

LAPORAN PRAKTIKUM

MODUL 9

TREE



Nama :

Novita Syahwa Tri Hapsari (2311104007)

Dosen :

Yudha Islami Sulistya,S.Kom,M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

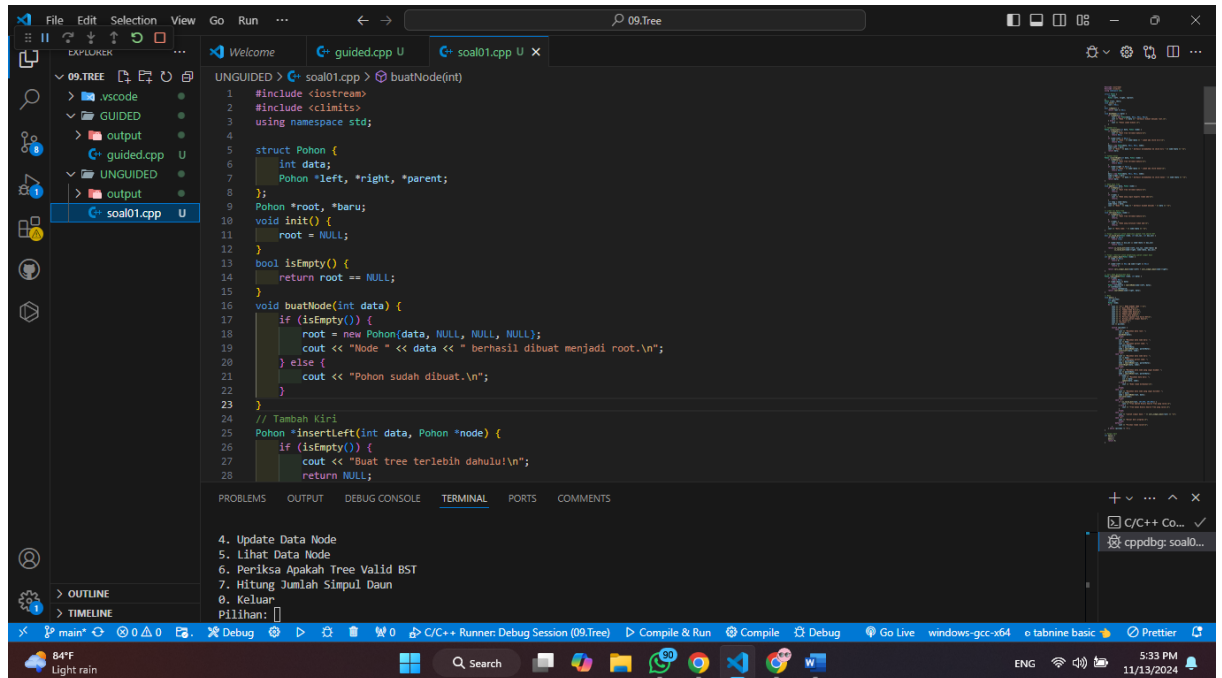
FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

