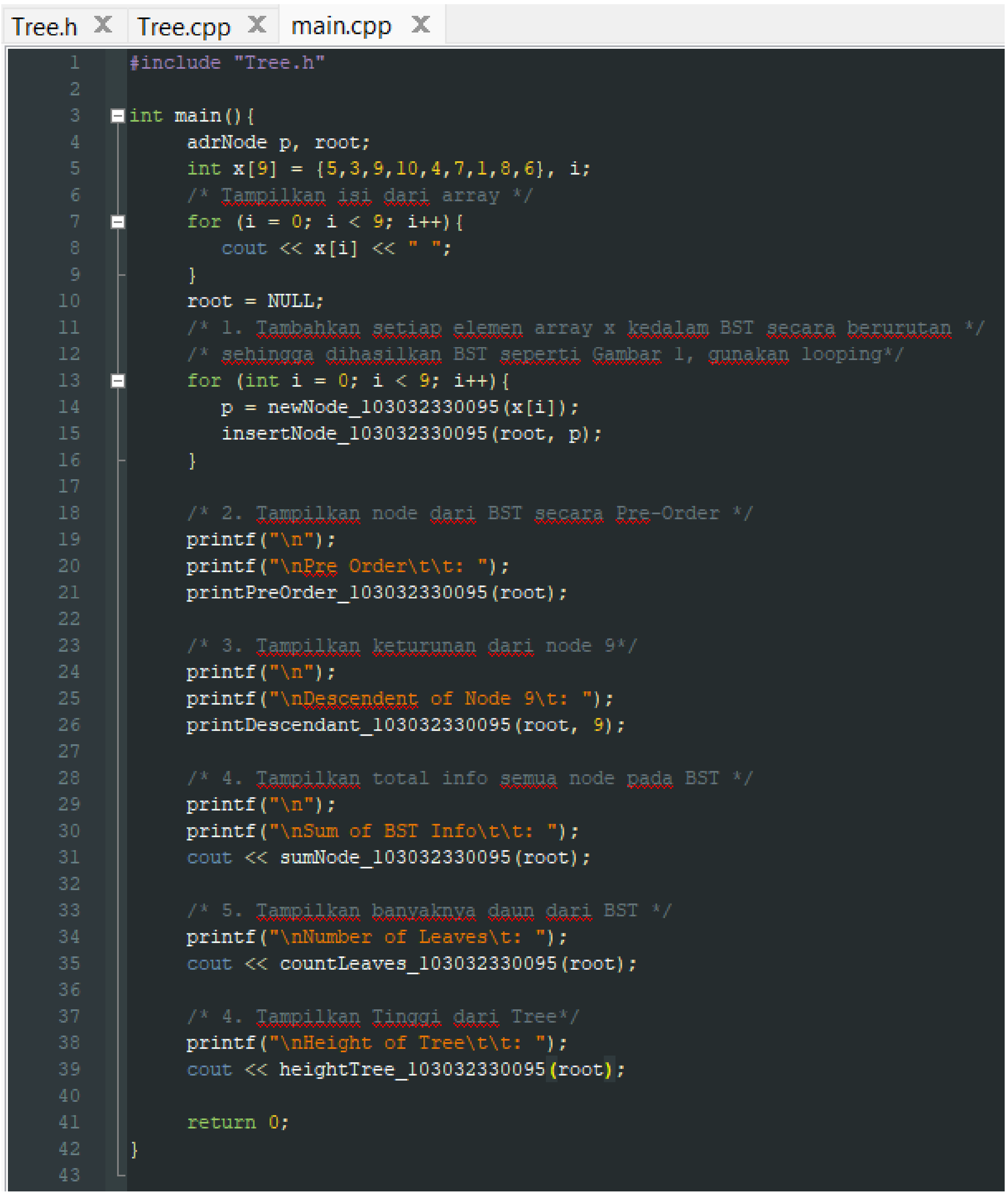


```cpp

void printPreOrder_103032330095(adrNode root){
    if (root != NULL){
        cout << root->info << " ";
        printPreOrder_103032330095(root->left);
        printPreOrder_103032330095(root->right);
    }
}

void printDescendant_103032330095(adrNode root, infotype x){
    adrNode P;
    P = findNode_103032330095(root, x);
    if (P != NULL){
        if (P->left != NULL){
            cout << P->left->info << " ";
            printDescendant_103032330095(P->left, P->left->info);
        }
        if (P->right != NULL){
            cout << P->right->info << " ";
            printDescendant_103032330095(P->right, P->right->info);
        }
    }
}

int sumNode_103032330095(adrNode root){
    int sum = 0;
    if (root == NULL){
        return sum;
    } else {
        sum = sumNode_103032330095(root->left) + sumNode_103032330095(root->right);
        return sum + root->info;
    }
}

