

## POST TEST PRAKTIKUM STRUKTUR DATA DAN ALGORITME

NAMA :SRI REJEKI NIM :21102240
--------------------------------------

1. D
2. D
3. C
4. A
5. B

1.

```
/*  
Nama:Sri Rejeki  
NIM :21102240  
Kelas:S1 IF 09 G  
*/  
#include <iostream>  
using namespace std;  
struct data_node {  
    char nama[50];  
    char nim[20];  
};  
struct dlinkedlist {  
    dlinkedlist* prev;  
    data_node data;  
    dlinkedlist* next;  
};  
  
dlinkedlist* head;  
dlinkedlist* tail;  
dlinkedlist* vertex;  
void inisialisasi(){  
    head = NULL;  
    tail = NULL;  
}  
bool dLinkKosong(){  
    if(head == NULL && tail == NULL){  
        return true;  
    }else {  
        return false;  
    }  
}  
void tambahDepan(data_node du){  
    if(dLinkKosong() == true){
```

```

vertex = new dlinkedlist;
vertex->data = du;

vertex->prev = NULL;
vertex->next = NULL;
head = vertex;
tail = vertex;
} else {
vertex = new dlinkedlist;
vertex->data = du;
vertex->prev = NULL;
vertex->next = NULL;
vertex->next = head;
head->prev = vertex;
head = vertex;
}
}

void tambahBelakang(data_node du){
if(dLinkKosong() == true){
vertex = new dlinkedlist;
vertex->data = du;
vertex->prev = NULL;
vertex->next = NULL;
head = vertex;
tail = vertex;
} else {
vertex = new dlinkedlist;
vertex->data = du;
vertex->prev = NULL;
vertex->next = NULL;
tail->next = vertex;
vertex->prev = tail;
tail = vertex;
}
}

void tambahTengah(data_node du){
dlinkedlist* helper;
dlinkedlist* helper2;
int nomor = 2, pos;
vertex = new dlinkedlist;
vertex->data = du;
vertex->prev = NULL;
vertex->next = NULL;
cout << "Masukkan Posisi: ";
cin >> pos;
if (head == NULL){

cout << "List masih kosong!" << endl;
} else if (pos == 1){
cout << "Posisi bukan posisi tengah!" << endl;
} else if (pos < 1){

```

```

cout << "Posisi diluar jangkauan!" << endl;
}else{
helper = head;
while(helper){
if(nomor == pos){
helper2 = helper->next;
vertex->next = helper2;
helper2->prev = vertex;
helper->next = vertex;
vertex->prev = helper;
break;
}
helper = helper->next;
nomor++;
}
}
}

void hapusDepan(){
if(dLinkKosong() == true){
cout << "List Masih Kosong!" << endl;
}else{
dlinkedlist* helper, simpan;
simpan.data = head->data;
cout << "Data " << simpan.data.nama << " berhasil dihapus!" << endl;
helper = head;
if(head == tail){
head = NULL;
tail = NULL;
delete helper;
}else {
head = head->next;
head->prev = NULL;
helper->next = NULL;
delete helper;
}
}
}

void hapusBelakang(){
if(dLinkKosong() == true){
cout << "List Masih Kosong!" << endl;

}else {
if(head == tail){
dlinkedlist* helper, simpan;
simpan.data = tail->data;
cout << "Data " << simpan.data.nama << " berhasil dihapus!" << endl;
helper= head;
head = NULL;
tail = NULL;
delete helper;
}else {

```

```

dlinkedList* helper;
helper = tail;
tail = tail->prev;
tail->next = NULL;
helper->prev = NULL;
delete helper;
}
}
}

void hapusTengah(){
int nomor = 1, pos;
dlinkedList *hapus, *helper, *helper2;
cout << "Masukkan Posisi: ";
cin >> pos;
if (head == NULL){
cout << "List masih kosong!" << endl;
}else if (pos == 1){
cout << "Posisi bukan posisi tengah!" << endl;
}else if (pos < 1){
cout << "Posisi diluar jangkauan!" << endl;
}else{
hapus = head;
while(hapus){
if(nomor == pos){
helper2 = hapus->next;
helper = hapus->prev;
helper->next = helper2;
helper2->prev = helper;
delete hapus;
}
hapus = hapus->next;
nomor++;
}
}

void tampil(){
if(dLinkKosong() == true){
cout << "List Masih Kosong!" << endl;
}else {
dlinkedList* helper;
helper = head;
while(helper != NULL){
cout << helper->data.nama << " \t" << helper->data.nim << endl;
helper = helper->next;
}
}
}

int main(){
inisialisasi();
dlinkedList temp;