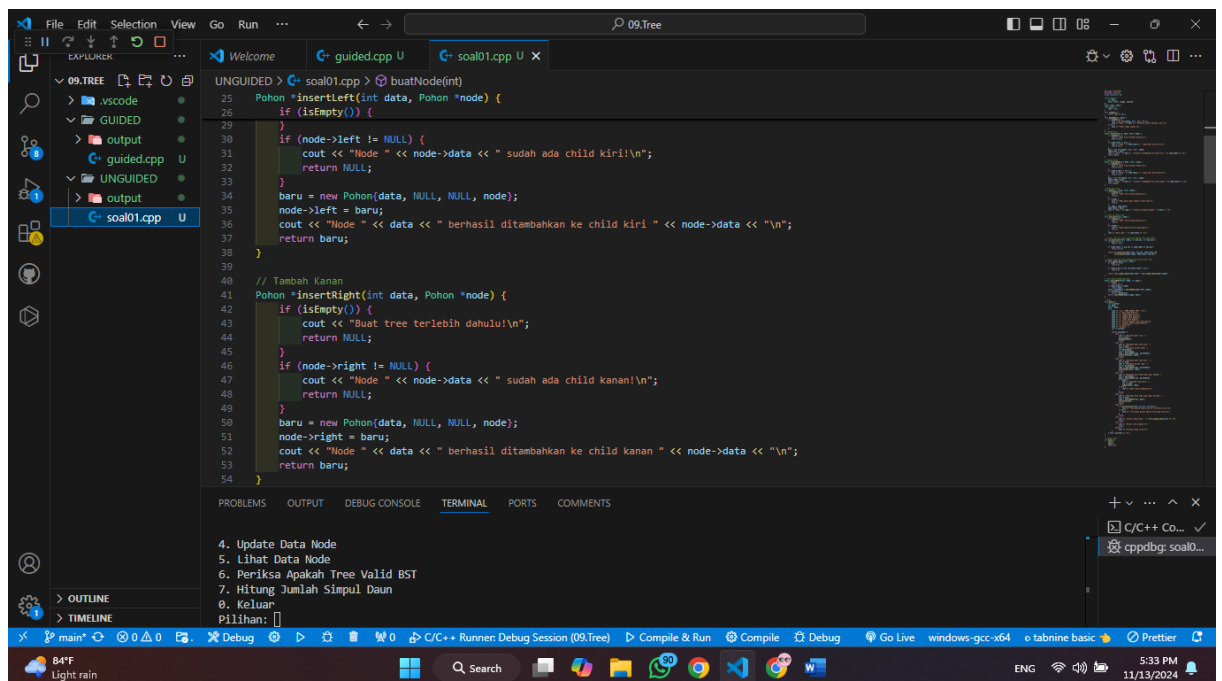
UNGUIDED

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!



