```
function Node(data) {
  this.data = data;
  this.children = [];
}

class Tree {
  constructor() {
    this.root = null;
  }

  add(data, toNodeData) {
    const node = new Node(data);
    const parent = toNodeData ? this.findBFS(toNodeData) : null;
    if(parent) {
      parent.children.push(node);
    } else {
      if(!this.root) {
        this.root = node;
      } else {
        return 'Root node is already assigned';
      }
    }
  }

  remove(data) {
    if(this.root.data === data) {
      this.root = null;
    }

    const queue = [this.root];
    while(queue.length) {
      const node = queue.shift();
      for (let [index, child] of node.children.entries()) {
        if(child.data === data) {
          node.children.splice(index, 1);
        } else {
          queue.push(child);
        }
      }
    }
  }

  contains(data) {
    return !!this.findBFS(data);
  }

  findBFS(data) {
    const queue = [this.root];
    while(queue.length) {
      const node = queue.shift();
      if(node.data === data) {
        return node;
      }
      for(const child of node.children) {
        queue.push(child);
      }
    }
    return null;
  }

```

```
  _preOrder(node, fn) {
    if(node) {
      if(fn) {
        fn(node);
      }
      for(const child of node.children) {
        this._preOrder(child, fn);
      }
    }
  }

  _postOrder(node, fn) {
    if(node) {
      for(const child of node.children) {
        this._postOrder(child, fn);
      }
      if(fn) {
        fn(node);
      }
    }
  }

  traverseDFS(fn, method) {
    const current = this.root;
    if(method) {
      this[`_${method}`](current, fn);
    } else {
      this._preOrder(current, fn);
    }
  }

  traverseBFS(fn) {
    const queue = [this.root];
    while(queue.length) {
      const node = queue.shift();
      if(fn) {
        fn(node);
      }
      for(const child of node.children) {
        queue.push(child);
      }
    }
  }

  print() {
    if(!this.root) {
      return console.log('No root node found');
    }
    const newline = new Node('|');
    const queue = [this.root, newline];
    let string = '';
    while(queue.length) {
      const node = queue.shift();
      string += `${node.data.toString()} `;
      if(node === newline && queue.length) {
        queue.push(newline);
      }
      for(const child of node.children) {
        queue.push(child);
      }
```

```
    }
    console.log(string.slice(0, -2).trim());
  }

  printByLevel() {
    if(!this.root) {
      return console.log('No root node found');
    }
    const newline = new Node('\n');
    const queue = [this.root, newline];
    let string = '';
    while(queue.length) {
      const node = queue.shift();
      string += node.data.toString() + (node.data !== '\n' ? ' ' : '');
      if(node === newline && queue.length) {
        queue.push(newline);
      }
      for(const child of node.children) {
        queue.push(child);
      }
    }
    console.log(string.trim());
  }
}

const tree = new Tree();
tree.add('ceo');
tree.add('cto', 'ceo');
tree.add('dev1', 'cto');
tree.add('dev2', 'cto');
tree.add('dev3', 'cto');
tree.add('cfo', 'ceo');
tree.add('accountant', 'cfo');
tree.add('cmo', 'ceo');
tree.print(); // => ceo | cto cfo cmo | dev1 dev2 dev3 accountant
tree.printByLevel();  // => ceo \n cto cfo cmo \n dev1 dev2 dev3 accountant
console.log('tree contains dev1 is true:', tree.contains('dev1')); // => true
console.log('tree contains dev4 is false:', tree.contains('dev4')); // => false
console.log('--- BFS');
tree.traverseBFS(node => { console.log(node.data); }); // => ceo cto cfo cmo dev1
dev2 dev3 accountant
console.log('--- DFS preOrder');
tree.traverseDFS(node => { console.log(node.data); }, 'preOrder'); // => ceo cto dev1
dev2 dev3 cfo accountant cmo
console.log('--- DFS postOrder');
tree.traverseDFS(node => { console.log(node.data); }, 'postOrder'); // => dev1 dev2
dev3 cto accountant cfo cmo ceo
tree.remove('cmo');
tree.print(); // => ceo | cto cfo | dev1 dev2 dev3 accountant
tree.remove('cfo');
tree.print(); // => ceo | cto | dev1 dev2 dev3

```