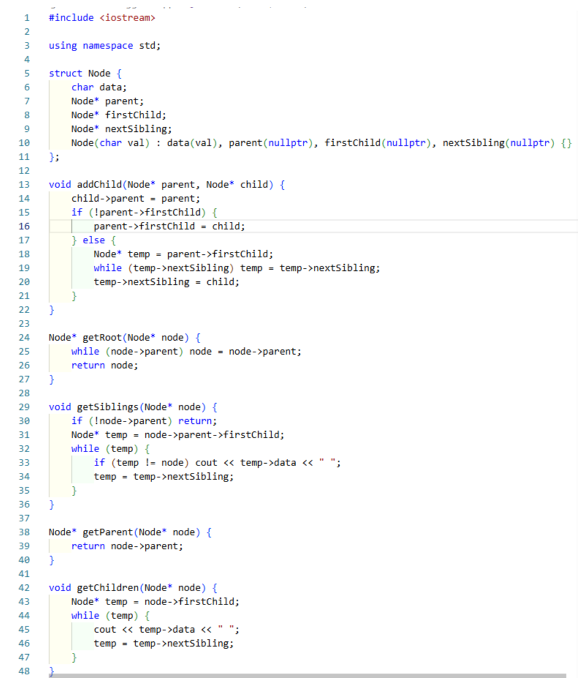
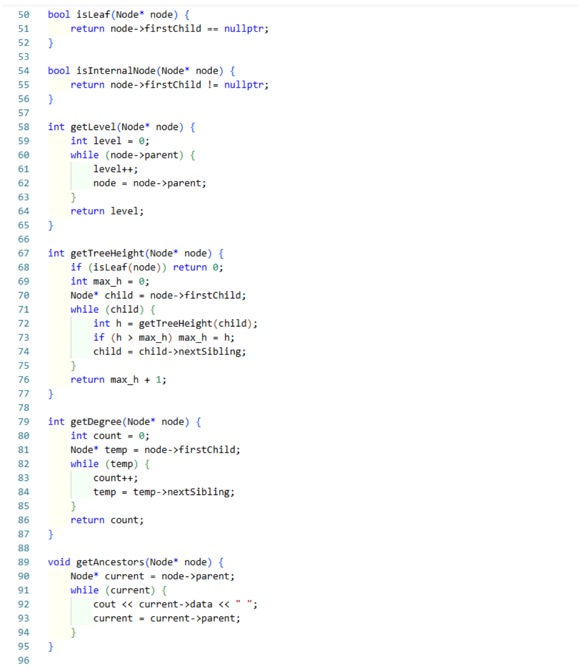
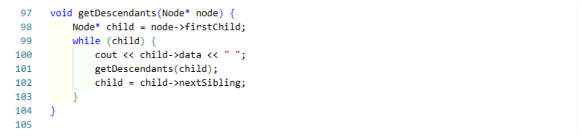


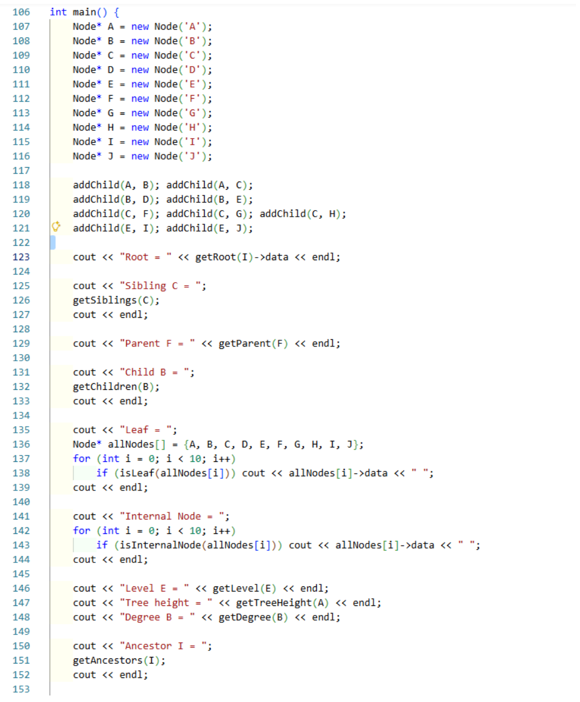
```cpp
#include <iostream>

using namespace std;

struct Node {
    char data;
    Node* parent;
    Node* firstChild;
    Node* nextSibling;
    Node(char val) : data(val), parent(nullptr), firstChild(nullptr), nextSibling(nullptr) {}
};

void addChild(Node* parent, Node* child) {
    child->parent = parent;
    if (!parent->firstChild) {
        parent->firstChild = child;
    } else {
        Node* temp = parent->firstChild;
        while (temp->nextSibling) temp = temp->nextSibling;
        temp->nextSibling = child;
    }
}

Node* getRoot(Node* node) {
    while (node->parent) node = node->parent;
    return node;
}

void getSiblings(Node* node) {
    if (!node->parent) return;
    Node* temp = node->parent->firstChild;
    while (temp) {
        if (temp != node) cout << temp->data << " ";
        temp = temp->nextSibling;
    }
}

Node* getParent(Node* node) {
    return node->parent;
}

void getChildren(Node* node) {
    Node* temp = node->firstChild;
    while (temp) {
        cout << temp->data << " ";
        temp = temp->nextSibling;
    }
}
```



