```
#include <iostream>
#include <queue>
#include <climits> // for INT_MIN and INT_MAX
using namespace std;
// Definition of a Node in the Binary Tree
struct Node {
  int val;
  Node* left;
  Node* right;
  Node(int x) {
    val = x;
    left = nullptr;
    right = nullptr;
};
// Function to calculate the height of the tree using
BFS (level-order traversal)
int getHeight(Node* root) {
  if (root == nullptr) return 0;
  queue<Node*> q;
  q.push(root);
  int height = 0;
  while (!q.empty()) {
    int levelSize = q.size();
    height++:
    for (int i = 0; i < levelSize; i++) {
       Node* node = q.front();
       q.pop();
       if (node->left != nullptr) q.push(node->left);
       if (node->right != nullptr) q.push(node->right);
  }
  return height;
// Function to count the number of nodes in the tree
using BFS (level-order traversal)
int getNodeCount(Node* root) {
  if (root == nullptr) return 0;
  queue<Node*> q;
  q.push(root);
  int count = 0;
  while (!q.empty()) {
    Node* node = q.front();
    q.pop();
    count++:
    if (node->left != nullptr) q.push(node->left);
    if (node->right != nullptr) q.push(node->right);
  return count;
```

Tree Structure:

```
1
/\
2 3
/\
4 5
```

Iterative tree operations in C++

♦ Function: getHeight(root)

This uses level-order traversal (BFS).

Level	Nodes at Level	Height So Far
1	1	1
2	2, 3	2
3	4, 5	3

[⊘] Result: 3

♦ Function: getNodeCount(root)

Counts nodes using BFS:

Step	Node Processed	Count	Queue
1	1	1	2, 3
2	2	2	3, 4, 5
3	3	3	4, 5
4	4	4	5
5	5	5	

[⊘] Result: 5

◆ Function: getMax(root)

Finds maximum using BFS:

Step	Node Processed	Max So Far
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5 ∜

[⊘] Result: 5

◆ Function: getMin(root)

Finds minimum using BFS:

```
// Function to find the maximum value in the tree
using BFS (level-order traversal)
int getMax(Node* root) {
  if (root == nullptr) throw invalid argument("Tree is
empty");
  queue<Node*> q;
  q.push(root);
  int maxValue = INT_MIN;
  while (!q.empty()) {
    Node* node = q.front();
    q.pop();
    maxValue = max(maxValue, node->val);
    if (node->left != nullptr) q.push(node->left);
    if (node->right != nullptr) q.push(node->right);
  return maxValue;
// Function to find the minimum value in the tree
using BFS (level-order traversal)
int getMin(Node* root) {
  if (root == nullptr) throw invalid_argument("Tree is
empty");
  queue<Node*> q;
  q.push(root);
  int minValue = INT MAX;
  while (!q.empty()) {
    Node* node = q.front();
    q.pop();
    minValue = min(minValue, node->val);
    if (node->left != nullptr) q.push(node->left);
    if (node->right != nullptr) q.push(node->right);
  return minValue;
}
int main() {
  // Constructing the example binary tree
  Node* root = new Node(1);
  root->left = new Node(2);
  root->right = new Node(3);
  root->left->left = new Node(4);
  root->left->right = new Node(5);
  // Using the functions to demonstrate the
functionality
  cout << "Height of the tree: " << getHeight(root) <<</pre>
endl:
  cout << "Number of nodes in the tree: " <<
getNodeCount(root) << endl;</pre>
    cout << "Maximum value in the tree: " <<
getMax(root) << endl:
    cout << "Minimum value in the tree: " <<
```

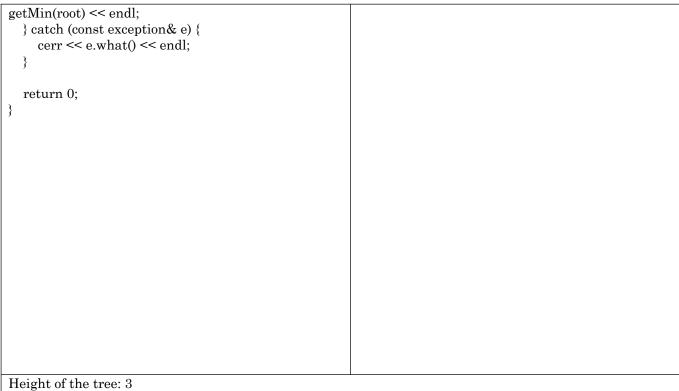
Š	Step	Node Processed	Min So Far
1		1	1 ∜
2		2	1
3		3	1
4	:	4	1
5		5	1

⊘ Result: 1

∜ Final Output:

Height of the tree: 3

Number of nodes in the tree: 5 Maximum value in the tree: 5 Minimum value in the tree: 1



Number of nodes in the tree: 5 Maximum value in the tree: 5 Minimum value in the tree: 1