hw3_1

July 26, 2024

1. Write a JavaScript function that reverse a number. Example x = 32243; Expected Output: 34223

[ ]:
```
function reverse(num) {
  let myString = num.toString();
  myString = myString.split("").reverse().join(""); //reverse
  let res = Number(myString);
  return res;
}
```

[ ]:
```
x = 32243;
console.log(reverse(x));
```

```
34223
```

2. Write a JavaScript function that checks whether a passed string is palindrome or not? A palindrome is word, phrase, or sequence that reads the same backward as forward, e.g., madam or nurses run.

[ ]:
```
function isPalindrome(s) {
  let left = 0;
  let right = s.length - 1;
  while (left < right) {
    while (left < right && s[left] === " ") {
      left++;
    }
    while (left < right && s[right] === " ") {
      right--;
    }
    if (s[left] !== s[right]) {
      return false;
    }
    left++;
    right--;
  }
  return true;
}
```

[ ]:
```
console.log(isPalindrome("madam"));
console.log(isPalindrome("nurses run"));
console.log(isPalindrome("tree"));
```

```
true
true
false
```

3. Write a JavaScript function that generates all combinations of a string. Example string: 'dog' Expected Output: d, do, dog, o, og, g

[ ]:
```
function allSubstrings(s) {
  let res = [];
  for (let i = 0; i < s.length; i++) {
    var temp = s[i];
    res.push(temp);
    for (let j = i + 1; j < s.length; j++) {
      temp += s[j];
      res.push(temp);
    }
  }
  return res;
}
```

[ ]:
```
console.log(allSubstrings("dog"));
```

```
[ 'd', 'do', 'dog', 'o', 'og', 'g' ]
```

4. Write a JavaScript function that returns a passed string with letters in alphabetical order. Example string: 'webmaster' Expected Output: 'abeemrstw' Assume punctuation and numbers symbols are not included in the passed string.

[ ]:
```
function sortString(s) {
  return s.split("").sort().join("");
}
```

[ ]:
```
console.log(sortString("webmaster"));
```

```
abeemrstw
```

5. Write a JavaScript function that accepts a string as a parameter and converts the first letter of each word of the string in upper case. Example string: 'the quick brown fox' Expected Output: 'The Quick Brown Fox'

[ ]:
```
//string is imputable
function fixUpperCase(s){
    let list = s.split(" ")
    var tempt = ""
```

```
    for (let i in list){
        tempt = list[i].split("")
        tempt[0] = tempt[0].toUpperCase()
        list[i] = tempt.join("")
    }
    return list.join(" ")
}
```

[ ]:
```
console.log(fixUpperCase("the quick brown fox"))
```

```
The Quick Brown Fox
```

6. Write a JavaScript function that accepts a string as a parameter and find the longest word within the string. Example string: 'Web Development Tutorial' Expected Output: 'Development'

[ ]:
```
function longestWord(s) {
  if (s === "" || s === null || s === undefined) {
    return "Check input again";
  }
  let maxCount = 0;
  let longestIndex = 0;
  let words = s.split(" ");
  for (let i in words) {
    if (maxCount < words[i].length) {
      maxCount = words[i].length;
      longestIndex = i;
    }
  }
  return words[longestIndex];
}
```

[ ]:
```
console.log(longestWord("Web Development Tutorial"));
console.log(longestWord(""));
```

```
Development
Check input again
```

7. Write a JavaScript function that accepts a string as a parameter and counts the number of vowels within the string. Note: As the letter 'y' can be regarded as both a vowel and a consonant, we do not count 'y' as vowel here. Example string: 'The quick brown fox' Expected Output: 5

[ ]:
```
function countVowels(s) {
  const vowels = new Set(["a", "e", "i", "o", "u"]);
  let count = 0;
  for (let c of s) {
    if (vowels.has(c)) {
```

```
      count++;
    }
  }
  return count;
}
```

[ ]:
```
console.log(countVowels("The quick brown fox"));
```

```
5
```

8. Write a JavaScript function that accepts a number as a parameter and check the number is prime or not. Note: A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself.

[ ]:
```
function isPrime(num){
    let sqrt = Math.sqrt(num)
    for (let i = 1; i<=sqrt;i++){
        if (num%i==0 && i!=1 && i!=num){
            return false
        }
    }
    return true
}
```

[ ]:
```
console.log(isPrime(83))
console.log(isPrime(2))
console.log(isPrime(293))
console.log(isPrime(24))
```

```
true
true
true
false
```

9. Write a JavaScript function which accepts an argument and returns the type. Note: There are six possible values that typeof returns: object, boolean, function, number, string, and undefined.

