PRAKTIKUM STRUKTUR DATA

UNGUIDED 9

Nama :

Aulia Ahmad Ghaus Adzam (2311104028)

Dosen :

Yudha Islami Sulistya, S.
Kom.,M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

B. UNGUIDED

1. Modifikasi guided tree diatas dengan program menu menggunakan input data tree dari user dan berikan fungsi tambahan untuk menampilkan node child dan descendant dari node yang diinputkan!

```
// Menampilkan Descendant Dan Child nya
void displayDescendants(Pohon *node) {
    if (node == NULL) return;
    cout << node->data << " ";
    displayDescendants(node->left);
    displayDescendants(node->right);
}
```

Fungsi void displayDescendants memiliki parameter Pohon dan Pointer node di dalam nya terdapat kondisi jika node hasil nya NULL maka dia akan return dan akan meng outputkan data keseluruhan dan node left dan right, contoh output nya seperti dibawah jika user memilih opsi no 4

```
Menu:
1. Buat Root Baru
2. Masukan Left Child
3. Masukan Right Child
4. Display Descendants of a Node
5. Cek Valid BST
6. Hitung Leaf Nodes nya
7. Exit
Masukan Pilihan 1-7: 4
Masukan Data Node Yang Ingin Dilihat Descedant Nya: 44
Descendants of node 44: 44
```

2. Buatlah fungsi rekursif is_valid_bst(node, min_val, max_val) untuk memeriksa apakah suatu pohon memenuhi properti Binary Search Tree. Uji fungsi ini pada berbagai pohon, baik yang valid maupun tidak valid sebagai BST.

```
bool isValidBST(__null                val, int max_val) {
    if (node == NULL) return true;
    if (node->data <= min_val || node->data >= max_val) return false;
    return isValidBST(node->left, min_val, node->data) && isValidBST(node->right, node->data, max_val);
}
```

Min_val dan max_val adalah rentang nilai valid untuk node tersebut. Setiap node di kiri harus memiliki nilai lebih kecil dari node-> data. Dan node di kanan harus lebih besar, Contoh Pohon 1 BST nya True, Pohon 2 BST nya False karena node kiri dari root punya anak 15 yang mana lebih besar dari 5 maka output nya akan False/No

```
Menu:
1. Buat Root Baru
2. Masukan Left Child
3. Masukan Right Child
4. Display Descendants of a Node
5. Cek Valid BST
6. Hitung Leaf Nodes nya
7. Exit
Masukan Pilihan 1-7: 5
The tree is a valid BST.
```

3. Buatlah fungsi rekursif cari_simpul_daun(node) untuk menghitung jumlah simpul daun dalam Binary Tree. Simpul daun adalah node yang tidak memiliki anak kiri maupun kanan.

```
int countLeafNodes(Pohon *node) {
    if (node == NULL) return 0;
    if (node->left == NULL && node->right == NULL) return 1;
    return countLeafNodes(node->left) + countLeafNodes(node->right);
}
```

Fungsi ini berfungsikan untuk menghitung ada berapa Leaf Nodes di dalam data Pohon Leaf adalah data yang tidak memiliki Child, contoh suatu parent datanya 7 dan punya child datanya 8 nah child nya ini tidak memiliki child lagi jadi total dari keseluruhan hanya ada 1 Leaf Node yaitu sih 8, contoh Output:

```
Menu:
1. Buat Root Baru
2. Masukan Left Child
3. Masukan Right Child
4. Display Descendants of a Node
5. Cek Valid BST
6. Hitung Leaf Nodes nya
7. Exit
Masukan Pilihan 1-7: 6
Total leaf nodes: 1
```

4. KODE SECARA KESULURUHAN

```cpp
#include <iostream>
#include <limits>
using namespace std;

// DEKLARASI
struct Pohon {
    int data;
    Pohon *left, *right, *parent;
    Pohon(int value) : data(value), left(NULL), right(NULL), parent(NULL) {}
};

Pohon *root = NULL;

// Inisialisasi
void init() {
    root = NULL;
}

// Cek Pohon isEmpty?
bool isEmpty() {
    return root == NULL;
}

// Buat Node Baru
void createNode(int data) {
    if (isEmpty()) {
        root = new Pohon(data);
        cout << "Node " << data << " -> Root." << endl;
    } else {
        cout << "Data Di Dalam Pohon Sudah Dibuat" << endl;
    }
}

// Masukan Eelemen Kiri
Pohon* insertLeft(int data, Pohon *node) {
    if (isEmpty()) {
        cout << "Pohon Masih Kosong! Buat Dulu" << endl;
        return NULL;
    }
    if (node->left != NULL) {
        cout << "Node " << node->data << " Sudah Memiliki Child Left" << endl;
        return NULL;
    }
    Pohon *newNode = new Pohon(data);
    node->left = newNode;
    newNode->parent = node;
    cout << "Node " << data << " Telah Ditambahkan Sebagai Child dari " << node->data << endl;
    return newNode;
}

// Masukan Eelemen Kanan
Pohon* insertRight(int data, Pohon *node) {
    if (isEmpty()) {
        cout << "Pohon Masih Kosong! Buat Dulu" << endl;
        return NULL;
    }
    if (node->right != NULL) {
        cout << "Node " << node->data << " Sudah Memiliki Child Right" << endl;
        return NULL;
    }
    Pohon *newNode = new Pohon(data);
    node->right = newNode;
    newNode->parent = node;
    cout << "Node " << data << " Telah Ditambahkan Sebagai Child dari " << node->data << endl;
    return newNode;
}

// Menampilkan Descendant Dan Child nya
void displayDescendants(Pohon *node) {
    if (node == NULL) return;
    cout << node->data << " ";
```