```

```

int pilih;
menu:
system("cls");
cout << "PROGRAM DOUBLE LINKED LIST NON-CIRCULAR" << endl;
cout << "\n";
cout << "1. Tambah Depan" << endl;
cout << "2. Tambah Belakang" << endl;
cout << "3. Tambah Tengah" << endl;
cout << "4. Hapus Depan" << endl;
cout << "5. Hapus Belakang" << endl;
cout << "6. Hapus Tengah" << endl;
cout << "7. Tampilkan List" << endl;
cout << "0. KELUAR" << endl;
cout << "\n";
cout << "Pilih Operasi : ";
cin >> pilih;
cout << "\n";
switch(pilih){
case 1:
if (pilih == 1){
cout << "-Tambah Depan-" << endl;
cout << "Masukkan Nama : ";
cin >> temp.data.nama;
cout << "Masukkan NIM : ";
cin >> temp.data.nim;

cout << "Data " << temp.data.nama << " berhasil diinput!" << endl;
tambahDepan(temp.data);
}
break;
case 2:
if (pilih == 2){
cout << "-Tambah Belakang-" << endl;
cout << "Masukkan Nama : ";
cin >> temp.data.nama;
cout << "Masukkan NIM : ";
cin >> temp.data.nim;
cout << "Data " << temp.data.nama << " berhasil diinput!" << endl;
tambahBelakang(temp.data);
}
break;
case 3:
if (pilih == 3){
cout << "-Tambah Tengah-" << endl;
cout << "Masukkan Nama : ";
cin >> temp.data.nama;
cout << "Masukkan NIM : ";
cin >> temp.data.nim;
tambahTengah(temp.data);
cout << "Data " << temp.data.nama << " berhasil diinput!" << endl;
}
}

