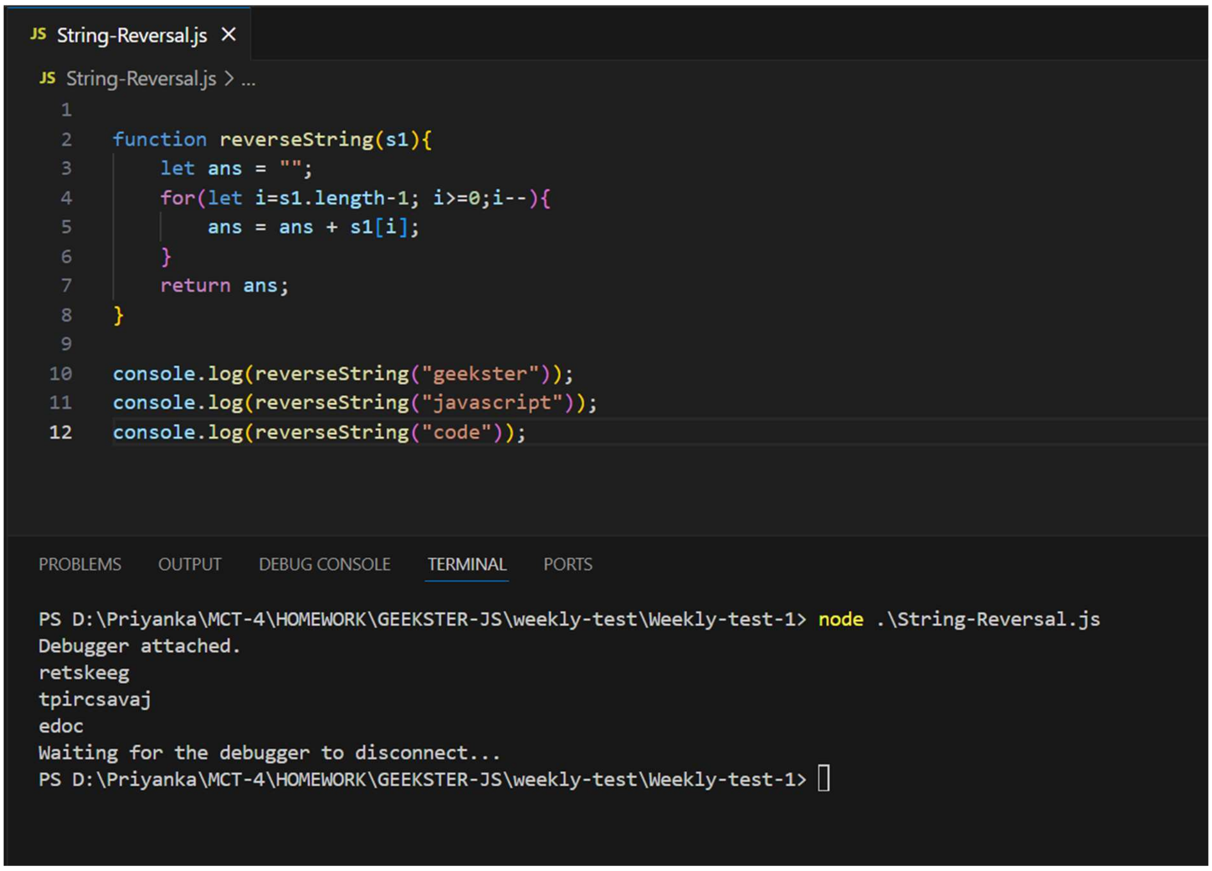
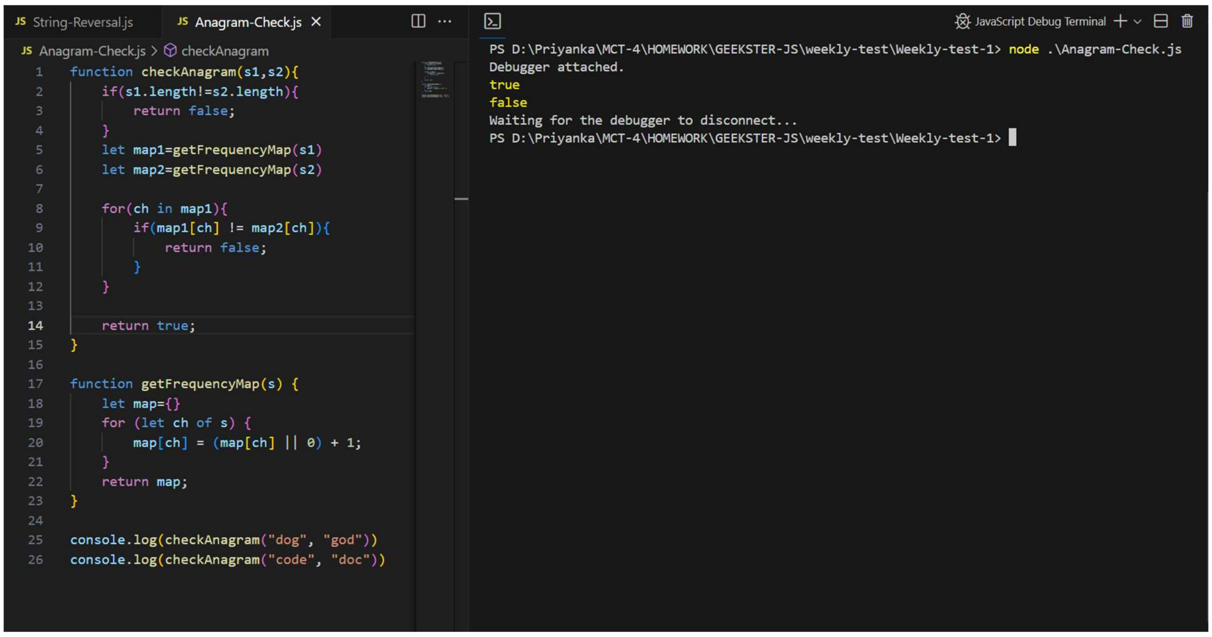
String Reversal: Write a function to reverse a given string in JavaScript without using built-in reverse functions.



```javascript

function reverseString(s1){
    let ans = "";
    for(let i=s1.length-1; i>=0;i--){
        ans = ans + s1[i];
    }
    return ans;
}

console.log(reverseString("geekster"));
console.log(reverseString("javascript"));
console.log(reverseString("code"));
```

