**SOLUTION 4**

#include <iostream>

#include <sstream>

#include <string>

#include <vector>

#include <queue>

using namespace std;

class TreeNode {

public:

int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(int x) : val(x), left(NULL), right(NULL) {}

};

class Codec {

public:

string serialize(TreeNode\* root) {

ostringstream out;

serialize(root, out);

return out.str();

}

TreeNode\* deserialize(const string& data) {

istringstream in(data);

return deserialize(in);

}

private:

void serialize(TreeNode\* root, ostringstream& out) {

if (!root) {

out << "# ";

return;

}

out << root->val << " ";

serialize(root->left, out);

serialize(root->right, out);

}

TreeNode\* deserialize(istringstream& in) {

string val;

in >> val;

if (val == "#") {

return NULL;

}

TreeNode\* root = new TreeNode(stoi(val));

root->left = deserialize(in);

root->right = deserialize(in);

return root;

}

};

void printTree(TreeNode\* root) {

if (!root) return;

queue<TreeNode\*> q;

q.push(root);

while (!q.empty()) {

TreeNode\* node = q.front();

q.pop();

if (node) {

cout << node->val << " ";

q.push(node->left);

q.push(node->right);

} else {

cout << "# ";

}

}

cout << endl;

}

int main() {

Codec ser, deser;

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

root->right->left = new TreeNode(4);

root->right->right = new TreeNode(5);

string serialized = ser.serialize(root);

cout << "Serialized: " << serialized << endl;

TreeNode\* deserialized = deser.deserialize(serialized);

cout << "Deserialized (Level Order): ";

printTree(deserialized);

}