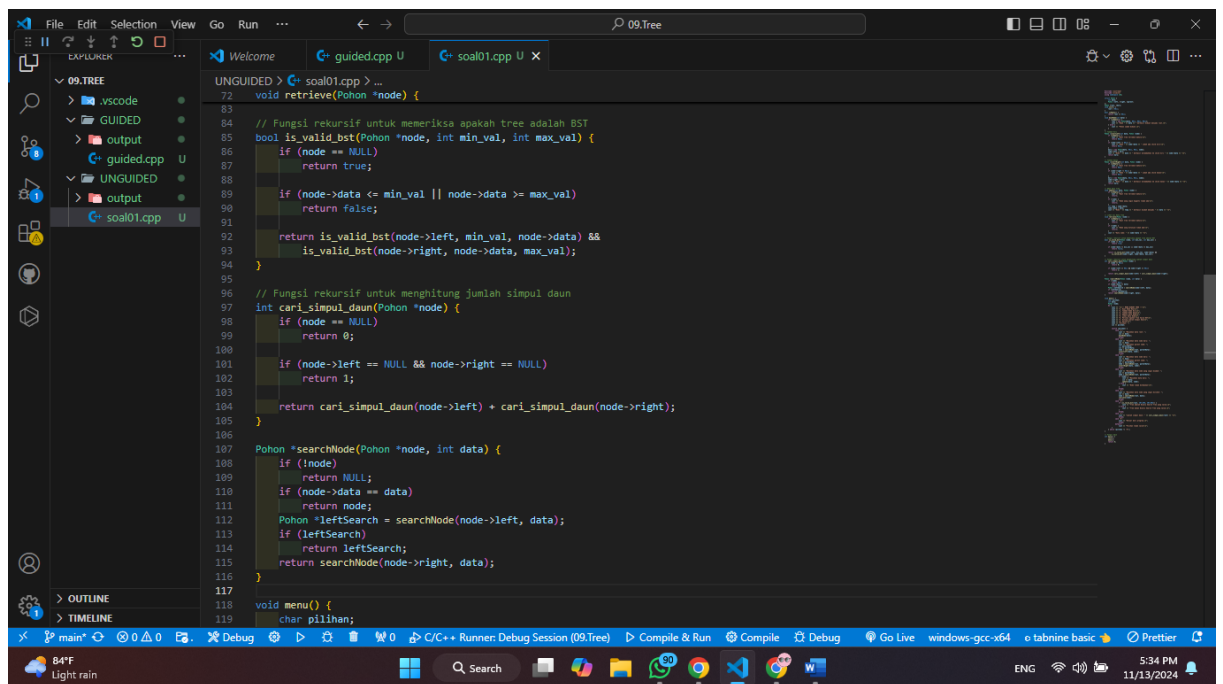
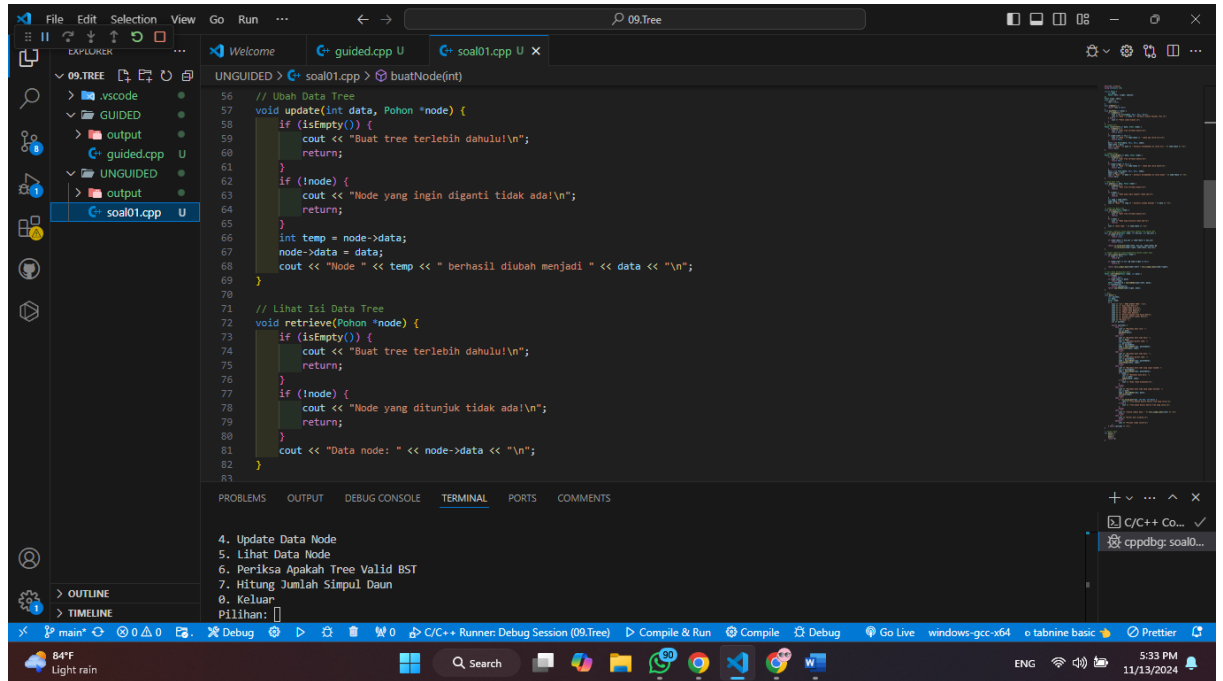
```
1 #include <iostream>
2 #include <limits>
3 using namespace std;
4
5 struct Pohon {
6     int data;
7     Pohon *left, *right, *parent;
8 };
9 Pohon *root, *baru;
10 void init() {
11     root = NULL;
12 }
13 bool isEmpty() {
14     return root == NULL;
15 }
16 void buatNode(int data) {
17     if (isEmpty()) {
18         root = new Pohon(data, NULL, NULL, NULL);
19         cout << "Node " << data << " berhasil dibuat menjadi root.\n";
20     } else {
21         cout << "Pohon sudah dibuat.\n";
22     }
23 }
24 // Tambah Kiri
25 Pohon *insertLeft(int data, Pohon *node) {
26     if (isEmpty()) {
27         cout << "Buat tree terlebih dahulu!\n";
28     }
29 }
```