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node .\String-Reversal.js
Debugger attached.
retskeeg
tpircsavaj
edoc
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```

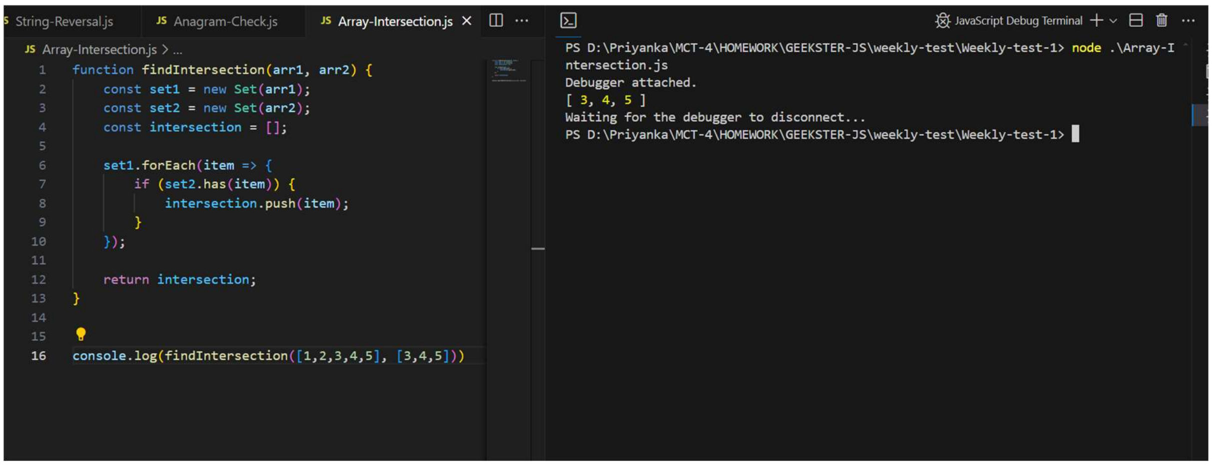
Anagram Check: Implement an algorithm to check if two strings are anagrams of each other (contain the same characters with the same frequency)



```javascript
function checkAnagram(s1,s2){
    if(s1.length!=s2.length){
        return false;
    }
    let map1=getFrequencyMap(s1)
    let map2=getFrequencyMap(s2)

    for(ch in map1){
        if(map1[ch] != map2[ch]){
            return false;
        }
    }

    return true;
}

function getFrequencyMap(s) {
    let map={}
    for (let ch of s) {
        map[ch] = (map[ch] || 0) + 1;
    }
    return map;
}

console.log(checkAnagram("dog", "god"))
console.log(checkAnagram("code", "doc"))
```

