**Class employee:**

**First we add three private fields: parent, name and Dictionary:**

Parent contains current employee parent.

Name contain current employee name (we use it to add it in Dictionary after that).

In Dictionary we add as a key leader (first input after split) of current employee and for a value employee (as second input after split).

**After that we add four properties: position, parent, name and Dictionary of employees:**

Position property: we use it to set position of hierarchy of all employees.

Parent property: we use it to set parent of current employee.

Name property: we use it to set name of current employee

Dictionary of employees’ property: we use it to set leader as a key and employee as a value

**We have two constructors:**

For employee with only name

For employee with name and value

**We have one method:**

To add Dictionary of employees key and value (we use it in constructor with employee with name and value)

**Class FindEmployee:**

**Main methods: SplitInput, InsertLeader, InsertEmployee, GiveIndexes, SearchLeader, PrintResult:**

**SplitInput**: split current input with expression for two words with some special characters between them and give us array with 3 parts – arr[0] all input, arr[1] first word, arr[2] second word.

**InsertLeader**: check if Dictionary with allEmployees contains firstInput as a key and if not check if allEmployees contains secondInput as a key, if not we assume that second input is a child and create a new Employee (in Employee class) and we set new employee in allEmployess with key that employee. If second child exists we pull it from Dictionary and create new Employee (in Employee class) with key and value and add it allEmployees Dictionary. If firstInput is not in allEmployees Dictionary we set firstInput as parent and check if secondInput is in Dictionary as a child of this employee (firstInput) if yes we pull it and add it to that parent if not we creates it and then add it as child to that input.

**InsertEmployee:** check if Dictionary with allEmployees contain second input as a key and if not we create new Employee (in Employee class) and we assume that he have parent so we add as a parent firstInput and increase his position then we add to Dictionary with all employees secondInput as a key. If Dictionary doesn’t contain secondInput as a key we create new Employee (in Employee class) and add it to Dictionary with key secondInput and set firstInput as a parent.

**GiveIndexes:**  to be sure we’ll give all employees right indexes of their position we need one for and for each loops. We rotate first for loop with total employees count. First for each we rotate with all elements in allEmployees Dictionary and the second for each we rotate with values of each nested employee and give him indexes which depend of his parent position. (Example \*e1 has two children, and his position is 1, so his children position will be \*e1 position + 1 and so on).

**SearchLeader:**  from input we know which employees leader we are searching so we set them as a new employees (we assume that input is correct and we have them added in Dictionary). So we check which position is bigger and then we take his parent value and so on. After they are at the same position we check if their parent has same name if not we get both parent value and check it again and so on.

**PrintResult:** we take result from **SearchLeader** and print it on console.