#include <iostream>

#include <stdlib.h>

#include <queue>

#include <vector>

#include <omp.h>

using namespace std;

class node {

public:

node \*left, \*right;

int data;

};

class Breadthfs {

public:

node \*insert(node \*, int);

void bfs(node \*);

};

node \*Breadthfs::insert(node \*root, int data) {

if (!root) {

root = new node;

root->left = NULL;

root->right = NULL;

root->data = data;

return root;

}

queue<node \*> q;

q.push(root);

while (!q.empty()) {

node \*temp = q.front();

q.pop();

if (temp->left == NULL) {

temp->left = new node;

temp->left->left = NULL;

temp->left->right = NULL;

temp->left->data = data;

return root;

} else {

q.push(temp->left);

}

if (temp->right == NULL) {

temp->right = new node;

temp->right->left = NULL;

temp->right->right = NULL;

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();

vector<node \*> levelNodes(qSize);

for (int i = 0; i < qSize; i++) {

levelNodes[i] = q.front();

q.pop();

}

#pragma omp parallel for

for (int i = 0; i < qSize; i++) {

cout << "\t" << levelNodes[i]->data;

}

for (int i = 0; i < qSize; i++) {

if (levelNodes[i]->left)

q.push(levelNodes[i]->left);

if (levelNodes[i]->right)

q.push(levelNodes[i]->right);

}

}

}

int main() {

node \*root = NULL;

Breadthfs tree;

int data;

char ans;

do {

cout << "\nEnter data => ";

cin >> data;

root = tree.insert(root, data);

cout << "Do you want to insert one more node? (y/n) ";

cin >> ans;

} while (ans == 'y' || ans == 'Y');

cout << "\nBFS Traversal: ";

tree.bfs(root);

cout << endl;

return 0;

}