```

```

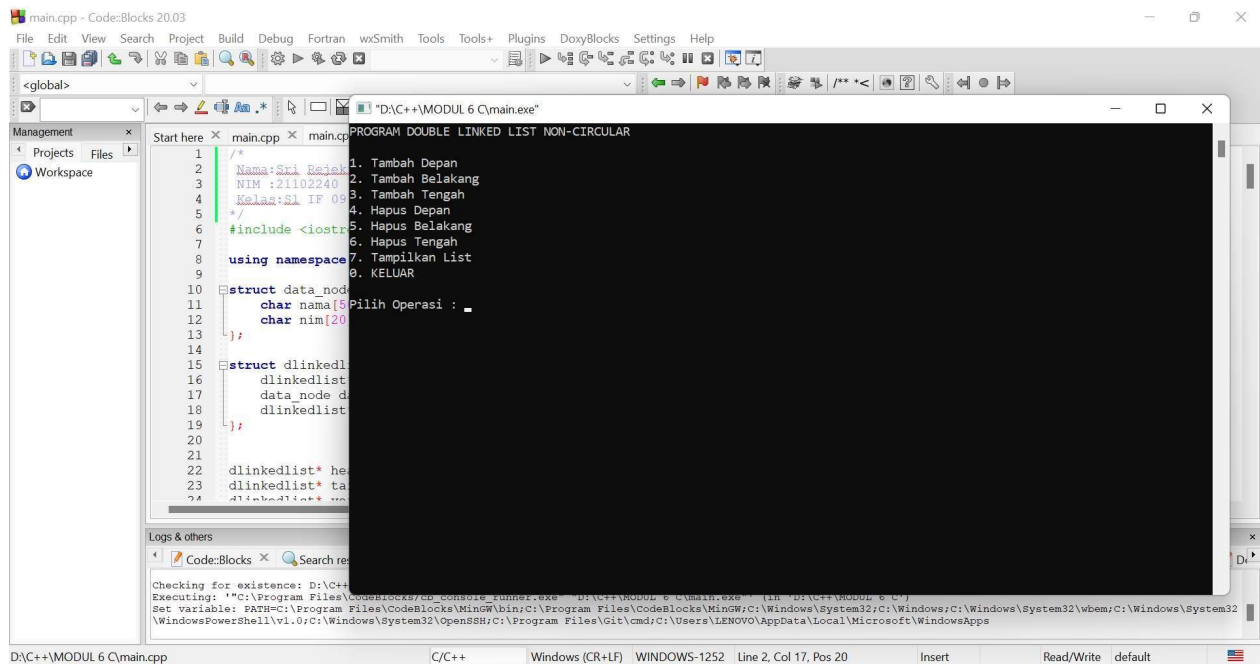
break;
case 4:
if (pilih == 4){
hapusDepan();
}
break;
case 5:
if (pilih == 5){
hapusBelakang();
}
break;
case 6:
if (pilih == 6){
hapusTengah();
cout << "Data berhasil dihapus!" << endl;
}
break;
case 7:

if (pilih == 7){
cout << "DATA MAHASISWA" << endl;
cout << "\n";
tampil();
}
break;
case 0:
if (pilih == 0){
system("cls");
}
break;
default:
exit(0);
}
cout << "\n";
system("pause");
goto menu;
cout << "/n";
return 0;
}

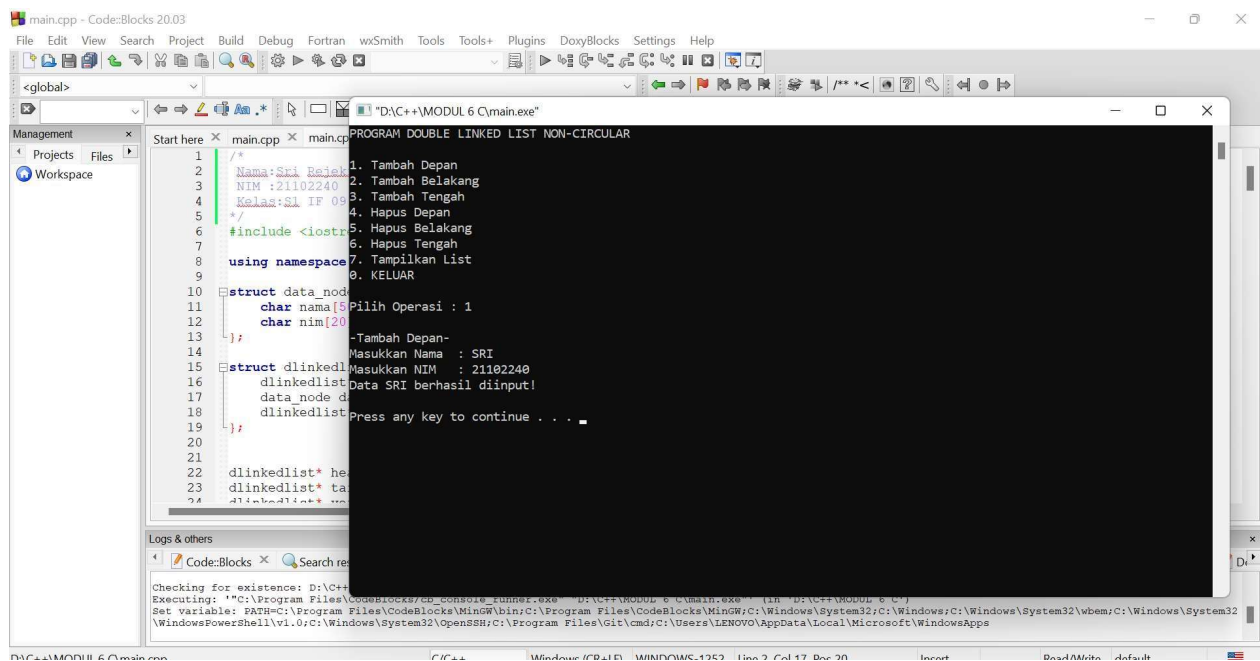
```