4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
8. Keluar
Pilihan: []



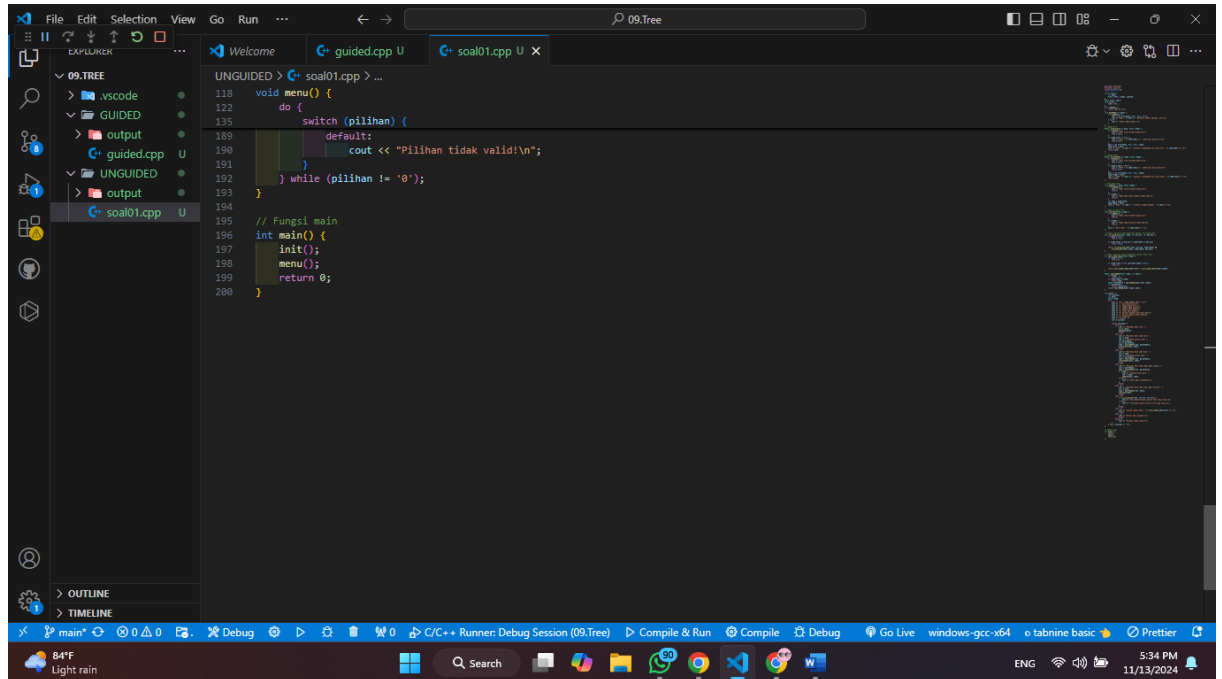
```
25 Pohon *insertLeft(int data, Pohon *node) {
26     if (isEmpty()) {
27     }
28     if (node->left != NULL) {
29         cout << "Node " << node->data << " sudah ada child kiri!\n";
30         return NULL;
31     }
32     baru = new Pohon(data, NULL, NULL, node);
33     node->left = baru;
34     cout << "Node " << data << " berhasil ditambahkan ke child kiri " << node->data << "\n";
35     return baru;
36 }
37
38 // Tambah Kanan
39 Pohon *insertRight(int data, Pohon *node) {
40     if (isEmpty()) {
41         cout << "Buat tree terlebih dahulu!\n";
42         return NULL;
43     }
44     if (node->right != NULL) {
45         cout << "Node " << node->data << " sudah ada child kanan!\n";
46         return NULL;
47     }
48     baru = new Pohon(data, NULL, NULL, node);
49     node->right = baru;
50     cout << "Node " << data << " berhasil ditambahkan ke child kanan " << node->data << "\n";
51     return baru;
52 }
53
54 }
```

