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

function Tree() {
  this.root = null;
}

Tree.prototype.add = function(data, toNodeData) {
  var node = new Node(data);
  var 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';
    }
  }
};
Tree.prototype.remove = function(data) {
  if(this.root.data === data) {
    this.root = null;
  }

  var queue = [this.root];
  while(queue.length) {
    var node = queue.shift();
    for(var i = 0; i < node.children.length; i++) {
      if(node.children[i].data === data) {
        node.children.splice(i, 1);
      } else {
        queue.push(node.children[i]);
      }
    }
  }
};
Tree.prototype.contains = function(data) {
  return this.findBFS(data) ? true : false;
};
Tree.prototype.findBFS = function(data) {
  var queue = [this.root];
  while(queue.length) {
    var node = queue.shift();
    if(node.data === data) {
      return node;
    }
    for(var i = 0; i < node.children.length; i++) {
      queue.push(node.children[i]);
    }
  }
  return null;
};
Tree.prototype._preOrder = function(node, fn) {
  if(node) {
    if(fn) {
      fn(node);
    }
```

```
    for(var i = 0; i < node.children.length; i++) {
      this._preOrder(node.children[i], fn);
    }
  }
};
Tree.prototype._postOrder = function(node, fn) {
  if(node) {
    for(var i = 0; i < node.children.length; i++) {
      this._postOrder(node.children[i], fn);
    }
    if(fn) {
      fn(node);
    }
  }
};
Tree.prototype.traverseDFS = function(fn, method) {
  var current = this.root;
  if(method) {
    this['_' + method](current, fn);
  } else {
    this._preOrder(current, fn);
  }
};
Tree.prototype.traverseBFS = function(fn) {
  var queue = [this.root];
  while(queue.length) {
    var node = queue.shift();
    if(fn) {
      fn(node);
    }
    for(var i = 0; i < node.children.length; i++) {
      queue.push(node.children[i]);
    }
  }
};
Tree.prototype.print = function() {
  if(!this.root) {
    return console.log('No root node found');
  }
  var newline = new Node('|');
  var queue = [this.root, newline];
  var string = '';
  while(queue.length) {
    var node = queue.shift();
    string += node.data.toString() + ' ';
    if(node === newline && queue.length) {
      queue.push(newline);
    }
    for(var i = 0; i < node.children.length; i++) {
      queue.push(node.children[i]);
    }
  }
  console.log(string.slice(0, -2).trim());
};
Tree.prototype.printByLevel = function() {
  if(!this.root) {
    return console.log('No root node found');
  }
  var newline = new Node('\n');
  var queue = [this.root, newline];
```

```
  var string = '';
  while(queue.length) {
    var node = queue.shift();
    string += node.data.toString() + (node.data !== '\n' ? ' ' : '');
    if(node === newline && queue.length) {
      queue.push(newline);
    }
    for(var i = 0; i < node.children.length; i++) {
      queue.push(node.children[i]);
    }
  }
  console.log(string.trim());
};

var 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(function(node) { console.log(node.data); }); // => ceo cto cfo cmo
dev1 dev2 dev3 accountant
console.log('--- DFS preOrder');
tree.traverseDFS(function(node) { console.log(node.data); }, 'preOrder'); // => ceo
cto dev1 dev2 dev3 cfo accountant cmo
console.log('--- DFS postOrder');
tree.traverseDFS(function(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

```