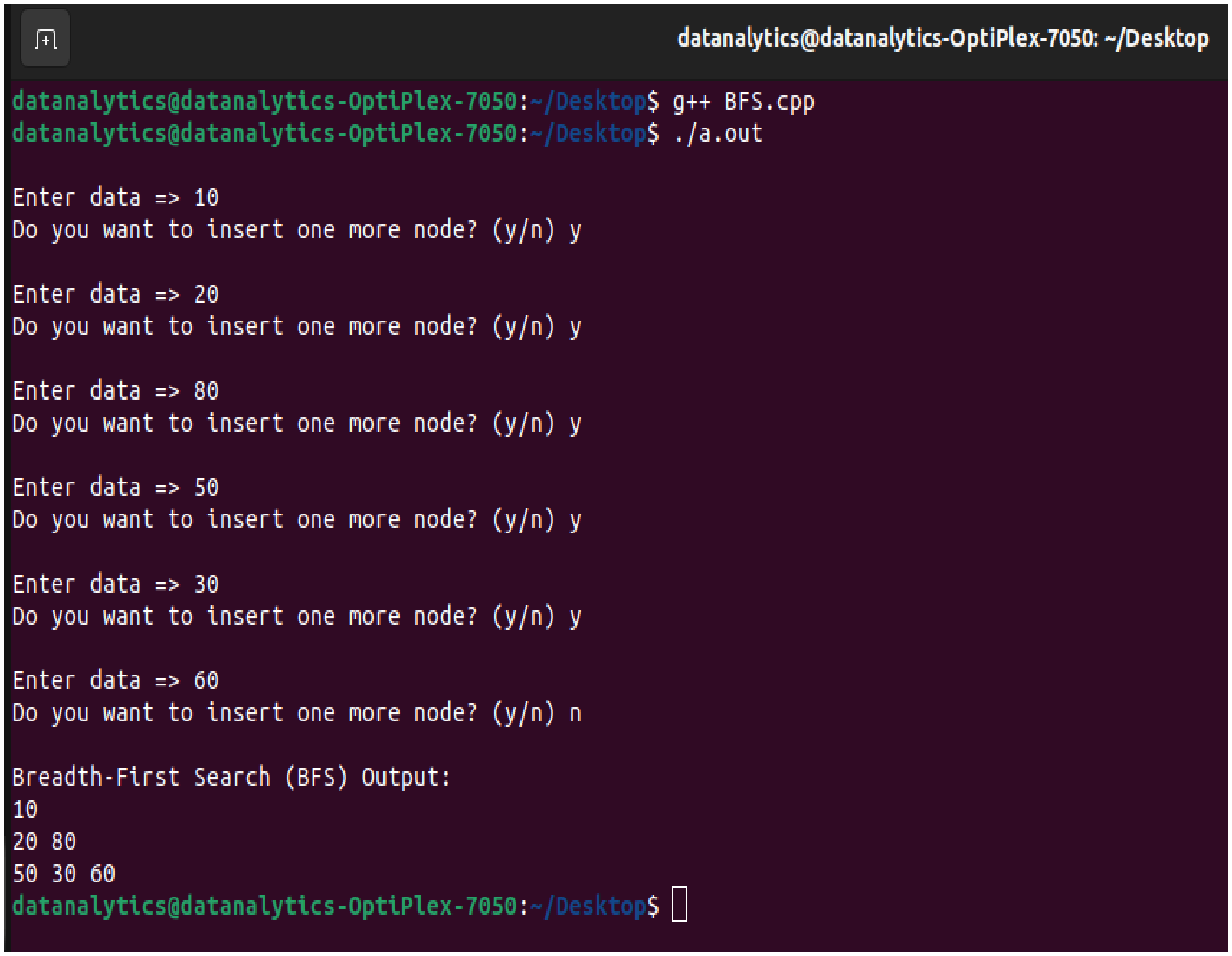
14162 HPC PR-1

Title: Parallel Breadth First Search based on existing algorithms using OpenMP

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

using namespace std;

class Node {
public:
    Node *left, *right;
    int data;
};

class Breadthfs {
public:
    Node* insert(Node* root, int data);
    void bfs(Node* head);
};

Node* Breadthfs::insert(Node* root, int data) {
    if (!root) {
        root = new Node;
        root->left = root->right = nullptr;
        root->data = data;
        return root;
    }

    queue<Node*> q;
    q.push(root);
while (!q.empty()) {
        Node* temp = q.front();
        q.pop();

        if (!temp->left) {
            temp->left = new Node;
            temp->left->left = temp->left->right = nullptr;
            temp->left->data = data;
            return root;
        } else {
            q.push(temp->left);
        }
        if (!temp->right) {
            temp->right = new Node;
            temp->right->left = temp->right->right = nullptr;
            temp->right->data = data;
            return root;
        } else {
            q.push(temp->right);
        }
    }
    return root;
}

void Breadthfs::bfs(Node* head) {
    if (!head) return;
queue<Node*> q;
    q.push(head);

    while (!q.empty()) {
        int qSize = q.size();

        #pragma omp parallel for
        for (int i = 0; i < qSize; i++) {
            Node* currNode;

            #pragma omp critical
            {
                currNode = q.front();
                q.pop();
                cout << currNode->data << " ";
            }

            #pragma omp critical
            {
                if (currNode->left) q.push(currNode->left);
                if (currNode->right) q.push(currNode->right);
            }
        }
        cout << endl;
    }
}

int main() {
    Node* root = nullptr;
    Breadthfs bfsHandler;
    int data;
    char ans;
    do {
        cout << "\nEnter data => ";
        cin >> data;
        root = bfsHandler.insert(root, data);

        cout << "Do you want to insert one more node? (y/n) ";
        cin >> ans;
    } while (ans == 'y' || ans == 'Y');

    cout << "\nBreadth-First Search (BFS) Output:\n";
    bfsHandler.bfs(root);

    return 0;
}
```

Output :


datanalytics@datanalytics-OptiPlex-7050: ~/Desktop

```
datanalytics@datanalytics-OptiPlex-7050:~/Desktop$ g++ BFS.cpp
datanalytics@datanalytics-OptiPlex-7050:~/Desktop$ ./a.out

Enter data => 10
Do you want to insert one more node? (y/n) y

Enter data => 20
Do you want to insert one more node? (y/n) y

Enter data => 80
Do you want to insert one more node? (y/n) y

Enter data => 50
Do you want to insert one more node? (y/n) y

Enter data => 30
Do you want to insert one more node? (y/n) y

Enter data => 60
Do you want to insert one more node? (y/n) n

Breadth-First Search (BFS) Output:
10
20 80
50 30 60
datanalytics@datanalytics-OptiPlex-7050:~/Desktop$
```