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node .\Anagram-Check.js
Debugger attached.
true
false
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```

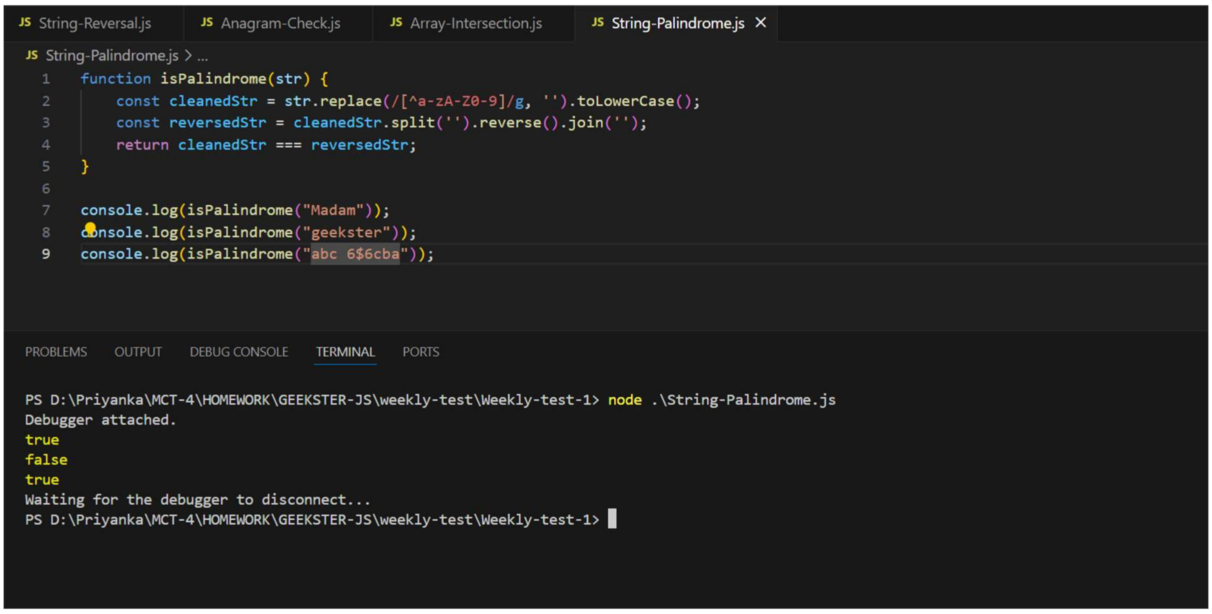
Array Intersection: Given two arrays, write a function to find their intersection (common elements).



```javascript
function findIntersection(arr1, arr2) {
    const set1 = new Set(arr1);
    const set2 = new Set(arr2);
    const intersection = [];

    set1.forEach(item => {
        if (set2.has(item)) {
            intersection.push(item);
        }
    });

    return intersection;
}


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

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node .\Array-I
ntersection.js
Debugger attached.
[ 3, 4, 5 ]
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```

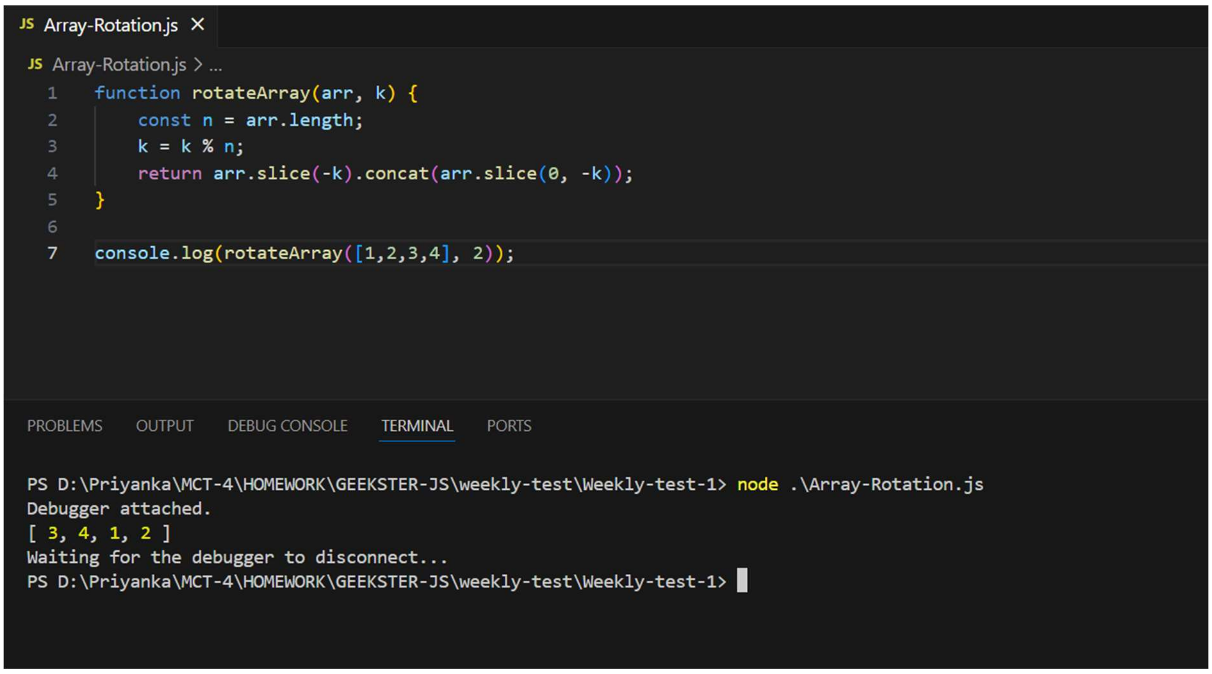
String Palindrome: Create a function to check if a given string is a palindrome (reads the same forwards and backwards) while ignoring non-alphanumeric characters.



```javascript
function isPalindrome(str) {
    const cleanedStr = str.replace(/[^a-zA-Z0-9]/g, '').toLowerCase();
    const reversedStr = cleanedStr.split('').reverse().join('');
    return cleanedStr === reversedStr;
}