[ ]:
```
function checkType(s){
    return typeof(s)
}
```

[ ]:
```
console.log(checkType("code"));
console.log(checkType(123));
console.log(checkType(1 == 2));
console.log(checkType([1, 2, 3]));
```

```
console.log(checkType({ name: "john", age: 20 }));
var i;
console.log(checkType(i));
const plus1 = (x) => {
  x + 1;
};
console.log(checkType(plus1));
```

```
string
number
boolean
object
object
undefined
function
```

10. Write a JavaScript function which returns the n rows by n columns identity matrix.

[ ]:
```
function identityMatrix(n) {
  let myArray = Array.from(Array(n), _ => Array(n).fill(0));
  for (let i = 0; i < n; i++) {
    myArray[i][i] = 1;
  }
  return myArray;
}
```

[ ]:
```
console.log(identityMatrix(4));
```

```
[ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ]
```

11. Write a JavaScript function which will take an array of numbers stored and find the second lowest and second greatest numbers, respectively. Sample array: [1,2,3,4,5] Expected Output: 2,4

[ ]:
```
function find2ndMinMax(arr){
    let maxNum = -Infinity, secondMax = -Infinity
    let minNum = Infinity, secondMin = Infinity
    for(let a of arr){
        if(a>maxNum){
            let tempt = maxNum
            maxNum = a
            secondMax = tempt
        }else if(a < maxNum && a>secondMax){
```

```
            secondMax = a
        }
        if(a<minNum){
            let tempt2 = minNum
            minNum = a
            secondMin = tempt2
        }else if(a>minNum && a <secondMin){
            secondMin = a
        }
    }
    if (secondMax===-Infinity || secondMin===Infinity){
        return "second max or second min doesn't exist."
    }
    return [secondMin, secondMax]
}
```

[ ]:
```
console.log(find2ndMinMax([1,2,3,4,5]))
```

```
[ 2, 4 ]
```

12. Write a JavaScript function which says whether a number is perfect. According to Wikipedia: In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum). Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself). Example: The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and $1 + 2 + 3 = 6$. Equivalently, the number 6 is equal to half the sum of all its positive divisors: $(1 + 2 + 3 + 6) / 2 = 6$. The next perfect number is $28 = 1 + 2 + 4 + 7 + 14$. This is followed by the perfect numbers 496 and 8128.

[ ]:
```
function perfectNumber(num) {
    let sqrt = Math.sqrt(num);
    let factors = [1];
    for (let i = 2; i <= sqrt; i++) {
      if (num % i == 0) {
        factors.push(i);
        if (i != num / i) {
          factors.push(num / i);
        }
      }
    }
    // console.log(factors)
    const factor_sum = factors.reduce((acc, curr) => acc+curr, 0);
    if (num === factor_sum) return true;
    else return false;
}
```

[ ]:
```
console.log(perfectNumber(28))
console.log(perfectNumber(296))
console.log(perfectNumber(496))
```

```
true
false
true
```

13. Write a JavaScript function to compute the factors of a positive integer.

[ ]:
```
function findFactors(num) {
  let sqrt = Math.sqrt(num);
  let res = [];
  for (let i = 0; i <= sqrt; i++) {
    if (num % i == 0) {
      res.push(i);
      if (i != num / i) {
        res.push(num / i);
      }
    }
  }
  return res.sort((a, b) => a - b);
}
```

[ ]:
```
console.log(findFactors(60));
```

```
[
   1,  2,  3,  4,  5,
   6, 10, 12, 15, 20,
  30, 60
]
```

14. Write a JavaScript function to convert an amount to coins. Sample function: amountTocoins(46, [25, 10, 5, 2, 1]) Here 46 is the amount. and 25, 10, 5, 2, 1 are coins. Output: 25, 10, 10, 1

[ ]:
```
function calCoins(amount, coins) {
    coins.sort((a,b) => b-a);
    const res = [];
    coins.forEach((coin) => {
        while(amount >= coin) {
            res.push(coin);
            amount -= coin;
        }
    });
    return res;
}
```

[ ]:
```
console.log(calCoins(46,[25, 10, 5, 2, 1]))
```

```
[ 25, 10, 10, 1 ]
```

15. Write a JavaScript function to compute the value of bn where n is the exponent and b is the bases. Accept b and n from the user and display the result.

