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

using namespace std;

struct Node {
    char info;
    Node* left;
    Node* right;
};

Node* createNode(char data) {
    Node* newNode = new Node();
    newNode->info = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

void preOrder(Node* root) {
    if (root != NULL) {
        cout << root->info << " ";
        preOrder(root->left);
        preOrder(root->right);
    }
}

void inOrder(Node* root) {
    if (root != NULL) {
        inOrder(root->left);
        cout << root->info << " ";
        inOrder(root->right);
    }
}


void postOrder(Node* root) {
    if (root != NULL) {
        postOrder(root->left);
        postOrder(root->right);
        cout << root->info << " ";
    }
}

void levelOrder(Node* root) {
    if (root == NULL) return;

    queue<Node*> q;
    q.push(root);

    while (!q.empty()) {
        Node* current = q.front();
        q.pop();

        cout << current->info << " ";

        if (current->left != NULL) q.push(current->left);
        if (current->right != NULL) q.push(current->right);
    }
}

```

Exercise 1

```cpp
int main() {
    Node* root = createNode('/');

    root->left = createNode('+');
    root->right = createNode('g');

    root->left->left = createNode('*');
    root->left->right = createNode('^');

    root->left->left->left = createNode('+');
    root->left->left->right = createNode('/');

    root->left->right->left = createNode('e');
    root->left->right->right = createNode('f');

    root->left->left->left->left = createNode('a');
    root->left->left->left->right = createNode('b');

    root->left->left->right->left = createNode('c');
    root->left->left->right->right = createNode('d');

    cout << "\n1. Pre-order (Root-Left-Right):" << endl;
    preOrder(root);

    cout << "\n\n2. In-order (Left-Root-Right):" << endl;
    inOrder(root);

    cout << "\n\n3. Post-order (Left-Right-Root):" << endl;
    postOrder(root);

    cout << "\n\n4. Level-order (BFS):" << endl;
    levelOrder(root);

    cout << endl;
    return 0;
}
```

Output

```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuliah\semester3\strukturData\teori\tugasTree\" ; if ($?) { g++ exercise1.cpp -o exercise1 } ; if ($?) { .\exercise1 }

1. Pre-order (Root-Left-Right):
/ + * + a b / c d ^ e f g

2. In-order (Left-Root-Right):
a + b * c / d + e ^ f / g

3. Post-order (Left-Right-Root):
a b + c d / * e f ^ + g /

4. Level-order (BFS):
/ + g * ^ + / e f a b c d
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree>
```

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

using namespace std;

struct Node {
    char info;
    Node* left;
    Node* right;
};

Node* createNode(char data) {
    Node* newNode = new Node();
    newNode->info = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

void preOrder(Node* root) {
    if (root != NULL) {
        cout << root->info << " ";
        preOrder(root->left);
        preOrder(root->right);
    }
}

void inOrder(Node* root) {
    if (root != NULL) {
        inOrder(root->left);
        cout << root->info << " ";
        inOrder(root->right);
    }
}


void postOrder(Node* root) {
    if (root != NULL) {
        postOrder(root->left);
        postOrder(root->right);
        cout << root->info << " ";
    }
}

void levelOrder(Node* root) {
    if (root == NULL) return;

    queue<Node*> q;
    q.push(root);

    while (!q.empty()) {
        Node* current = q.front();
        q.pop();

        cout << current->info << " ";

        if (current->left != NULL) q.push(current->left);
        if (current->right != NULL) q.push(current->right);
    }
}

```

Exercise 2

```cpp
int main() {
    Node* root = createNode('/');

    root->left = createNode('*');
    root->right = createNode('H');

    root->left->left = createNode('-');
    root->left->right = createNode('G');

    root->left->left->left = createNode('-');
    root->left->left->right = createNode('F');

    root->left->left->left->left = createNode('-');
    root->left->left->left->right = createNode('E');

    root->left->left->left->left->left = createNode('+');
    root->left->left->left->left->right = createNode('N');

    root->left->left->left->left->left->left = createNode('+');
    root->left->left->left->left->left->right = createNode('M');

    root->left->left->left->left->left->left->left = createNode('K');
    root->left->left->left->left->left->left->right = createNode('L');

    cout << "\n1. Pre-order (Root-Left-Right):" << endl;
    preOrder(root);

    cout << "\n\n2. In-order (Left-Root-Right):" << endl;
    inOrder(root);

    cout << "\n\n3. Post-order (Left-Right-Root):" << endl;
    postOrder(root);

    cout << "\n\n4. Level-order (BFS):" << endl;
    levelOrder(root);

    cout << endl;
    return 0;
}
```

Output

```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuliah\semester3\strukturData\teori\tugasTree\" ; if ($?) { g++ exercise2.cpp -o exercise2 } ; if ($?) { .\exercise2 }

1. Pre-order (Root-Left-Right):
/ * - - - + + K L M N E F G H

2. In-order (Left-Root-Right):
K + L + M - N - E - F * G / H

3. Post-order (Left-Right-Root):
K L + M + N - E - F - G * H /

4. Level-order (BFS):
/ * H - G - F - E + N + M K L
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree>
```