4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
8. Keluar
Pilihan: []



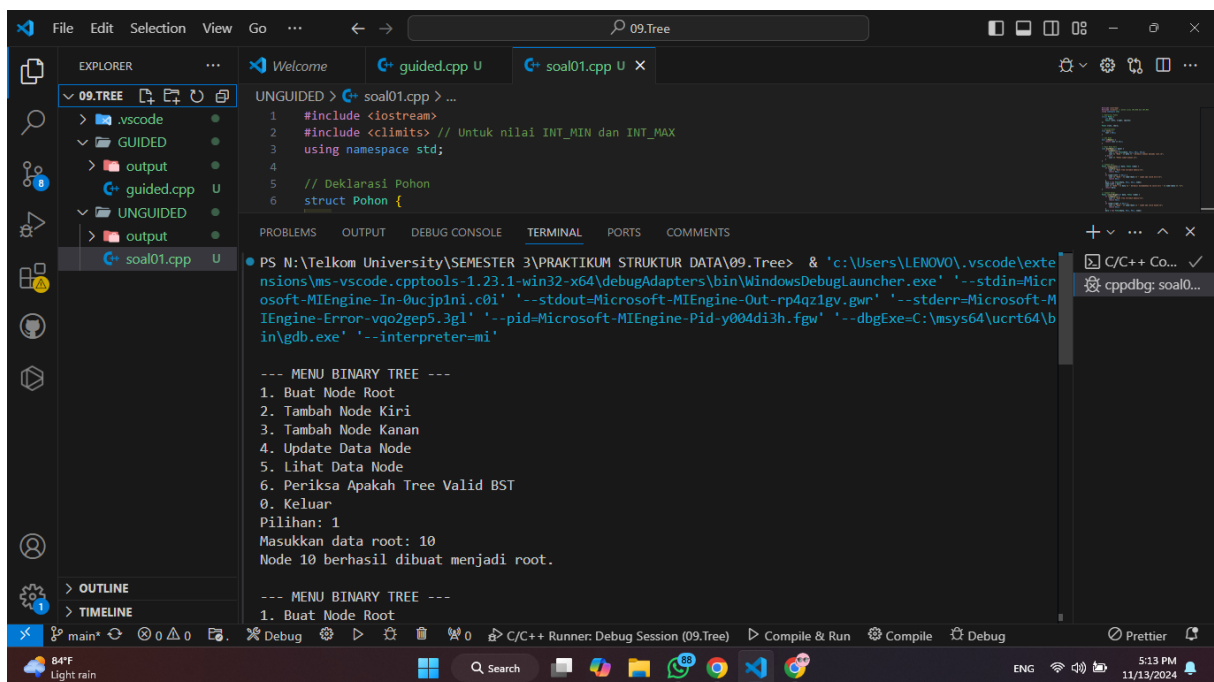
```
118 void menu() {
119     char pilihan;
120     int data;
121     Pohon *node;
122     do {
123         cout << "\n--- MENU BINARY TREE ---\n";
124         cout << "1. Buat Node Root\n";
125         cout << "2. Tambah Node Kiri\n";
126         cout << "3. Tambah Node Kanan\n";
127         cout << "4. Update Data Node\n";
128         cout << "5. Lihat Data Node\n";
129         cout << "6. Periksa Apakah Tree Valid BST\n";
130         cout << "7. Hitung Jumlah Simpul Daun\n";
131         cout << "8. Keluar\n";
132         cout << "Pilihan: ";
133         cin >> pilihan;
134
135         switch (pilihan) {
136             case '1':
137                 cout << "Masukkan data root: ";
138                 cin >> data;
139                 buatNode(data);
140                 break;
141             case '2':
142                 cout << "Masukkan data node baru: ";
143                 cin >> data;
144                 cout << "Masukkan parent node: ";
145                 int parentData;
146                 cin >> parentData;
147                 node = searchNode(root, parentData);
148                 insertLeft(data, node);
149                 break;
150             case '3':
151                 cout << "Masukkan data node baru: ";
152                 cin >> data;
153                 cout << "Masukkan parent node: ";
154                 cin >> parentData;
155                 node = searchNode(root, parentData);
```

