```javascript
/* jshint esversion: 6 */

// Solve the following prompts using recursion.

// 1. Calculate the factorial of a number. The factorial of a non-negative integer n,
// denoted by n!, is the product of all positive integers less than or equal to n.
// Example: 5! = 5 x 4 x 3 x 2 x 1 = 120
// factorial(5); // 120
var factorial = function(n) {
};

// 2. Compute the sum of an array of integers.
// sum([1,2,3,4,5,6]); // 21
var sum = function(array) {
};

// 3. Sum all numbers in an array containing nested arrays.
// arraySum([1,[2,3],[[4]],5]); // 15
var arraySum = function(array) {
};

// 4. Check if a number is even.
var isEven = function(n) {
};

// 5. Sum all integers below a given integer.
// sumBelow(10); // 45
// sumBelow(7); // 21
var sumBelow = function(n) {
};

// 6. Get the integers within a range (x, y).
// range(2,9); // [3,4,5,6,7,8]
var range = function(x, y) {
};

// 7. Compute the exponent of a number.
// The exponent of a number says how many times the base number is used as a factor.
// 8^2 = 8 x 8 = 64. Here, 8 is the base and 2 is the exponent.
// exponent(4,3); // 64
// https://www.khanacademy.org/computing/computer-science/algorithms/recursive-algorithms/a/computing-powers-of-a-number
var exponent = function(base, exp) {
};

// 8. Determine if a number is a power of two.
// powerOfTwo(1); // true
// powerOfTwo(16); // true
// powerOfTwo(10); // false
var powerOfTwo = function(n) {
};

// 9. Write a function that reverses a string.
var reverse = function(string) {
};

// 10. Write a function that determines if a string is a palindrome.
var palindrome = function(string) {
};

// 11. Write a function that returns the remainder of x divided by y without using the
// modulo (%) operator.
// modulo(5,2) // 1
// modulo(17,5) // 2
// modulo(22,6) // 4
var modulo = function(x, y) {
};

// 12. Write a function that multiplies two numbers without using the * operator or
// Math methods.
var multiply = function(x, y) {
};

// 13. Write a function that divides two numbers without using the / operator or
// Math methods to arrive at an approximate quotient (ignore decimal endings).
var divide = function(x, y) {
};

// 14. Find the greatest common divisor (gcd) of two positive numbers. The GCD of two
// integers is the greatest integer that divides both x and y with no remainder.
// gcd(4,36); // 4
// http://www.cse.wustl.edu/~kjg/cse131/Notes/Recursion/recursion.html
// https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm
var gcd = function(x, y) {
};

// 15. Write a function that compares each character of two strings and returns true if
// both are identical.
// compareStr('house', 'houses') // false
// compareStr('tomato', 'tomato') // true
var compareStr = function(str1, str2) {
};

// 16. Write a function that accepts a string and creates an array where each letter
// occupies an index of the array.
var createArray = function(str) {
};

// 17. Reverse the order of an array
var reverseArr = function(array) {
};

// 18. Create a new array with a given value and length.
// buildList(0,5) // [0,0,0,0,0]
// buildList(7,3) // [7,7,7]
var buildList = function(value, length) {
};

// 19. Implement FizzBuzz. Given integer n, return an array of the string representations of 1 to n.
// For multiples of three, output 'Fizz' instead of the number.
// For multiples of five, output 'Buzz' instead of the number.
// For numbers which are multiples of both three and five, output "FizzBuzz" instead of the number.
// fizzBuzz(5) // ['1','2','Fizz','4','Buzz']
var fizzBuzz = function(n) {
};

// 20. Count the occurrence of a value in a list.
// countOccurrence([2,7,4,4,1,4], 4) // 3
// countOccurrence([2,'banana',4,4,1,'banana'], 'banana') // 2
var countOccurrence = function(array, value) {
};

// 21. Write a recursive version of map.
// rMap([1,2,3], timesTwo); // [2,4,6]
var rMap = function(array, callback) {
};

// 22. Write a function that counts the number of times a key occurs in an object.
// var obj = {'e':{'x':'y'},'t':{'r':{'e':'r'},'p':{'y':'r'}},'y':'e'};
// countKeysInObj(obj, 'r') // 1
// countKeysInObj(obj, 'e') // 2
var countKeysInObj = function(obj, key) {
};

// 23. Write a function that counts the number of times a value occurs in an object.
