Design and implement Parallel Breadth First on tree based on existing algorithms using OpenMP.

```
#include <iostream>
#include <queue>
#include <vector>
#include <omp.h>

struct Node {
    int data;
    std::vector<Node*> children;
};

void parallelBFS(Node* root) {
    std::queue<Node*> bfsQueue;
    bfsQueue.push(root);

    #pragma omp parallel
    {
        while (!bfsQueue.empty()) {
            #pragma omp for
            for (int i = 0; i < bfsQueue.size(); ++i) {
                Node* current = bfsQueue.front();
                bfsQueue.pop();

                // Process current node
                std::cout << current->data << " ";

                // Enqueue children of the current node
                #pragma omp for
                for (int j = 0; j < current->children.size(); ++j) {
                    bfsQueue.push(current->children[j]);
                }
            }
        }
    }
}

int main() {
    // Create a tree for testing
    Node* root = new Node{1, {}};
    Node* child1 = new Node{2, {}};
    Node* child2 = new Node{3, {}};
    Node* grandchild1 = new Node{4, {}};
    Node* grandchild2 = new Node{5, {}};

    root->children.push_back(child1);
    root->children.push_back(child2);
    child1->children.push_back(grandchild1);
    child2->children.push_back(grandchild2);

    parallelBFS(root);

    // Clean up allocated memory
    delete grandchild1;
    delete grandchild2;
    delete child1;
    delete child2;
    delete root;

    return 0;
}
```