```



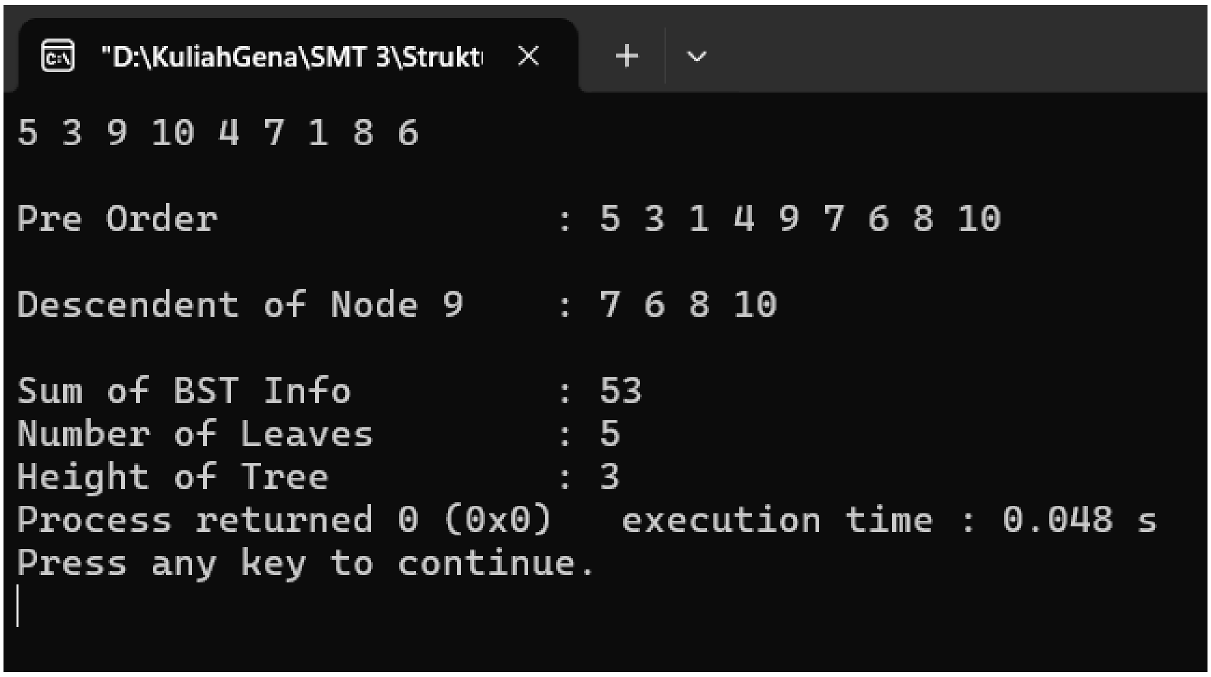
```cpp
int countLeaves_103032330095(adrNode root){
    if (root == NULL){
        return 0;
    } else {
        if (root->left == NULL && root->right == NULL){
            return 1;
        } else {
            return countLeaves_103032330095(root->left) + countLeaves_103032330095(root->right);

        }
    }
}

int heightTree_103032330095(adrNode root){
    int leftHeight = 0;
    int rightHeight = 0;
    if (root == NULL){
        return -1;
    } else {
        leftHeight = heightTree_103032330095(root->left) + 1;
        rightHeight = heightTree_103032330095(root->right) + 1;
        if (leftHeight > rightHeight){
            return leftHeight;
        } else {
            return rightHeight;
        }
    }
}

```

3. Main.cpp


Tree.h ✕ Tree.cpp ✕ main.cpp ✕

```cpp
#include "Tree.h"

int main(){
      adrNode p, root;
      int x[9] = {5,3,9,10,4,7,1,8,6}, i;
      /* Tampilkan isi dari array */
      for (i = 0; i < 9; i++){
         cout << x[i] << " ";
      }
      root = NULL;
      /* 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++){
         p = newNode_103032330095(x[i]);
         insertNode_103032330095(root, p);
      }

      /* 2. Tampilkan node dari BST secara Pre-Order */
      printf("\n");
      printf("\nPre Order\t\t: ");
      printPreOrder_103032330095(root);

      /* 3. Tampilkan keturunan dari node 9*/
      printf("\n");
      printf("\nDescendent of Node 9\t: ");
      printDescendant_103032330095(root, 9);

      /* 4. Tampilkan total info semua node pada BST */
      printf("\n");
      printf("\nSum of BST Info\t\t: ");
      cout << sumNode_103032330095(root);

      /* 5. Tampilkan banyaknya daun dari BST */
      printf("\nNumber of Leaves\t: ");
      cout << countLeaves_103032330095(root);

      /* 4. Tampilkan Tinggi dari Tree*/
      printf("\nHeight of Tree\t\t: ");
      cout << heightTree_103032330095(root);

      return 0;
}

```

4. Output



```
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
Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```