Employees Under Manager in C++

```
#include <iostream>
#include <unordered_map>
#include <unordered_set>
#include <string>

using namespace std;

int getSize(unordered_map<string,
unordered_set<string>>& tree, const string&
manager, unordered_map<string, int>& result) {
    if (tree.find(manager) == tree.end()) {
        result[manager] = 0;
        return 1;
    }
    int size = 0;
    for (const string& employee : tree[manager]) {
        int currentSize = getSize(tree, employee, result);
        size += currentSize;
    }
    result[manager] = size;
    return size + 1;
}

void findCount(unordered_map<string, string>&
map) {
    unordered_map<string, unordered_set<string>>
tree;
    string ceo = "";

    for (const auto& entry : map) {
        string employee = entry.first;
        string manager = entry.second;

        if (manager == employee) {
            ceo = manager;
        } else {
            tree[manager].insert(employee);
        }
    }

    unordered_map<string, int> result;
    getSize(tree, ceo, result);

    for (const auto& entry : result) {
        cout << entry.first << " " << entry.second <<
endl;
    }
}

int main() {
    unordered_map<string, string> map;
    map["A"] = "C";
    map["B"] = "C";
    map["C"] = "F";
    map["D"] = "E";
    map["E"] = "F";
    map["F"] = "F";

    findCount(map);

    return 0;
```

Step-by-Step Walkthrough of the Example:

Input:

```
unordered_map<string, string> map;
map["A"] = "C";
map["B"] = "C";
map["C"] = "F";
map["D"] = "E";
map["E"] = "F";
map["F"] = "F";
```

- A, B, and C report to C
- D reports to E, and E reports to F
- C reports to F and F reports to F (CEO)

Building the Tree:

- **Manager-Employee Tree Structure:**

  F: {C, E} (F manages C and E)
  E: {D} (E manages D)
  C: {A, B} (C manages A and B)
  D: {} (D has no subordinates)
  A: {} (A has no subordinates)
  B: {} (B has no subordinates)

- **CEO (F)** has employees C and E.
- **E** manages D.
- **C** manages A and B.

Dry Run:

Call getSize(tree, "F", result):

- Starting with **F**:
  - F has two direct subordinates: **C** and **E**.
  - Recursively calculate the size for **C** and **E**:
    - **C** has two direct subordinates: **A** and **B**.
      - Both **A** and **B** have no subordinates (base case).
      - Size of **C** = 2 (A and B).
    - **E** has one direct subordinate: **D**.
      - **D** has no subordinates (base case).
      - Size of **E** = 1 (D).
  - Total size of **F** = Size of C (2) + Size of E (1) = 5.

Final Sizes:

- **F**: 5 (Subordinates: C, E, A, B, D)

}	• **E**: 1 (Subordinate: D) • **C**: 2 (Subordinates: A, B) • **A**: 0 (No subordinates) • **B**: 0 (No subordinates) • **D**: 0 (No subordinates) **Output:** F 5 E 1 B 0 A 0 D 0 C 2
Output: F 5 E 1 B 0 A 0 D 0 C 2	