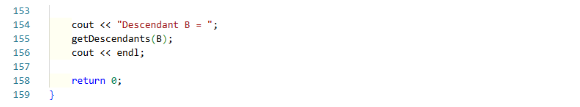
```cpp
bool isLeaf(Node* node) {
    return node->firstChild == nullptr;
}

bool isInternalNode(Node* node) {
    return node->firstChild != nullptr;
}

int getLevel(Node* node) {
    int level = 0;
    while (node->parent) {
        level++;
        node = node->parent;
    }
    return level;
}

int getTreeHeight(Node* node) {
    if (isLeaf(node)) return 0;
    int max_h = 0;
    Node* child = node->firstChild;
    while (child) {
        int h = getTreeHeight(child);
        if (h > max_h) max_h = h;
        child = child->nextSibling;
    }
    return max_h + 1;
}

int getDegree(Node* node) {
    int count = 0;
    Node* temp = node->firstChild;
    while (temp) {
        count++;
        temp = temp->nextSibling;
    }
    return count;
}

void getAncestors(Node* node) {
    Node* current = node->parent;
    while (current) {
        cout << current->data << " ";
        current = current->parent;
    }
}

```



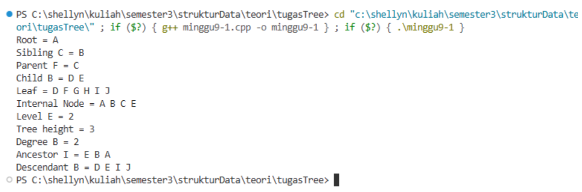
```cpp
void getDescendants(Node* node) {
    Node* child = node->firstChild;
    while (child) {
        cout << child->data << " ";
        getDescendants(child);
        child = child->nextSibling;
    }
}

```



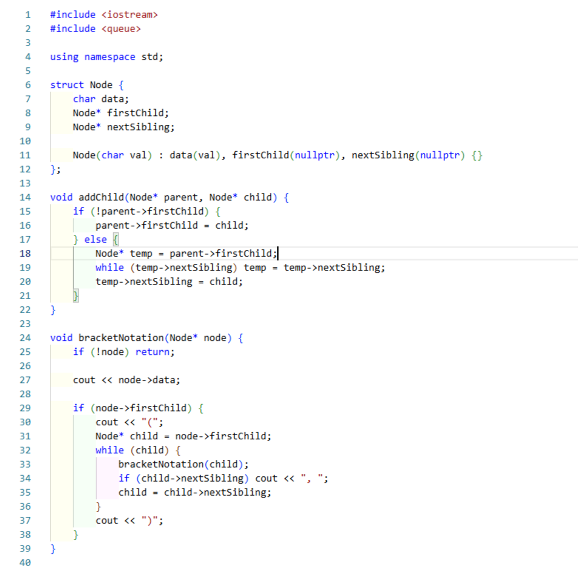
```cpp
int main() {
    Node* A = new Node('A');
    Node* B = new Node('B');
    Node* C = new Node('C');
    Node* D = new Node('D');
    Node* E = new Node('E');
    Node* F = new Node('F');
    Node* G = new Node('G');
    Node* H = new Node('H');
    Node* I = new Node('I');
    Node* J = new Node('J');

    addChild(A, B); addChild(A, C);
    addChild(B, D); addChild(B, E);
    addChild(C, F); addChild(C, G); addChild(C, H);
    addChild(E, I); addChild(E, J);

    cout << "Root = " << getRoot(I)->data << endl;

    cout << "Sibling C = ";
    getSiblings(C);
    cout << endl;

    cout << "Parent F = " << getParent(F) << endl;

    cout << "Child B = ";
    getChildren(B);
    cout << endl;

    cout << "Leaf = ";
    Node* allNodes[] = {A, B, C, D, E, F, G, H, I, J};
    for (int i = 0; i < 10; i++)
        if (isLeaf(allNodes[i])) cout << allNodes[i]->data << " ";
    cout << endl;

    cout << "Internal Node = ";
    for (int i = 0; i < 10; i++)
        if (isInternalNode(allNodes[i])) cout << allNodes[i]->data << " ";
    cout << endl;

    cout << "Level E = " << getLevel(E) << endl;
    cout << "Tree height = " << getTreeHeight(A) << endl;
    cout << "Degree B = " << getDegree(B) << endl;

    cout << "Ancestor I = ";
    getAncestors(I);
    cout << endl;