### ***Screenshort program***

>>      Menu



>> tambah depan



>> Tambah belakang

```
1 /*
2 Nama:Sri Beisaki
3 NIM :21102240
4 Kelas:SI IF 09
5 */
6 #include <iostream>
7 using namespace std;
8
9
10 struct data_node{
11     char nama[50];
12     char nim[20];
13 };
14
15 struct dllinkedlist{
16     dllinkedlist* head;
17     data_node data;
18     dllinkedlist* tail;
19 };
20
21
22 dllinkedlist* head;
23 dllinkedlist* tail;
24 dllinkedlist* tambahTengah;
```

PROGRAM DOUBLE LINKED LIST NON-CIRCULAR

1. Tambah Depan  
2. Tambah Belakang  
3. Tambah Tengah  
4. Hapus Depan  
5. Hapus Belakang  
6. Hapus Tengah  
7. Tampilkan List  
8. KELUAR

Pilih Operasi : 2

-Tambah Belakang-

Masukkan Nama : DIYAIH

Masukkan NIM : 21102241

Data DIYAIH berhasil diinput!

Press any key to continue . . .

>> Tambah tengah

```
1 /*
2 Nama:Sri Beisaki
3 NIM :21102240
4 Kelas:SI IF 09
5 */
6 #include <iostream>
7 using namespace std;
8
9
10 struct data_node{
11     char nama[50];
12     char nim[20];
13 };
14
15 struct dllinkedlist{
16     dllinkedlist* head;
17     data_node data;
18     dllinkedlist* tail;
19 };
20
21
22 dllinkedlist* head;
23 dllinkedlist* tail;
24 dllinkedlist* tambahTengah;
```

PROGRAM DOUBLE LINKED LIST NON-CIRCULAR

1. Tambah Depan  
2. Tambah Belakang  
3. Tambah Tengah  
4. Hapus Depan  
5. Hapus Belakang  
6. Hapus Tengah  
7. Tampilkan List  
8. KELUAR

Pilih Operasi : 3

-Tambah Tengah-

Masukkan Nama : RAHMAT

Masukkan NIM : 21102242

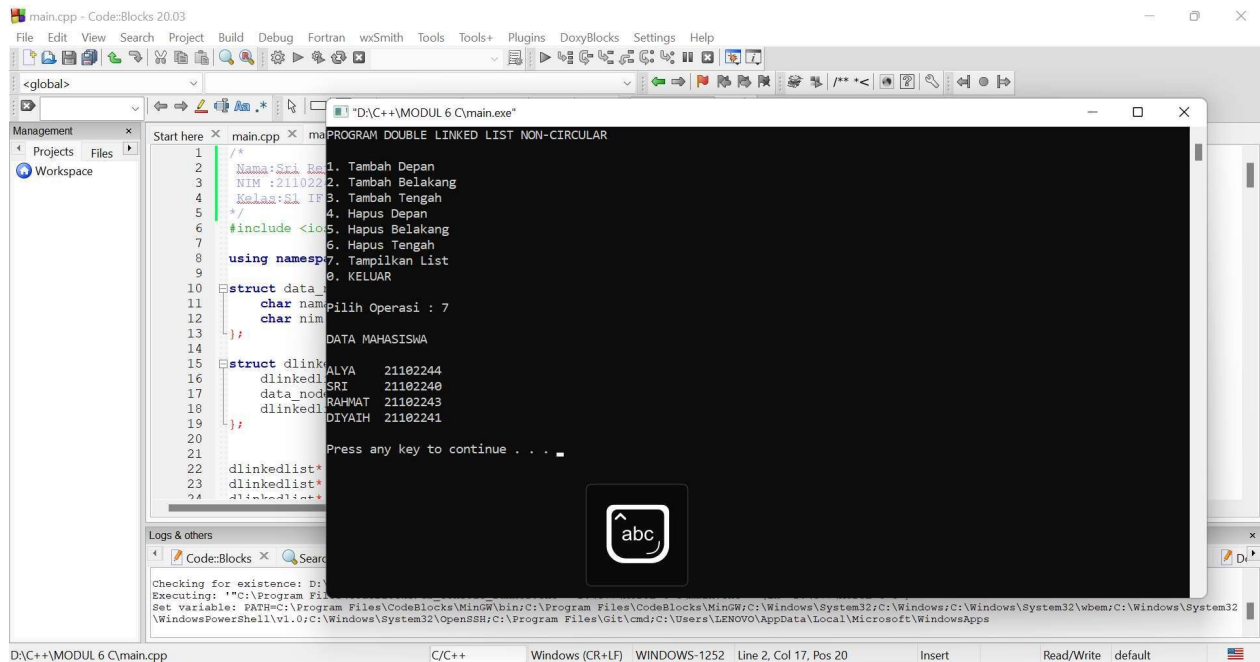
Masukkan Posisi : 2

List masih kosong!

