

```

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

```

```

106 int main() {
107     Node* A = new Node('A');
108     Node* B = new Node('B');
109     Node* C = new Node('C');
110     Node* D = new Node('D');
111     Node* E = new Node('E');
112     Node* F = new Node('F');
113     Node* G = new Node('G');
114     Node* H = new Node('H');
115     Node* I = new Node('I');
116     Node* J = new Node('J');
117
118     addChild(A, B); addChild(A, C);
119     addChild(B, D); addChild(B, E);
120     addChild(C, F); addChild(C, G); addChild(C, H);
121     addChild(E, I); addChild(E, J);
122
123     cout << "Root = " << getRoot(I)->data << endl;
124
125     cout << "Sibling C = ";
126     getSiblings(C);
127     cout << endl;
128
129     cout << "Parent F = " << getParent(F) << endl;
130
131     cout << "Child B = ";
132     getChildren(B);
133     cout << endl;
134
135     cout << "Leaf = ";
136     Node* allNodes[] = {A, B, C, D, E, F, G, H, I, J};
137     for (int i = 0; i < 10; i++)
138         if (isLeaf(allNodes[i])) cout << allNodes[i]->data << " ";
139     cout << endl;
140
141     cout << "Internal Node = ";
142     for (int i = 0; i < 10; i++)
143         if (isInternalNode(allNodes[i])) cout << allNodes[i]->data << " ";
144     cout << endl;
145
146     cout << "Level E = " << getLevel(E) << endl;
147     cout << "Tree height = " << getTreeHeight(A) << endl;
148     cout << "Degree B = " << getDegree(B) << endl;
149
150     cout << "Ancestor I = ";
151     getAncestors(I);
152     cout << endl;
153
154     cout << "Descendant B = ";
155     getDescendants(B);
156     cout << endl;
157     return 0;
158 }
159

```

• PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree\"; if (\$?) { g++ minggu0-1.cpp -o minggu0-1 }; if (\$?) { .\minggu0-1 }

```

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>

```

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(Q, 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\teori\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>

```