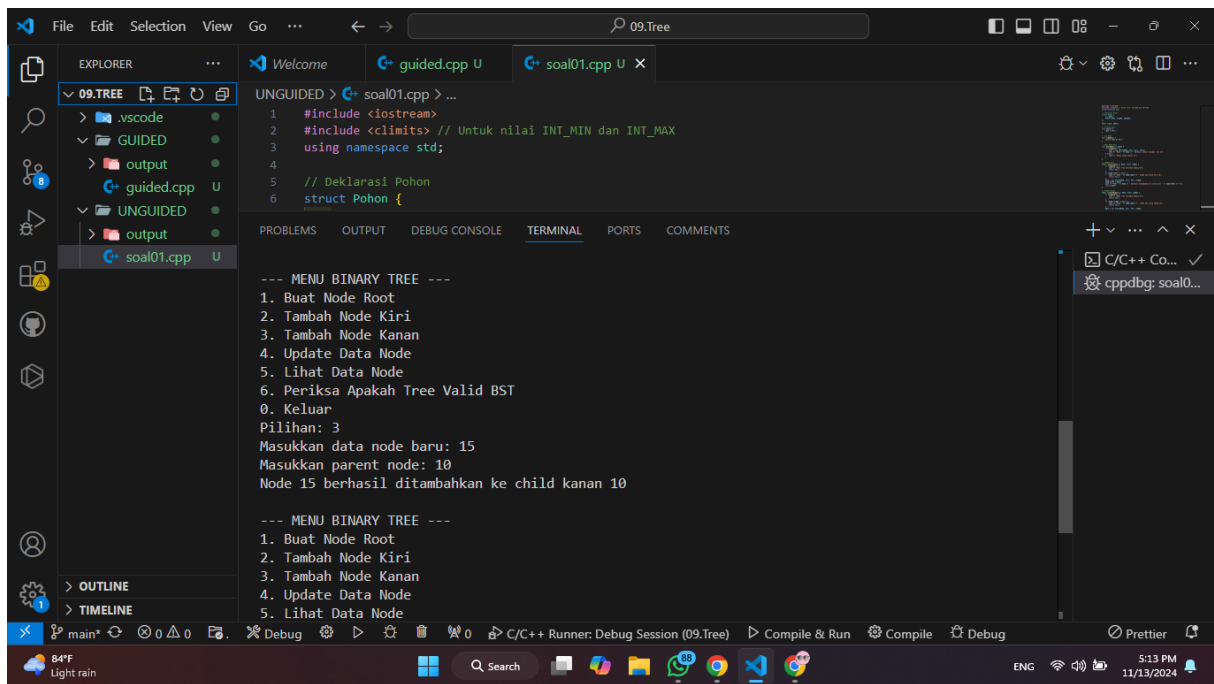
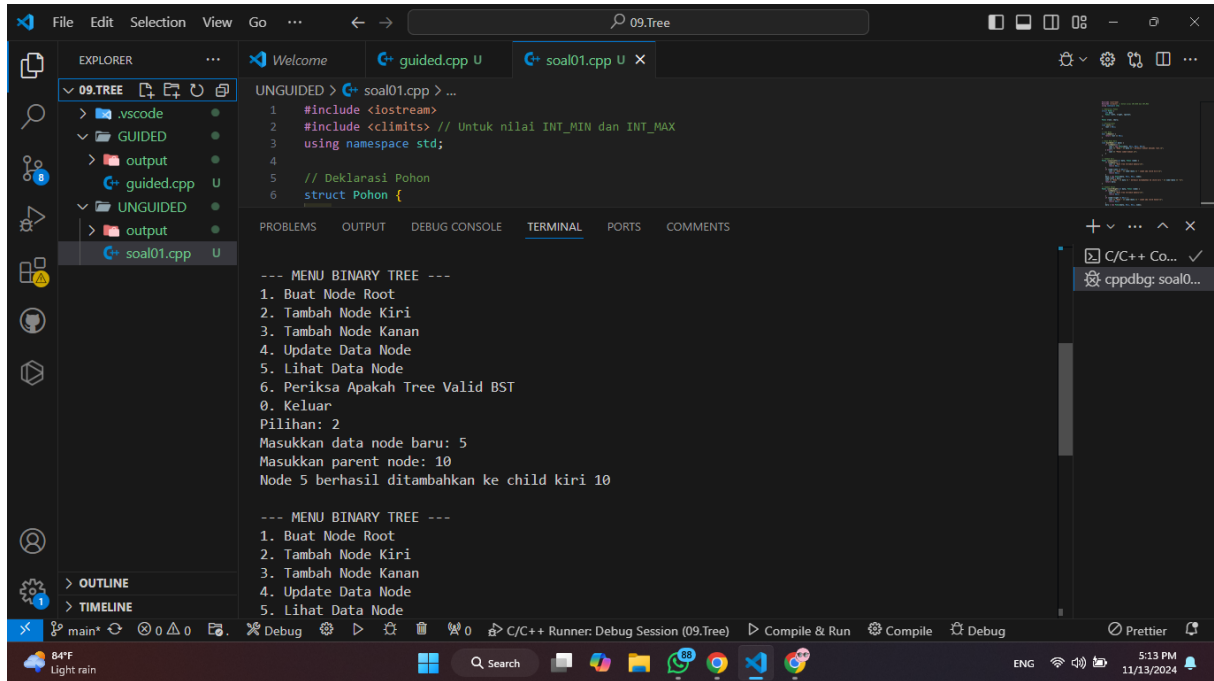
```
155         node = searchNode(root, parentData);
156         insertRight(data, node);
157         break;
158     case '4':
159         cout << "Masukkan data node yang ingin diubah: ";
160         cin >> parentData;
161         node = searchNode(root, parentData);
162         if (node) {
163             cout << "Masukkan data baru: ";
164             cin >> data;
165             update(data, node);
166         } else {
167             cout << "Node tidak ditemukan!\n";
168         }
169         break;
170     case '5':
171         cout << "Masukkan data node yang ingin dilihat: ";
172         cin >> data;
173         node = searchNode(root, data);
174         retrieve(node);
175         break;
176     case '6':
177         if (is_valid_bst(root, INT_MIN, INT_MAX)) {
178             cout << "Tree adalah Binary Search Tree yang valid.\n";
179         } else {
180             cout << "Tree bukan Binary Search Tree yang valid.\n";
181         }
182         break;
183     case '7':
184         cout << "Jumlah simpul daun: " << cari_simpul_daun(root) << "\n";
185         break;
186     case '8':
187         cout << "Keluar dari program.\n";
188         break;
189     default:
```