Data RAHMAT berhasil diinput!

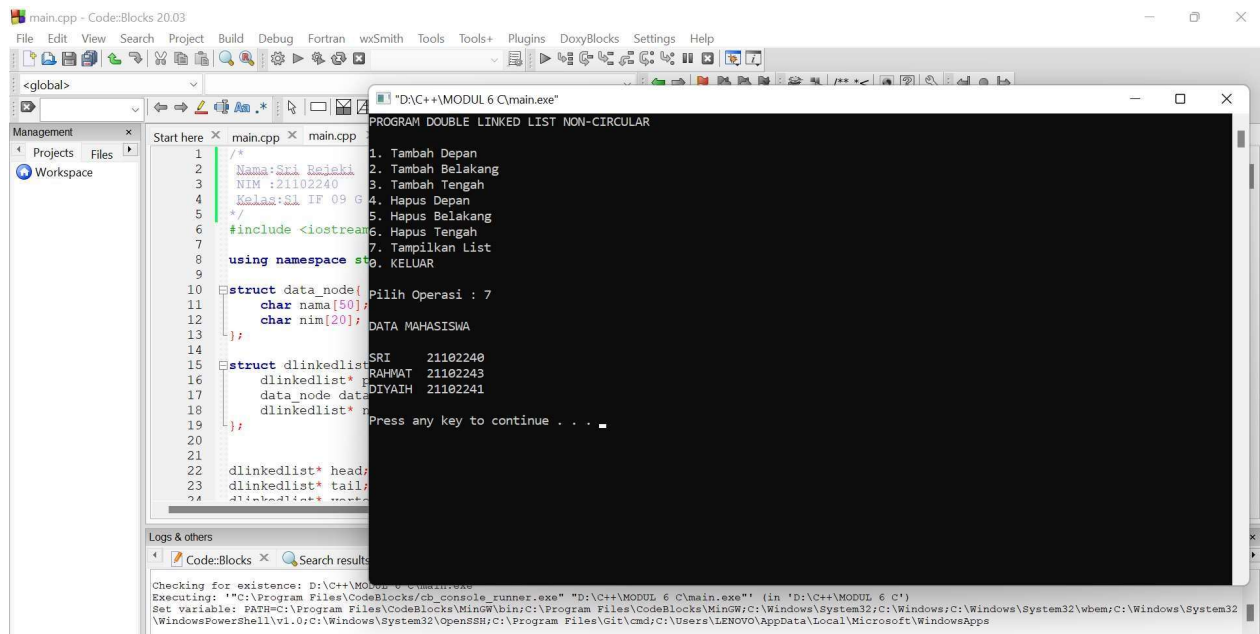
Press any key to continue . . .





```
1  /*
2  Nama: Sri 1. Tambah Depan
3  NIM : 21102240 2. Tambah Belakang
4  Kelas: SI IF 3. Tambah Tengah
5  */ 4. Hapus Depan
6  #include <iostream> 5. Hapus Belakang
7  6. Hapus Tengah
8  using namespace std; 7. Tampilkan List
9  0. KELUAR
10
11 struct data {
12     char nama;
13     char nim;
14 };
15
16 struct dlinklist {
17     data_node *head;
18     data_node *tail;
19 };
20
21 // Data Mahasiswa
22 data_node *dlinklist;
23 data_node *dlinklist;
24 data_node *dlinklist;
```

>> hapus depan



```
1  /*
2  Nama: Sri 1. Tambah Depan
3  NIM : 21102240 2. Tambah Belakang
4  Kelas: SI IF 3. Tambah Tengah
5  */ 4. Hapus Depan
6  #include <iostream> 5. Hapus Belakang
7  6. Hapus Tengah
8  using namespace std; 7. Tampilkan List
9  0. KELUAR
10
11 struct data_node {
12     char nama[50];
13     char nim[20];
14 };
15
16 struct dlinklist {
17     data_node *head;
18     data_node *tail;
19 };
20
21 // Data Mahasiswa
22 data_node *dlinklist;
23 data_node *dlinklist;
24 data_node *dlinklist;
```

