|  |  |
| --- | --- |
| **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 1: Construct tree and Identify CEO**   * Input mapping:   A -> C  B -> C  C -> F  D -> E  E -> F  F -> F (CEO identified)   * Constructing tree:   C -> {A, B}  F -> {C, E}  E -> {D}   * **CEO Identified**: F   **Step 2: Recursive Calls of getSize(tree, manager, result)**   | **Function Call** | **Processing Employee Set** | **Recursive Calls** | **Result Updates (result[manager])** | **Return Value** | | --- | --- | --- | --- | --- | | getSize(tree, "F", result) | {C, E} | getSize(tree, "C"), getSize(tree, "E") | F → 5 | 6 | | getSize(tree, "C", result) | {A, B} | getSize(tree, "A"), getSize(tree, "B") | C → 2 | 3 | | getSize(tree, "A", result) | {} (Base Case) | - | A → 0 | 1 | | getSize(tree, "B", result) | {} (Base Case) | - | B → 0 | 1 | | getSize(tree, "E", result) | {D} | getSize(tree, "D") | E → 1 | 2 | | getSize(tree, "D", result) | {} (Base Case) | - | D → 0 | 1 |   **Step 3: Output Values**  Final result map:  mathematica  CopyEdit  A → 0  B → 0  C → 2  D → 0  E → 1  F → 5  **Final Output**  A 0  B 0  C 2  D 0  E 1  F 5 |
| Output: F 5  E 1  B 0  A 0  D 0  C 2 | |