Recursion

1. Given a positive number. Write a recursive function that returns its factorial. ( n! = n \* (n-1) \* (n-2) \* … \* 3 \* 2 \* 1 , 0! = 1 )

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 | “5! = 120” |
| 1 | “1! = 1” |
| 7 | “7! = 5040” |

1. Given an integer. Write a recursive function that returns **n**-th element in the Fibonacci sequence.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 0 | 0 |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 21 | 10946 |

1. Given an array. Write a recursive function that returns the sum of it. (Hint: *arr.pop()*)

|  |  |
| --- | --- |
| **Input** | **Output** |
| [] | 0 |
| [1, -1, 35] | 35 |
| [1, 10, 12, 3] | 26 |

1. Given a number. Write a recursive function that reverse the number. Return the new number.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | 2 |
| 13 | 31 |
| 815796 | 697518 |

1. Given an array of nested arrays. Write a recursive function that flattens it. (Hint create function that concats arrays).

|  |  |
| --- | --- |
| **Input** | **Output** |
| [1, [3, 4, [1, 2]], 10] | [1, 3, 4, 1, 2, 10] |
| [14, [1, [[[3, []]], 1], 0] | [14, 1, 3, 1, 0] |

1. Given a number. Write a function that calculates its sum of the digits and if that sum has more than 1 digit find the sum of digits of that number. Repeat that process if needed and return the result.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 14 | 5 |
| 29 | 2 |
| 999999999999 | 9 |

Object Literals

1. Given an object. Invert it (keys become values and values become keys). If there is more than key for that given value create an array.

|  |  |
| --- | --- |
| **Input** | **Output** |
| { a: ‘1’, b: ‘2’ } | { 1: ‘a’, 2: ‘b’ } |
| { a: ‘1’, b: ‘2’, c: ‘2’ } | { 1: ‘a’, 2: [‘b’, ‘c’] } |
| { a: ‘1’, b: ‘2’, c: ‘2’, d: ‘2’ } | { 1: ‘a’, 2: [‘b’, ‘c’, ‘d’] } |

1. Given two objects. Write a function that performs shallow compare.

|  |  |
| --- | --- |
| **Input** | **Output** |
| var a = { a: ‘1’ };  var b = { a: ‘1’};  shallowCompare(a, b) | true |
| var a = { a: ‘1’ };  var b = { a: ‘1’, b: ‘2’ };  shallowCompare(a, b) | false |
| var a = { };  var b = { };  shallowCompare(a, b) | true |

1. Given an object. Write a function that creates a deep copy of it.

|  |  |
| --- | --- |
| **Input** | **Output** |
| var a = { a: ‘1’, b: { a: 2} };  var b = deepCopy(a);  a.b.a = 123;  console.log(b.b.a !== 123); | true |