```
function Node(data) {
  this.data = data;
  this.isWord = false;
  this.prefixes = 0;
  this.children = {};
}

function Trie() {
  this.root = new Node('');
}

Trie.prototype.add = function(word) {
  if(!this.root) {
    return null;
  }
  this._addNode(this.root, word);
};
Trie.prototype._addNode = function(node, word) {
  if(!node || !word) {
    return null;
  }
  node.prefixes++;
  var letter = word.charAt(0);
  var child = node.children[letter];
  if(!child) {
    child = new Node(letter);
    node.children[letter] = child;
  }
  var remainder = word.substring(1);
  if(!remainder) {
    child.isWord = true;
  }
  this._addNode(child, remainder);
};
Trie.prototype.remove = function(word) {
  if(!this.root) {
    return;
  }
  if(this.contains(word)) {
    this._removeNode(this.root, word);
  }
};
Trie.prototype._removeNode = function(node, word) {
  if(!node || !word) {
    return;
  }
  node.prefixes--;
  var letter = word.charAt(0);

  var child = node.children[letter];
  if(child) {
    var remainder = word.substring(1);
    if(remainder) {
      if(child.prefixes === 1) {
        delete node.children[letter];
      } else {
        this._removeNode(child, remainder);
      }
    } else {
      if(child.prefixes === 0) {
```

```
        delete node.children[letter];
      } else {
        child.isWord = false;
      }
    }
  }
};
Trie.prototype.contains = function(word) {
  if(!this.root) {
    return false;
  }
  return this._contains(this.root, word);
};
Trie.prototype._contains = function(node, word) {
  if(!node || !word) {
    return false;
  }
  var letter = word.charAt(0);
  var child = node.children[letter];
  if(child) {
    var remainder = word.substring(1);
    if(!remainder && child.isWord) {
      return true;
    } else {
      return this._contains(child, remainder);
    }
  } else {
    return false;
  }
};
Trie.prototype.countWords = function() {
  if(!this.root) {
    return console.log('No root node found');
  }
  var queue = [this.root];
  var counter = 0;
  while(queue.length) {
    var node = queue.shift();
    if(node.isWord) {
      counter++;
    }
    for(var child in node.children) {
      if(node.children.hasOwnProperty(child)) {
        queue.push(node.children[child]);
      }
    }
  }
  return counter;
};
Trie.prototype.getWords = function() {
  var words = [];
  var word = '';
  this._getWords(this.root, words, word);
  return words;
};
Trie.prototype._getWords = function(node, words, word) {
  for(var child in node.children) {
    if(node.children.hasOwnProperty(child)) {
      word += child;
      if (node.children[child].isWord) {
```

```
        words.push(word);
      }
      this._getWords(node.children[child], words, word);
      word = word.substring(0, word.length - 1);
    }
  }
};
Trie.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 child in node.children) {
      if(node.children.hasOwnProperty(child)) {
        queue.push(node.children[child]);
      }
    }
  }
  console.log(string.slice(0, -2).trim());
};
Trie.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 child in node.children) {
      if(node.children.hasOwnProperty(child)) {
        queue.push(node.children[child]);
      }
    }
  }
  console.log(string.trim());
};

var trie = new Trie();
trie.add('one');
trie.add('two');
trie.add('fifth');
trie.add('fifty');
trie.print(); // => | o t f | n w i | e o f | t | h y
trie.printByLevel(); // => o t f \n n w i \n e o f \n t \n h y
console.log('words are: one, two, fifth, fifty:', trie.getWords()); // => [ 'one',
'two', 'fifth', 'fifty' ]
console.log('trie count words is 4:', trie.countWords()); // => 4
```

```
180 console.log('trie contains one is true:', trie.contains('one')); // => true
181 console.log('trie contains on is false:', trie.contains('on')); // => false
182 trie.remove('one');
183 console.log('trie contains one is false:', trie.contains('one')); // => false
184 console.log('trie count words is 3:', trie.countWords()); // => 3
185 console.log('words are two, fifth, fifty:', trie.getWords()); // => [ 'two', 'fifth',
    'fifty' ]
186
```