[ ]:
```
function exponent(b, n) {
    let res = 1;
    const positiveExponent = Math.abs(n);
    for (let i = 0; i < positiveExponent; i++) {
      res *= b;
    }
    // If exponent is negative, return the reciprocal
    return n < 0 ? 1 / res : res;
  }
```

[ ]:
```
console.log(exponent(2,3))
console.log(exponent(4,-2))
```

```
8
0.0625
```

16. Write a JavaScript function to extract unique characters from a string. Example string: “thequickbrownfoxjumpsoverthelazydog” Expected Output: “thequickbrownfxjmpsvlazydg”

[ ]:
```
function extractCharacters(s){
    let set = new Set()
    let myString = ""
    for (let c of s){
        if (!set.has(c)){
            myString+=c
            set.add(c)
        }
    }
    return myString
}
```

[ ]:
```
console.log(extractCharacters("thequickbrownfoxjumpsoverthelazydog"))
```

```
thequickbrownfxjmpsvlazydg
```

17. Write a JavaScript function to get the number of occurrences of each letter in specified string.

[ ]:
```
function countFrequency(s) {
  let map = new Map();
  if (s === "" || s === null || s === undefined) {
    return "Check input again";
```

```
  }
  for (let c of s) {
    if (map.has(c)) {
      map.set(c, map.get(c) + 1);
    } else {
      map.set(c, 1);
    }
  }
  return map;
}
```

[ ]:
```
console.log(countFrequency("happy happi"))
console.log(countFrequency(null))
```

```
Map(6) { 'h' => 2, 'a' => 2, 'p' => 4, 'y' => 1, ' ' => 1, 'i' => 1 }
Check input again
```

18. Write a function for searching JavaScript arrays with a binary search.

[ ]:
```
function binary(target, arr) {
  arr.sort((a,b) => a-b)
  let left = 0;
  let right = arr.length-1;
  while (left <= right) {
      let mid = Math.floor((left+right)/2);
      if (target < arr[mid]) {
          right = mid-1;
      } else if (target > arr[mid]) {
          left = mid+1;
      } else {
          return mid;
      }
  }
  return -1;
}
```

[ ]:
```
console.log(binary(5,[1,2,5,8,9]))
```

```
2
```

19. Write a JavaScript function that returns array elements larger than a number.

[ ]:
```
function largerElements(arr,num){
    let res = []
    for(let i of arr){
        if (i > num){
            res.push(i)
        }
```

```
    }
    return res
}
```

[ ]:
```
console.log(largerElements([1,2,3,4,5,10,12], 4))
```

```
[ 5, 10, 12 ]
```

20. Write a JavaScript function that generates a string id (specified length) of random characters. Sample character list: "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"

[ ]:
```
function randomString(len) {
    const chars =␣
 ↪'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
    let res = '';
    for (let i=0; i<len; i++){
        res += chars[Math.floor(Math.random() * chars.length)];
    }
    return res;
}
```

[ ]:
```
console.log(randomString(5))
```

```
QBmwr
```

21. Write a JavaScript function to get all possible subset with a fixed length (for example 2) combinations in an array. Sample array: [1, 2, 3] and subset length is 2 Expected output: [[2, 1], [3, 1], [3, 2]]

[ ]:
```
function subset(arr, len) {
    const res = [];
    function backtrack(i, curr) {
        if (curr.length == len) {
            res.push(curr);
            return;
        }
        for (let j=i; j<arr.length; j++){
            curr.push(arr[j]);
            backtrack(j+1, [...curr]);
            curr.pop();
        }
    }
    backtrack(0, []);
    for (l of res){
        l.sort((a, b) => b - a)
    }
    return res
```

```
}
```

[ ]:
```
console.log(subset([1,2,3],2))
```

```
[ [ 2, 1 ], [ 3, 1 ], [ 3, 2 ] ]
```

22. Write a JavaScript function that accepts two arguments, a string and a letter and the function will count the number of occurrences of the specified letter within the string. Sample arguments: 'microsoft.com', 'o' Expected output: 3

[ ]:
```
function countLetterOccurrences(str, letter) {
    let count = 0;
    for (let i = 0; i < str.length; i++) {
      if (str[i] === letter) {
        count++;
      }
    }
    return count;
  }
  console.log(countLetterOccurrences('microsoft.com', 'o'));
```

```
3
```

23. Write a JavaScript function to find the first not repeated character. Sample arguments: 'abacddbec' Expected output: 'e'

[ ]:
```
function firstUnqiueCharacter(s){
    let map = new Map()
    for(let i in s){
        if(!map.has(s[i])){
            map.set(s[i],1)
        }else{
            map.set(s[i],map.get(s[i])+1)
        }
    }
    // console.log(map)

    for (let char of s) {
        if (map.get(char) === 1) {
            return char;
        }
    }
    return -1 // can't find

}
```

