Our Solution(s) Run Code

Solution 1

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

class Node {
public:
  string name;
  vector<Node *> children;

  Node(string name) { this->name = name; }

  // O(v + e) time | O(v) space
  vector<string> depthFirstSearch(vector<string> *array) {
    array->push_back(this->name);
    for (int i = 0; i < this->children.size(); i++) {
      children[i]->depthFirstSearch(array);
    }
    return *array;
  }

  Node *addChild(string name) {
    Node *child = new Node(name);
    children.push_back(child);
    return this;
  }
};

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
#include <vector>
using namespace std;

// Do not edit the class below except
// for the depthFirstSearch method.
// Feel free to add new properties
// and methods to the class.
class Node {
public:
  string name;
  vector<Node *> children;

  Node(string str) { name = str; }

  vector<string> depthFirstSearch(vector<string> *array) {
    // Write your code here.
    return {};
  }

  Node *addChild(string name) {
    Node *child = new Node(name);
    children.push_back(child);
    return this;
  }
};

```

Our Tests

```cpp
class ProgramTest : public TestSuite {
public:
  void Run() {

    RunTest("Test Case 1", []() {
      Node test1("A");
      test1.addChild("B")->addChild("C");
```

Custom Output Submit Code

```
    test1.children[0]->addChild("D");

    vector<string> expected{"A", "B", "D", "C"};
    vector<string> inputArray{};
    assert(test1.depthFirstSearch(&inputArray) == expected);
  });

  RunTest("Test Case 2", []() {
    Node test2("A");
    test2.addChild("B")->addChild("C")->addChild("D")->addChild("E")
    test2.children[1]->addChild("F");
    vector<string> expected{"A", "B", "C", "F", "D", "E"};
    vector<string> inputArray{};
    assert(test2.depthFirstSearch(&inputArray) == expected);
  });
```

Run or submit code when you're ready.