```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int data;
    Node* left;
    Node* right;
    // Val is the key or the value that
    // has to be added to the data part
    Node(int val)
    {
        data = val;
        // Left and right child for node
        // will be initialized to null
        left = NULL;
        right = NULL;
    }
};

int main()
{
    /*create root*/
    Node* root = new Node(1);

    /* following is the tree after above statement
    1
    / \
    NULL NULL
    */
    root->left = new Node(2);
    root->right = new Node(3);

    /* 2 and 3 become left and right children of 1
       1
      / \
     2   3
    / \ / \
    NULL NULL NULL NULL
    */
    root->left->left = new Node(4);
    /* 4 becomes left child of 2
         1
       /   \
      2     3
     / \    / \
    4 NULL NULL NULL
    /   \
    NULL NULL
    */
    return 0;
}
```

```
#include <stdlib.h>

#include <iostream>

using namespace std;

struct node {
  int data;
  struct node *left;
  struct node *right;
};

// New node creation
struct node *newNode(int data) {
  struct node *node = (struct node *)malloc(sizeof(struct node));

  node->data = data;

  node->left = NULL;
  node->right = NULL;
  return (node);
}

// Traverse Preorder
void traversePreOrder(struct node *temp) {
  if (temp != NULL) {
    cout << " " << temp->data;
    traversePreOrder(temp->left);
    traversePreOrder(temp->right);
  }
}

// Traverse Inorder
void traverseInOrder(struct node *temp) {
  if (temp != NULL) {
    traverseInOrder(temp->left);
    cout << " " << temp->data;
    traverseInOrder(temp->right);
  }
}

// Traverse Postorder
void traversePostOrder(struct node *temp) {
```

```
  if (temp != NULL) {
    traversePostOrder(temp->left);
    traversePostOrder(temp->right);
    cout << " " << temp->data;
  }
}

int main() {
  struct node *root = newNode(1);
  root->left = newNode(2);
  root->right = newNode(3);
  root->left->left = newNode(4);

  cout << "preorder traversal: ";
  traversePreOrder(root);
  cout << "\nInorder traversal: ";
  traverseInOrder(root);
  cout << "\nPostorder traversal: ";
  traversePostOrder(root);
}
```