[ ]:
```
console.log(firstUnqiueCharacter("abacddbec"))
```

```
e
```

24. Write a JavaScript function to apply Bubble Sort algorithm.

[ ]:
```
function bubbleSort(arr){
    for (let i = 0; i<arr.length;i++){
        swapped = false
        for (let j = 0; j<arr.length-1-i;j++){
            if (arr[j]< arr[j+1]){
                let tempt = arr[j]
                arr[j] = arr[j+1]
                arr[j+1] = tempt
                swapped= true
            }
        }
        if (swapped == false){
            break
        }
    }
    return arr

}
```

[ ]:
```
console.log(bubbleSort([6, 0, 3, 5]))
console.log(bubbleSort([12, 345, 4, 546, 122, 84, 98, 64, 9, 1, 3223, 455, 23,␣
 ↪234, 213]))
```

```
[ 6, 5, 3, 0 ]
[
  3223, 546, 455, 345, 234,
   213, 122,  98,  84,  64,
    23,  12,   9,   4,   1
]
```

25. Write a JavaScript function that accept a list of country names as input and returns the longest country name as output. Sample function: Longest_Country_Name([“Australia”, “Germany”, “United States of America”]) Expected output: “United States of America”

[ ]:
```
function longestCountryName(countries){
    if (countries === [] || countries === null || countries === undefined) {
        return "Invalid Input"
    }
    let maxLength = 0
    let maxLengthCountry = ""
    for (let country of countries){
        if(maxLength<country.length){
            maxLength = country.length
            maxLengthCountry = country
        }
    }
```

```
    return maxLengthCountry
}
```

[ ]:
```
console.log(longestCountryName(["Australia", "Germany", "United States of␣
↪America"]))
```

```
United States of America
```

26. Write a JavaScript function to find longest substring in a given a string without repeating characters.

[ ]:
```
function longestSubstring(s){
    let set = new Set()
    let maxLength = -1
    let res=""
    for (let i = 0;i<s.length;i++){
        let tempt = s[i]
        set.add(s[i])
        for (let j = i+1; j<s.length;j++){
            if(!set.has(s[j])){
                set.add(s[j])
                tempt+=s[j]
            }else{
                break
            }
        }
        if(tempt.length>maxLength){
            res = tempt
            maxLength = tempt.length
        }
    }
    return res
}
```

[ ]:
```
console.log(longestSubstring("abcdccde"))
console.log(longestSubstring("pwwkew"))
console.log(longestSubstring('bbbbb'))
```

```
abcd
wke
b
```

27. Write a JavaScript function that returns the longest palindrome in a given string. Note: According to Wikipedia “In computer science, the longest palindromic substring or longest symmetric factor problem is the problem of finding a maximum-length contiguous substring of a given string that is also a palindrome. For example, the longest palindromic substring of”bananas” is “anana”. The longest palindromic substring is not guaranteed to be unique; for

example, in the string “abracadabra”, there is no palindromic substring with length greater than three, but there are two palindromic substrings with length three, namely, “aca” and “ada”. In some applications it may be necessary to return all maximal palindromic substrings (that is, all substrings that are themselves palindromes and cannot be extended to larger palindromic substrings) rather than returning only one substring or returning the maximum length of a palindromic substring.

[ ]:
```
function longestPalindromeSubstring(str) {
    let res = [''];
    for (let i=0; i<str.length; i++) {
        let j = str.length-1;
        while(i<=j) {
            if (str[i] === str[j] && isPalindrome(str.slice(i,j+1)) && !res.
 ↪includes(str.slice(i,j+1))) {
                if (j-i+1 > res[0].length) //reuse isPalindrome earlier
                    res = [str.slice(i,j+1)];
                else if (j-i+1 === res[0].length)
                    res.push(str.slice(i,j+1));
                break;
            } else j --;
        }
    }
    return res
}
```

[ ]:
```
console.log(longestPalindromeSubstring("bananas"))
console.log(longestPalindromeSubstring("abracadabra"))
```

```
[ 'anana' ]
[ 'aca', 'ada' ]
```

28. Write a JavaScript program to pass a ‘JavaScript function’ as parameter.

[ ]:
```
function processData(data, callback) {
    console.log('Processing data:', data);
    callback(data);
  }
```

29. Write a JavaScript function to get the function name.

[ ]:
```
get_name = (func) => func.name;
function double(x){
    return x*2
}
console.log(get_name(double))
```

```
double
```