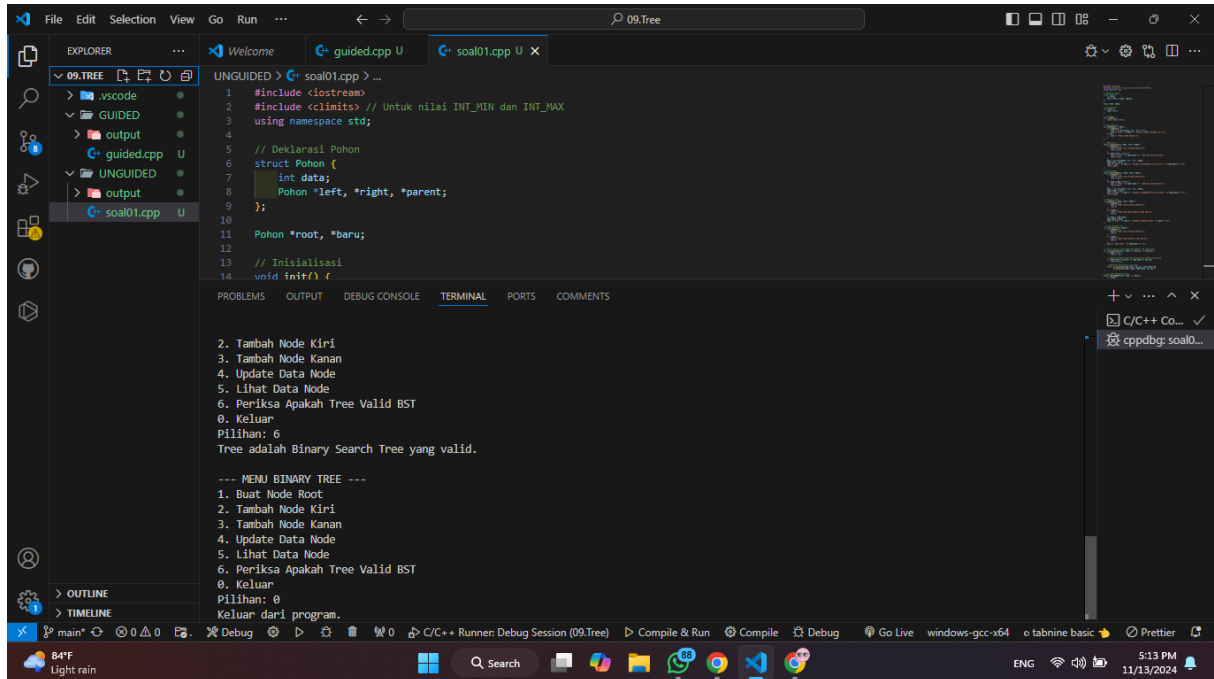


2. Buatlah fungsi rekursif `is_valid_bst(node, min_val, max_val)` untuk memeriksa apakah suatu pohon memenuhi properti Binary Search Tree. Uji fungsi ini pada berbagai pohon, baik yang valid maupun tidak valid sebagai BST.

#Pohon yan valid







#pohon yang tidak valid

```

--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 1
Masukkan data root: 10
Node 10 berhasil dibuat menjadi root.

```

```

--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 2
Masukkan data node baru: 15
Masukkan parent node: 10
Node 15 berhasil ditambahkan ke child kiri 10

```

```
--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 3
Masukkan data node baru: 5
Masukkan parent node: 10
Node 5 berhasil ditambahkan ke child kanan 10
```

```
--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 6
Tree bukan Binary Search Tree yang valid.
```

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.


```
--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 1
Masukkan data root: 10
Node 10 berhasil dibuat menjadi root.

--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 2
Masukkan data node baru: 5
Masukkan parent node: 10
Node 5 berhasil ditambahkan ke child kiri 10
```

```
--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 3
Masukkan data node baru: 15
Masukkan parent node: 10
Node 15 berhasil ditambahkan ke child kanan 10

--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 2
Masukkan data node baru: 2
Masukkan parent node: 5
Node 2 berhasil ditambahkan ke child kiri 5
```

```
--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 3
Masukkan data node baru: 7
Masukkan parent node: 5
Node 7 berhasil ditambahkan ke child kanan 5

--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 3
Masukkan data node baru: 20
Masukkan parent node: 15
Node 20 berhasil ditambahkan ke child kanan 15
```

```
--- MENU BINARY TREE ---
1. Buat Node Root
2. Tambah Node Kiri
3. Tambah Node Kanan
4. Update Data Node
5. Lihat Data Node
6. Periksa Apakah Tree Valid BST
7. Hitung Jumlah Simpul Daun
0. Keluar
Pilihan: 7
Jumlah simpul daun: 3
```