// var obj = {'e':{'x':'y'},'t':{'r':{'e':'r'},'p':{'y':'r'}},'y':'e'};
// countValuesInObj(obj, 'r') // 2
// countValuesInObj(obj, 'e') // 1
var countValuesInObj = function(obj, value) {
};

```

```javascript
// 24. Find all keys in an object (and nested objects) by a provided name and rename
// them to a provided new name while preserving the value stored at that key.
var replaceKeysInObj = function(obj, oldKey, newKey) {
};

// 25. Get the first n Fibonacci numbers. In the Fibonacci sequence, each subsequent
// number is the sum of the previous two.
// Example: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34.....
// fibonacci(5); // [0,1,1,2,3,5]
// Note: The 0 is not counted.
var fibonacci = function(n) {
};

// 26. Return the Fibonacci number located at index n of the Fibonacci sequence.
// [0,1,1,2,3,5,8,13,21]
// nthFibo(5); // 5
// nthFibo(7); // 13
// nthFibo(3); // 2
var nthFibo = function(n) {
};

// 27. Given an array of words, return a new array containing each word capitalized.
// var words = ['i', 'am', 'learning', 'recursion'];
// capitalizedWords(words); // ['I', 'AM', 'LEARNING', 'RECURSION']
var capitalizeWords = function(array) {
};

// 28. Given an array of strings, capitalize the first letter of each index.
// capitalizeFirst(['car','poop','banana']); // ['Car','Poop','Banana']
var capitalizeFirst = function(array) {
};

// 29. Return the sum of all even numbers in an object containing nested objects.
// var obj1 = {
//   a: 2,
//   b: {b: 2, bb: {b: 3, bb: {b: 2}}},
//   c: {c: {c: 2}, cc: 'ball', ccc: 5},
//   d: 1,
//   e: {e: {e: 2}, ee: 'car'}
// };
// nestedEvenSum(obj1); // 10
var nestedEvenSum = function(obj) {
};

// 30. Flatten an array containing nested arrays.
// flatten([1,[2],[3,[[4]]],5]); // [1,2,3,4,5]
var flatten = function(array) {
};

// 31. Given a string, return an object containing tallies of each letter.
// letterTally('potato'); // {p:1, o:2, t:2, a:1}
var letterTally = function(str, obj) {
};

// 32. Eliminate consecutive duplicates in a list. If the list contains repeated
// elements they should be replaced with a single copy of the element. The order of the
// elements should not be changed.
// compress([1,2,2,3,4,4,5,5,5]) // [1,2,3,4,5]
// compress([1,2,2,3,4,4,2,5,5,5,4,4]) // [1,2,3,4,2,5,4]
var compress = function(list) {
};

// 33. Augment every element in a list with a new value where each element is an array
// itself.
// augmentElements([[],[3],[7]], 5); // [[5],[3,5],[7,5]]
var augmentElements = function(array, aug) {
};

// 34. Reduce a series of zeroes to a single 0.
// minimizeZeroes([2,0,0,0,1,4]) // [2,0,1,4]
// minimizeZeroes([2,0,0,0,1,0,0,4]) // [2,0,1,0,4]
var minimizeZeroes = function(array) {
};

// 35. Alternate the numbers in an array between positive and negative regardless of
// their original sign. The first number in the index always needs to be positive.
// alternateSign([2,7,8,3,1,4]) // [2,-7,8,-3,1,-4]
// alternateSign([-2,-7,8,3,-1,4]) // [2,-7,8,-3,1,-4]
var alternateSign = function(array) {
};

// 36. Given a string, return a string with digits converted to their word equivalent.
// Assume all numbers are single digits (less than 10).
// numToText("I have 5 dogs and 6 ponies"); // "I have five dogs and six ponies"
var numToText = function(str) {
};


// *** EXTRA CREDIT ***

// 37. Return the number of times a tag occurs in the DOM.
var tagCount = function(tag, node) {
};

// 38. Write a function for binary search.
// var array = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15];
// binarySearch(array, 5) // 5
// https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search
var binarySearch = function(array, target, min, max) {
};

// 39. Write a merge sort function.
// mergeSort([34,7,23,32,5,62]) // [5,7,23,32,34,62]
// https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms
var mergeSort = function(array) {
};

// 40. Deeply clone objects and arrays.
// var obj1 = {a:1,b:{bb:{bbb:2}},c:3};
// var obj2 = clone(obj1);
// console.log(obj2); // {a:1,b:{bb:{bbb:2}},c:3}
// obj1 === obj2 // false
var clone = function(input) {
};
```