>> Hapus tengah



```

*/

#include <iostream>

using namespace std;

// node
struct Node{
    string label;
    Node *left, *right, *parent;
};

// variabel pointer global
Node *root, *newNode;

// create New Tree
void createNewTree(string label)
{
    if( root != NULL )
        cout << "\nTree sudah dibuat" << endl;
    else{
        root = new Node();
        root->label = label;
        root->left = NULL;
        root->right = NULL;
        root->parent = NULL;
        cout << "\nNode " << label << " berhasil dibuat menjadi root." << endl;
    }
}

// insert Left
Node *insertLeft( string label, Node *node )
{
    if( root == NULL ){
        cout << "\nBuat tree terlebih dahulu!!" << endl;
        return NULL;
    }else{
        // cek apakah anak kiri ada atau tidak
        if( node->left != NULL ){
            // kalau ada
            cout << "\nNode " << node->label << " sudah ada anak kiri!!" << endl;
            return NULL;
        }else{

```

```

// kalau tidak ada

    newNode = new Node();
    newNode->label = label;
    newNode->left = NULL;
    newNode->right = NULL;
    newNode->parent = node;
    node->left = newNode;
    cout << "\nNode " << label << " berhasil ditambahkan ke anak kiri " <<
    newNode->parent->label << endl;
    return newNode;
}
}
}

// insert right
Node *insertRight( string label, Node *node )
{
    if( root == NULL ){
        cout << "\nBuat tree terlebih dahulu!!" << endl;
        return NULL;
    }else{
        // cek apakah anak kiri ada atau tidak
        if( node->right != NULL ){
            // kalau ada
            cout << "\nNode " << node->label << " sudah ada anak kanan!!" << endl;
            return NULL;
        }else{
            // kalau tidak ada
            newNode = new Node();
            newNode->label = label;
            newNode->left = NULL;
            newNode->right = NULL;
            newNode->parent = node;
            node->right = newNode;
            cout << "\nNode" << label << " berhasil ditambahkan ke anak kanan " <<
            newNode->parent->label << endl;
            return newNode;
        }
    }
}

// Empty

```

```

bool empty()
{
    if( root == NULL )
        return true;
    else
        return false;
}

// update
void update(string label, Node *node)
{
    if( !root ){
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    }else{
        if( !node )
            cout << "\nNode yang ingin diganti tidak ada!!" << endl;
        else{
            string temp = node->label;
            node->label = label;
            cout << "\nLabel node " << temp << " berhasil diubah menjadi " << label << endl;
        }
    }
}

// retrieve
void retrieve( Node *node )
{
    if( !root ){
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    }else{
        if( !node )
            cout << "\nNode yang ditunjuk tidak ada!!" << endl;
        else{
            cout << "\nLabel node : " << node->label << endl;
        }
    }
}

// Find
void find( Node *node )
{
    if( !root ){
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    }
}

```

```

    }else{
        if( !node )
            cout << "\nNode yang ditunjuk tidak ada!!" << endl;

        else{
            cout << "\nLabel Node : " << node->label << endl;
            cout << "Root Node : " << root->label << endl;

            if( !node->parent )
                cout << "Parent Node : (tidak punya orang tua)" << endl;
            else
                cout << "Parent Node : " << node->parent->label << endl;

            if( node->parent != NULL && node->parent->left != node && node->parent->right == node
)
                cout << "Saudara : " << node->parent->left->label << endl;
            else if( node->parent != NULL && node->parent->right != node &&
node->parent->left == node )
                cout << "Saudara : " << node->parent->right->label << endl;
            else
                cout << "Saudara : (tidak punya saudara)" << endl;

            if( !node->left )
                cout << "Anak Kiri Node : (tidak punya anak kiri)" << endl;
            else
                cout << "Anak Kiri Node : " << node->left->label << endl;

            if( !node->right )
                cout << "Anak Kanan Node : (tidak punya anak kanan)" << endl;
            else
                cout << "Anak Kanan Node : " << node->right->label << endl;

        }
    }
}

// Tranversal
// preOrder
void preOrder( Node *node = root )
{
    if( !root )
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    else{

