

Main.cpp

main.cpp x

```
1  #include "Tree.h"
2  int main()
3  {
4      int x[9] = {5,3,9,10,4,7,1,8,6};
5      /* Tampilkan isi dari array */
6      adrNode root = NULL;
7      for (int i=0; i<9; i++){
8          cout << x[i] << " ";
9      }
10     /* 1. Tambahkan setiap elemen array x kedalam BST secara berurutan */
11     /* sehingga dihasilkan BST seperti Gambar 1, gunakan looping*/
12     for (int i=0; i<9; i++){
13         insertNode_103032330054(root, newNode_103032330054(x[i]));
14     }
15     /* 2. Tampilkan node dari BST secara Pre-Order */
16     cout << endl;
17     cout << "Pre Order :" << endl;
18     printPreOrder_103032330054(root);
19     cout << endl;
20
21     /* 3. Tampilkan keturunan dari node 9*/
22     cout << endl << "Descendent of Node 9 :" << endl;
23     printDescendant_103032330054(root, 9);
24
25     /* 4. Tampilkan total info semua node pada BST */
26     cout << endl;
27     cout << "Sum of BST Info :" << endl;
28     cout << sumNode_103032330054(root);
29     /* 5. Tampilkan banyaknya daun dari BST */
30     cout << endl;
31     cout << "Number of Leaves :" << endl;
32     cout << countLeaves_103032330054(root);
33     /* 6. Tampilkan Tinggi dari Tree*/
34     cout << endl << "Height of Tree :" << endl;
35     cout << heightTree_103032330054(root);
36     return 0;
37 }
38
```

Tree.h

```
Tree.h x main.cpp x Tree.cpp x
1  #ifndef TREE_H_INCLUDED
2  #define TREE_H_INCLUDED
3  #include <iostream>
4  using namespace std;
5  typedef int infotype;
6  typedef struct elm *adrNode;
7  struct elm{
8      adrNode right;
9      adrNode left;
10     infotype info;
11 };
12 adrNode newNode_103032330054(infotype x);
13 adrNode findNode_103032330054(adrNode root, infotype x);
14 void insertNode_103032330054(adrNode &root, adrNode p);
15 void printPreOrder_103032330054(adrNode root);
16 void printDescendant_103032330054(adrNode root, infotype x);
17 int sumNode_103032330054(adrNode root);
18 int countLeaves_103032330054(adrNode root);
19 int heightTree_103032330054(adrNode root);
20 #endif // TREE_H_INCLUDED
21
```

Tree.cpp

```
Tree.h X main.cpp X Tree.cpp X
1  #include "Tree.h"
2  adrNode newNode_103032330054 (infotype x) {
3      adrNode P = new elm;
4      P->left = NULL;
5      P->right = NULL;
6      P->info = x;
7      return P;
8  }
9  adrNode findNode_103032330054 (adrNode root, infotype x) {
10     if (root->info == x || root == NULL) {
11         return root;
12     }
13     if (x > root->info) {
14         return findNode_103032330054 (root->right, x);
15     } else if (x < root->info) {
16         return findNode_103032330054 (root->left, x);
17     }
18 }
19 void insertNode_103032330054 (adrNode &root, adrNode p) {
20     if (root == NULL) {
21         root = p;
22     } else {
23         if (p->info > root->info) {
24             insertNode_103032330054 (root->right, p);
25         } else {
26             insertNode_103032330054 (root->left, p);
27         }
28     }
29 }
30 void printPreOrder_103032330054 (adrNode root) {
31     if (root != NULL) {
32         cout << root->info << " ";
33         printPreOrder_103032330054 (root->left);
34         printPreOrder_103032330054 (root->right);
35     }
36 }
```

```

36 }
37 void printDescendant_103032330054(adNode root, infotype x){
38     adNode P = findNode_103032330054(root, x);
39     if (P == NULL){
40         cout << "Node " << x << "tidak ditemukan" << endl;
41     }else{
42         if (P->left != NULL){
43             cout << P->left->info << " ";
44             printDescendant_103032330054(P->left, P->left->info);
45         }
46         if (P->right != NULL){
47             cout << P->right->info << " ";
48             printDescendant_103032330054(P->right, P->right->info);
49         }
50     }
51 }
52 }
53 int sumNode_103032330054(adNode root){
54     if (root == NULL){
55         return 0;
56     }else{
57         return root->info + sumNode_103032330054(root->right) + sumNode_103032330054(root->left);
58     }
59 }
60 int countLeaves_103032330054(adNode root){
61     if (root == NULL){
62         return 0;
63     }else if (root->left == NULL && root->right == NULL){
64         return 1;
65     }else{
66         return countLeaves_103032330054(root->left) + countLeaves_103032330054(root->right);
67     }
68 }
69 int heightTree_103032330054(adNode root){
70     if (root == NULL){
71         return -1;
72     }
73     int left, right;
74     int height = 1;
75     left = heightTree_103032330054(root->left);
76     right = heightTree_103032330054(root->right);
77     if (left > right){
78         return height + left;
79     }else{
80         return height + right;
81     }
82 }

```

Output

```

u "D:\Kuliah\Semester 3\STD\Source-Code-Smt-3\Modul 13\TP 13\bin\Debug\TP 13.exe"
t 5 3 9 10 4 7 1 8 6
t Pre Order :
s 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
a Height of Tree :
a 3

```