

/\* Below is an example organisation chart. At the top is the ceo, Mark Zuckerberg. Mark's subordinates are Sarah, Tyler, Bruce and Georgina.

Mark Zuckerberg:

- Sarah Donald:
  - Cassandra Reynolds:
    - Mary Blue:
    - Bob Saget:
    - Tina Teff:
    - Will Turner:
- Tyler Simpson:
  - Harry Tobs:
    - Thomas Brown:
  - George Carrey:
  - Gary Styles:
- Bruce Willis:
- Georgina Flangy:
  - Sophie Turner:

The CEO is represented with the following structure.

```
interface Employee {  
    uniqueId: number;  
    name: string;  
    subordinates: Employee[];  
}
```

```
const ceo: Employee = {  
    uniqueId: 1  
    name: Mark Zuckerberg,  
    subordinates: [Employee, Employee, ....]  
}
```

Your task is to create a concrete class called EmployeeOrgApp that implements IEmployeeOrgApp. The class should be instantiable with the ceo as a constructor parameter.

E.g. const app = new EmployeeOrgApp(ceo)

**The class should:**

1. move employee A to become the subordinate of employee B (i.e. B becomes A's supervisor)
2. undo/redo the move action

**ASSUMPTIONS:**

You may assume:

- when an employee (e.g. Bob Saget) is moved to a new supervisor (e.g. Georgina), Bob's existing subordinates (Tina Teff) will become the subordinate of Cassandra, Bob's old supervisor.
- employees without any subordinates have an empty list for their subordinates property
- **You may not clone the state/tree during each action (move/undo/redo).**

**ASSESSMENT CRITERIA:**

1. The efficiency of the code
2. Object oriented programming design
3. Readability
4. Completeness of solution

**REQUIREMENTS:**

Must be written in Typescript.

```
*/  
interface Employee {  
    uniqueId: number;  
    name: string;  
    subordinates: Employee[];  
}  
  
interface IEmployeeOrgApp {  
    ceo: Employee;  
  
    /**  
     * Moves the employee with employeeID (uniqueId) under a supervisor  
(another employee) that has supervisorID (uniqueId).  
     * E.g. move Bob (employeeID) to be subordinate of Georgina  
(supervisorID). * @param employeeID  
     * @param supervisorID  
     */  
    move(employeeID: number, supervisorID: number): void;  
  
    /** Undo last move action */  
    undo(): void;  
  
    /** Redo last undone action */  
    redo(): void;  
}
```