JavaScript Problem Set 5 (PS5) Solutions

Problem 1   Find the Pair (Two Sum Revisited)

Write a function `f_positions/f_values(arr, num)` that returns the two positions/values of the array "arr" that add up to the value of "num".

Solution:

```
  // this one gives you the values of the array that add up to the
     number

function f_positions(arr, num) {
    let empty = [];
    for (let k = 0; k < arr.length; k++) {
      for (let m = k + 1; m < arr.length; m++) {
        if (arr[k] + arr[m] === num) {
          empty.push(k, m);
        }
      }
    }

    return empty;
  }

  // this one gives you the positions of the numbers on the array
     that add up tto the number

  function f_values(arr, num) {
    let empty = [];
    for (let k = 0; k < arr.length; k++) {
      for (let m = k + 1; m < arr.length; m++) {
        if (arr[k] + arr[m] === num) {
          empty.push(arr[k], arr[m]);
        }
      }
    }

    return empty;
  }

console.log(f_values([2,7,88,99],9))

// both of these work for all numbers in the real line.
```

Problem 2 Max Sum Subarray of Size K

Write `maxSubarraySum(arr, k)` that returns the maximum sum of a contiguous subarray of size k. (Sliding window problem)

Solution:

```
function maxSubarraySum(arr, k) {
  if (arr.length < k) return null;
  let maxSum = 0;
  let windowSum = 0;
  for (let i = 0; i < k; i++) {
    windowSum += arr[i];
  }
  maxSum = windowSum;
  for (let end = k; end < arr.length; end++) {
    windowSum = windowSum - arr[end - k] + arr[end];
    maxSum = Math.max(maxSum, windowSum);
  }
  return maxSum;
}

console.log(maxSubarraySum([2, 1, 5, 1, 3, 2], 3)); // Output: 9
console.log(maxSubarraySum([1, 9, 2, 4, 6, 2], 2)); // Output: 11
```

Problem 3 Anagram Checker

Write a function `isAnagram(str1, str2)` that checks if two strings are anagrams.

Solution:

```
function isAnagram(str1, str2) {
return str1.split('').sort().join('') === str2.split('').sort().join
   ('');
}
```

Problem 4 Move Zeroes to End

Write a function `moveZeroes(arr)` that moves all 0 values to the end while keeping the order of the non-zero values the same.

Solution:

```
function mz(arr) {
  let result = [];
  for (let n of arr) {
    if (n !== 0) {
      result.push(n);
    }
  }
  while (result.length < arr.length) {
    result.push(0);
  }
  return result;
}

console.log(mz([0, 1, 0, 3, 12]));
```

Problem 5 Merge Two Sorted Arrays

Write `mergeSorted(arr1, arr2)` that merges and sorts two arrays.
Solution:

```
function mergeSorted(arr1, arr2) {
let i = 0, j = 0, result = [];
while (i < arr1.length && j < arr2.length) {
if (arr1[i] < arr2[j]) result.push(arr1[i++]);
else result.push(arr2[j++]);
}
return result.concat(arr1.slice(i)).concat(arr2.slice(j));
}
```

Problem 6 Promise Timeout

Create `wait(ms)` that resolves after `ms` milliseconds.
Solution:

```
function wait(ms) {
return new Promise(resolve => setTimeout(resolve, ms));
}
```

Problem 7 Count Vowels

Write `countVowels(str)` that counts vowels in a string.

Solution:

```
function cv(str) {
  let vowels = ["a", "e", "i", "o", "u", "A", "E", "I", "O", "U"];
  let count = 0;
  for (let char of str) {
    if (vowels.includes(char)) {
      count++;
    }
  }
  return count;
}

console.log(cv("Hello World"));
```

Problem 8 Product of Array Except Self

Write `pes(arr)` to return a new array with the product of all other elements.

Solution:

```
function pes(arr) {
  const n = arr.length;
  const result = [];
  for (let i = 0; i < n; i++) {
    let p = 1;
    for (let k = 0; k < n; k++) {
      if (k !== i) {p *= arr[k];}}
    result.push(p);
  }
  return result;
}

console.log(pes([1, 2, 3, 4]));
```

Problem 9 Find Missing Number

Given an array of integers ”arr”, create a function `fetch(arr)` that fetches all of the missing numbers from 1 to n and the amount of numbers missing.

Solution:

```
function fetch(arr) {
  let result = [];
  let n = Math.max(...arr);
  for (let i = 1; i <= n; i++) {
    if (!arr.includes(i)) {
      result.push(i);
    }
  }

  let count = result.length;
  return "There are " + count + " values, which are: " + result;
}

console.log(fetch([2, 3, 1, 7, 5]));
```

Problem 10 Async Fetch Mock

Write an async function `getUser()` that simulates an API fetch.

Solution:

```
async function getUser() {
await new Promise(resolve => setTimeout(resolve, 1000));
return "User fetched!";
}
```