```

```

        if( node != NULL ){
            cout << node->label << ", ";
            preOrder(node->left);
            preOrder(node->right);
        }
    }
}

// inOrder
void inOrder( Node *node = root )
{
    if( !root )
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    else{
        if( node != NULL ){
            inOrder(node->left);
            cout << node->label << ", ";
            inOrder(node->right);
        }
    }
}

//postOrder
void postOrder ( Node *node = root )
{
    if( !root )
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    else{
        if( node != NULL ){
            postOrder(node->left);
            postOrder(node->right);
            cout << node->label << ", ";
        }
    }
}

// deleteTree
void deleteTree( Node *node )
{
    if( !root )
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    else{
        if( node != NULL ){

```

```

        if( node != root ){
            node->parent->left = NULL;
            node->parent->right = NULL;
        }
        deleteTree(node->left);
        deleteTree(node->right);

        if( node == root ){
            delete root;
            root = NULL;
        }else{
            delete node;
        }
    }
}

// delete Sub
void deleteSub(Node *node){
    if( !root )
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    else{
        deleteTree(node->left);
        deleteTree(node->right);
        cout << "\nSubtree node " << node->label << " berhasil dihapus." <<
        endl;
    }
}

// clear
void clear(){
    if( !root )
        cout << "\nBuat tree terlebih dahulu!!" << endl;
    else{
        deleteTree(root);
        cout << "\nTree berhasil dihapus." << endl;
    }
}

// size
float size(Node *node = root){
    if( !root ){
        cout << "\nBuat tree terlebih dahulu!!" << endl;

```



```

        return 0;
    }else{
        if( !node ){
            return 0;
        }else{
            return 1 + size( node->left ) + size(node->right);
        }
    }
}

// height
float height( Node *node = root )
{
    if( !root ){
        cout << "\nBuat tree terlebih dahulu!!" << endl;
        return 0;
    }else{
        if( !node ){
            return 0;
        }else{
            float heightKiri = height( node->left );
            float heightKanan = height( node->right );

            if( heightKiri >= heightKanan ){
                return heightKiri + 1;
            }else{
                return heightKanan + 1;
            }
        }
    }
}

// characteristic
void charateristic()
{
    cout << "\nSize Tree : " << size() << endl;
    cout << "Height Tree : " << height() << endl;
    cout << "Average Node of Tree : " << size() / height() << endl;
}

int main()
{
    createNewTree("JAWA");
}

```

```

Node *node2, *node3, *node4, *node5, *node6, *node7, *node8, *node9, *node10;
node2 = insertLeft("KALIMANTAN", root);
node3 = insertRight("SULAWESI", root);
node4 = insertLeft("PAPUA", node2);
node5 = insertRight("MALUKU", node2);
node6 = insertLeft("FLORES", node3);
node7 = insertLeft("TIMOR", node5);
cout << "\nPreOrder :" << endl;
preOrder(root);
cout << "\n" << endl;
cout << "InOrder :" << endl;
inOrder(root);
cout << "\n" << endl;
charateristic();
cout << "\nPreOrder SULAWESI :" << endl;
preOrder(node3);
cout << "\n" << endl;
cout << "InOrder TIMOR :" << endl;
inOrder(node7);
cout << "\n" << endl;
charateristic();
}

```

```

D:\C++\POST TES 2\bin\Debug\POST TES 2.exe
Node FLORES berhasil ditambahkan ke anak kiri SULAWESI
Node TIMOR berhasil ditambahkan ke anak kiri MALUKU

PreOrder :
JAWA, KALIMANTAN, PAPUA, MALUKU, TIMOR, SULAWESI, FLORES,

InOrder :
PAPUA, KALIMANTAN, TIMOR, MALUKU, JAWA, FLORES, SULAWESI,

Size Tree : 7
Height Tree : 4
Average Node of Tree : 1.75

PreOrder SULAWESI :
SULAWESI, FLORES,

InOrder TIMOR :
TIMOR,

Size Tree : 7
Height Tree : 4
Average Node of Tree : 1.75

Process returned 0 (0x0)   execution time : 0.152 s
Press any key to continue.

```