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

using namespace std;

struct Node {
    char info;
    Node* left;
    Node* right;
};

Node* createNode(char data) {
    Node* newNode = new Node();
    newNode->info = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

void preOrder(Node* root) {
    if (root != NULL) {
        cout << root->info << " ";
        preOrder(root->left);
        preOrder(root->right);
    }
}

void inOrder(Node* root) {
    if (root != NULL) {
        inOrder(root->left);
        cout << root->info << " ";
        inOrder(root->right);
    }
}


void postOrder(Node* root) {
    if (root != NULL) {
        postOrder(root->left);
        postOrder(root->right);
        cout << root->info << " ";
    }
}

void levelOrder(Node* root) {
    if (root == NULL) return;

    queue<Node*> q;
    q.push(root);

    while (!q.empty()) {
        Node* current = q.front();
        q.pop();

        cout << current->info << " ";

        if (current->left != NULL) q.push(current->left);
        if (current->right != NULL) q.push(current->right);
    }
}

```

Exercise 3

```cpp
int main() {
    Node* root = createNode('/');

    root->left = createNode('+');
    root->right = createNode('+');

    root->left->left = createNode('*');
    root->left->right = createNode('D');

    root->right->left = createNode('E');
    root->right->right = createNode('*');

    root->left->left->left = createNode('+');
    root->left->left->right = createNode('C');

    root->right->right->left = createNode('F');
    root->right->right->right = createNode('H');

    root->left->left->left->left = createNode('A');
    root->left->left->left->right = createNode('B');

    cout << "\n1. Pre-order (Root-Left-Right):" << endl;
    preOrder(root);

    cout << "\n\n2. In-order (Left-Root-Right):" << endl;
    inOrder(root);

    cout << "\n\n3. Post-order (Left-Right-Root):" << endl;
    postOrder(root);

    cout << "\n\n4. Level-order (BFS):" << endl;
    levelOrder(root);

    cout << endl;
    return 0;
}
```

Output

```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuliah\semester3\strukturData\teori\tugasTree\" ; if ($?) { g++ exercise3.cpp -o exercise3 } ; if ($?) { .\exercise3 }

1. Pre-order (Root-Left-Right):
/ + * + A B C D + E * F H

2. In-order (Left-Root-Right):
A + B * C + D / E + F * H

3. Post-order (Left-Right-Root):
A B + C * D + E F H * + /

4. Level-order (BFS):
/ + + * D E * + C F H A B
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree>
```

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

using namespace std;

struct Node {
    char info;
    Node* left;
    Node* right;
};

Node* createNode(char data) {
    Node* newNode = new Node();
    newNode->info = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

void preOrder(Node* root) {
    if (root != NULL) {
        cout << root->info << " ";
        preOrder(root->left);
        preOrder(root->right);
    }
}

void inOrder(Node* root) {
    if (root != NULL) {
        inOrder(root->left);
        cout << root->info << " ";
        inOrder(root->right);
    }
}


void postOrder(Node* root) {
    if (root != NULL) {
        postOrder(root->left);
        postOrder(root->right);
        cout << root->info << " ";
    }
}

void levelOrder(Node* root) {
    if (root == NULL) return;

    queue<Node*> q;
    q.push(root);

    while (!q.empty()) {
        Node* current = q.front();
        q.pop();

        cout << current->info << " ";

        if (current->left != NULL) q.push(current->left);
        if (current->right != NULL) q.push(current->right);
    }
}

```

Exercise 4

```cpp
int main() {
    Node* root = createNode('A');

    root->left = createNode('B');
    root->right = createNode('C');

    root->left->left = createNode('D');
    root->left->right = createNode('E');

    root->right->left = createNode('G');
    root->right->right = createNode('H');

    root->left->left->left = createNode('I');

    root->left->right->left = createNode('J');
    root->left->right->right = createNode('K');

    root->right->left->left = createNode('L');
    root->right->left->right = createNode('M');

    root->right->left->right->left = createNode('N');
    root->right->left->right->right = createNode('O');

    cout << "\n1. Pre-order (Root-Left-Right):" << endl;
    preOrder(root);

    cout << "\n\n2. In-order (Left-Root-Right):" << endl;
    inOrder(root);

    cout << "\n\n3. Post-order (Left-Right-Root):" << endl;
    postOrder(root);

    cout << "\n\n4. Level-order (BFS):" << endl;
    levelOrder(root);

    cout << endl;
    return 0;
}
```

Output

```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree> cd "c:\shellyn\kuliah\semester3\strukturData\teori\tugasTree\" ; if ($?) { g++ exercise4.cpp -o exercise4 } ; if ($?) { .\exercise4 }

1. Pre-order (Root-Left-Right):
A B D I E J K C G L M N O H

2. In-order (Left-Root-Right):
I D B J E K A L G N M O C H

3. Post-order (Left-Right-Root):
I D J K E B L N O M G H C A

4. Level-order (BFS):
A B C D E G H I J K L M N O
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasTree>
```