```



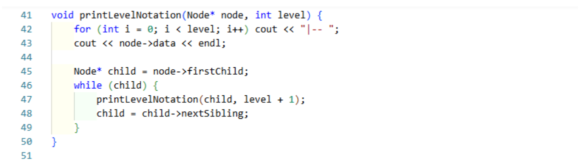
```cpp

    cout << "Descendant B = ";
    getDescendants(B);
    cout << endl;

    return 0;
}
```



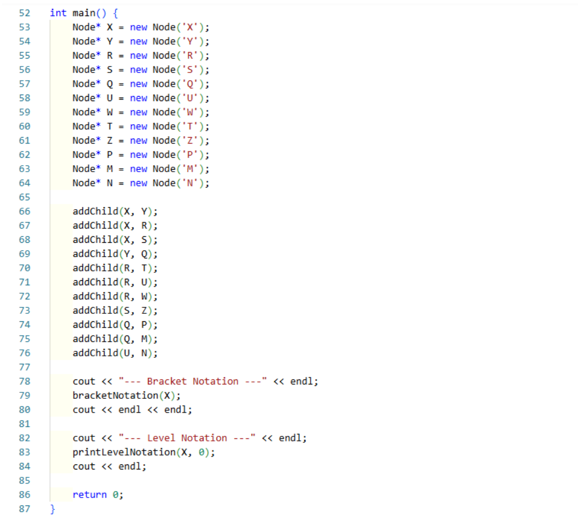
```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuliah\semester3\strukturData\te
ori\tugasTree\" ; if ($?) { g++ minggu9-1.cpp -o minggu9-1 } ; if ($?) { .\minggu9-1 }
Root = A
Sibling C = B
Parent F = C
Child B = D E
Leaf = D F G H I J
Internal Node = A B C E
Level E = 2
Tree height = 3
Degree B = 2
Ancestor I = E B A
Descendant B = D E I J
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> 
```



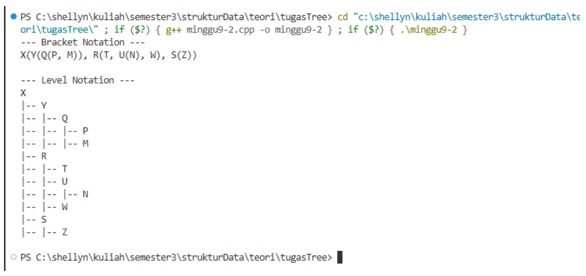
```cpp
#include <iostream>
#include <queue>

using namespace std;

struct Node {
    char data;
    Node* firstChild;
    Node* nextSibling;

    Node(char val) : data(val), firstChild(nullptr), nextSibling(nullptr) {}
};

void addChild(Node* parent, Node* child) {
    if (!parent->firstChild) {
        parent->firstChild = child;
    } else {
        Node* temp = parent->firstChild;
        while (temp->nextSibling) temp = temp->nextSibling;
        temp->nextSibling = child;
    }
}

void bracketNotation(Node* node) {
    if (!node) return;

    cout << node->data;

    if (node->firstChild) {
        cout << "(";
        Node* child = node->firstChild;
        while (child) {
            bracketNotation(child);
            if (child->nextSibling) cout << ", ";
            child = child->nextSibling;
        }
        cout << ")";
    }
}

```



```cpp
void printLevelNotation(Node* node, int level) {
    for (int i = 0; i < level; i++) cout << "|-- ";
    cout << node->data << endl;

    Node* child = node->firstChild;
    while (child) {
        printLevelNotation(child, level + 1);
        child = child->nextSibling;
    }
}

```



```cpp
int main() {
    Node* X = new Node('X');
    Node* Y = new Node('Y');
    Node* R = new Node('R');
    Node* S = new Node('S');
    Node* Q = new Node('Q');
    Node* U = new Node('U');
    Node* W = new Node('W');
    Node* T = new Node('T');
    Node* Z = new Node('Z');
    Node* P = new Node('P');
    Node* M = new Node('M');
    Node* N = new Node('N');

    addChild(X, Y);
    addChild(X, R);
    addChild(X, S);
    addChild(Y, Q);
    addChild(R, T);
    addChild(R, U);
    addChild(R, W);
    addChild(S, Z);
    addChild(Q, P);
    addChild(Q, M);
    addChild(U, N);

    cout << "--- Bracket Notation ---" << endl;
    bracketNotation(X);
    cout << endl << endl;

    cout << "--- Level Notation ---" << endl;
    printLevelNotation(X, 0);
    cout << endl;

    return 0;
}
```



```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuliah\semester3\strukturData\te
ori\tugasTree\" ; if ($?) { g++ minggu9-2.cpp -o minggu9-2 } ; if ($?) { .\minggu9-2 }
--- Bracket Notation ---
X(Y(Q(P, M)), R(T, U(N), W), S(Z))

--- Level Notation ---
X
|-- Y
|-- |-- Q
|-- |-- |-- P
|-- |-- |-- M
|-- R
|-- |-- T
|-- |-- U
|-- |-- |-- N
|-- |-- W
|-- S
|-- |-- Z

PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> 
```