

```

1 #include <iostream>
2
3 using namespace std;
4
5 struct Node {
6     char data;
7     Node* parent;
8     Node* firstChild;
9     Node* nextSibling;
10    Node(char val) : data(val), parent(nullptr), firstChild(nullptr), nextSibling(nullptr) {}
11 };
12
13 void addChild(Node* parent, Node* child) {
14     child->parent = parent;
15     if (!parent->firstChild) {
16         parent->firstChild = child;
17     } else {
18         Node* temp = parent->firstChild;
19         while (temp->nextSibling) temp = temp->nextSibling;
20         temp->nextSibling = child;
21     }
22 }
23
24 Node* getRoot(Node* node) {
25     while (node->parent) node = node->parent;
26     return node;
27 }
28
29 void getSiblings(Node* node) {
30     if (!node->parent) return;
31     Node* temp = node->parent->firstChild;
32     while (temp) {
33         if (temp != node) cout << temp->data << " ";
34         temp = temp->nextSibling;
35     }
36 }
37
38 Node* getParent(Node* node) {
39     return node->parent;
40 }
41
42 void getChildren(Node* node) {
43     Node* temp = node->firstChild;
44     while (temp) {
45         cout << temp->data << " ";
46         temp = temp->nextSibling;
47     }
48 }
49
50 bool isLeaf(Node* node) {
51     return node->firstChild == nullptr;
52 }
53
54 bool isInternalNode(Node* node) {
55     return node->firstChild != nullptr;
56 }
57
58 int getLevel(Node* node) {
59     int level = 0;
60     while (node->parent) {
61         level++;
62         node = node->parent;
63     }
64     return level;
65 }
66
67 int getTreeHeight(Node* node) {
68     if (!isLeaf(node)) return 0;
69     int max_h = 0;
70     Node* child = node->firstChild;
71     while (child) {
72         int h = getTreeHeight(child);
73         if (h > max_h) max_h = h;
74         child = child->nextSibling;
75     }
76     return max_h + 1;
77 }
78
79 int getDegree(Node* node) {
80     int count = 0;
81     Node* temp = node->firstChild;
82     while (temp) {
83         count++;
84         temp = temp->nextSibling;
85     }
86     return count;
87 }
88
89 void getAncestors(Node* node) {
90     Node* current = node->parent;
91     while (current) {
92         cout << current->data << " ";
93         current = current->parent;
94     }
95 }
96
97 void getDescendants(Node* node) {
98     Node* child = node->firstChild;
99     while (child) {
100         cout << child->data << " ";
101         getDescendants(child);
102         child = child->nextSibling;
103     }
104 }
105

```

```

1 int main() {
2     Node* A = new Node('A');
3     Node* B = new Node('B');
4     Node* C = new Node('C');
5     Node* D = new Node('D');
6     Node* E = new Node('E');
7     Node* F = new Node('F');
8     Node* G = new Node('G');
9     Node* H = new Node('H');
10    Node* I = new Node('I');
11    Node* J = new Node('J');
12
13    addChild(A, B); addChild(A, C);
14    addChild(B, D); addChild(B, E);
15    addChild(C, F); addChild(C, G); addChild(C, H);
16    addChild(E, I); addChild(E, J);
17
18    cout << "Root = " << getRoot(I)->data << endl;
19
20    cout << "Sibling C = ";
21    getsiblings(C);
22    cout << endl;
23
24    cout << "Parent F = " << getParent(F) << endl;
25
26    cout << "Child B = ";
27    getChildren(B);
28    cout << endl;
29
30    cout << "Leaf = ";
31    Node* allNodes[] = {A, B, C, D, E, F, G, H, I, J};
32    for (int i = 0; i < 10; i++)
33        if (!isleaf(allNodes[i])) cout << allNodes[i]->data << " ";
34    cout << endl;
35
36    cout << "Internal Node = ";
37    for (int i = 0; i < 10; i++)
38        if (isInternalNode(allNodes[i])) cout << allNodes[i]->data << " ";
39    cout << endl;
40
41    cout << "Level E = " << getLevel(E) << endl;
42    cout << "Tree height = " << getTreeHeight(A) << endl;
43    cout << "Degree B = " << getDegree(B) << endl;
44
45    cout << "Ancestor I = ";
46    getAncestors(I);
47    cout << endl;
48
49
50    cout << "Descendant B = ";
51    getDescendants(B);
52    cout << endl;
53
54    return 0;
55 }

```

PS C:\shellyn\kuallah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuallah\semester3\strukturData\teori\tugasTree" & 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 O I J  
 PS C:\shellyn\kuallah\semester3\strukturData\teori\tugasTree>

```

1  #include <iostream>
2  #include <queue>
3
4  using namespace std;
5
6  struct Node {
7      char data;
8      Node* firstChild;
9      Node* nextSibling;
10 }
11 Node(char val) : data(val), firstChild(nullptr), nextSibling(nullptr) {}
12 }
13
14 void addChild(Node* parent, Node* child) {
15     if (!parent->firstChild)
16         parent->firstChild = child;
17     else {
18         Node* temp = parent->firstChild;
19         while (temp->nextSibling) temp = temp->nextSibling;
20         temp->nextSibling = child;
21     }
22 }
23
24 void bracketNotation(Node* node) {
25     if (!node) return;
26
27     cout << node->data;
28
29     if (node->firstChild) {
30         cout << "(";
31         Node* child = node->firstChild;
32         while (child) {
33             bracketNotation(child);
34             if (child->nextSibling) cout << ", ";
35             child = child->nextSibling;
36         }
37         cout << ")";
38     }
39 }
40
41 void printLevelNotation(Node* node, int level) {
42     for (int i = 0; i < level; i++) cout << "|-- ";
43     cout << node->data << endl;
44
45     Node* child = node->firstChild;
46     while (child) {
47         printLevelNotation(child, level + 1);
48         child = child->nextSibling;
49     }
50 }
51
52 int main() {
53     Node* X = new Node('X');
54     Node* Y = new Node('Y');
55     Node* R = new Node('R');
56     Node* S = new Node('S');
57     Node* Q = new Node('Q');
58     Node* U = new Node('U');
59     Node* W = new Node('W');
60     Node* T = new Node('T');
61     Node* Z = new Node('Z');
62     Node* P = new Node('P');
63     Node* M = new Node('M');
64     Node* N = new Node('N');
65
66     addChild(X, Y);
67     addChild(X, R);
68     addChild(X, S);
69     addChild(Y, Q);
70     addChild(R, T);
71     addChild(R, U);
72     addChild(R, W);
73     addChild(S, Z);
74     addChild(S, P);
75     addChild(Q, M);
76     addChild(U, N);
77
78     cout << "... Bracket Notation ---" << endl;
79     bracketNotation(X);
80     cout << endl << endl;
81
82     cout << "... Level Notation ---" << endl;
83     printLevelNotation(X, 0);
84     cout << endl;
85
86     return 0;
87 }

```

```

PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuliah\semester3\strukturData\tk
or\tugasTree"; if ($?) { g++ minggu-2.cpp -o minggu-2 } ; if ($?) { ./minggu-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> 
```