console.log(isPalindrome("Madam"));
console.log(isPalindrome("geekster"));
console.log(isPalindrome("abc 6$6cba"));
```

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node .\String-Palindrome.js
Debugger attached.
true
false
true
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```

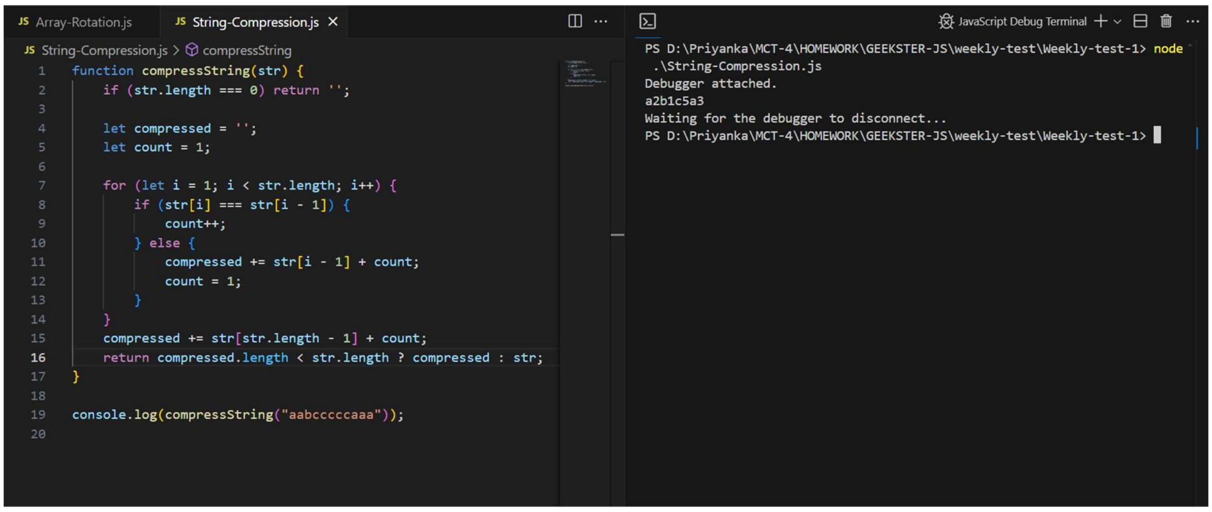
Array Rotation: Implement a function to rotate an array to the right by a specified number of positions.



```javascript
function rotateArray(arr, k) {
    const n = arr.length;
    k = k % n;
    return arr.slice(-k).concat(arr.slice(0, -k));
}

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

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node .\Array-Rotation.js
Debugger attached.
[ 3, 4, 1, 2 ]
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```

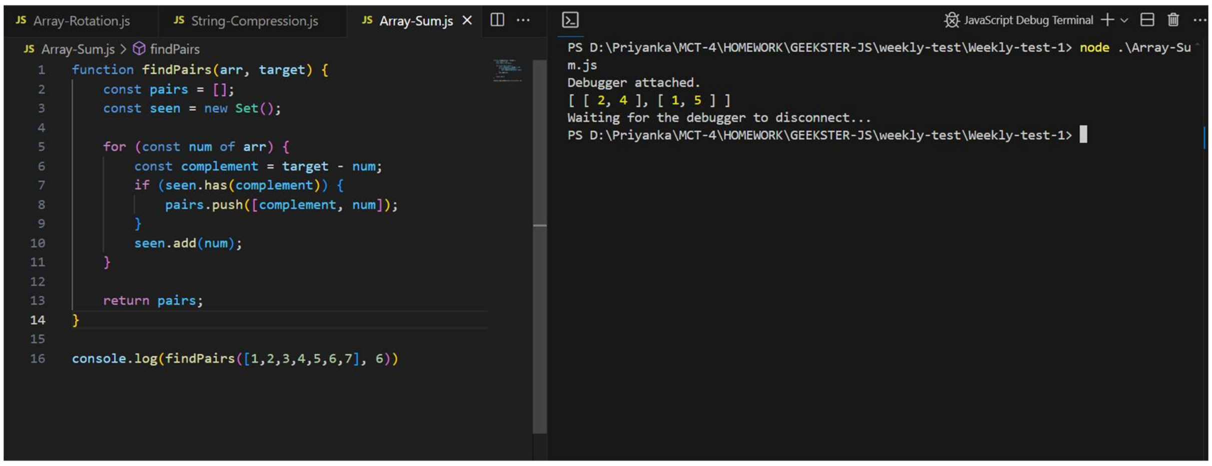
String Compression: Write a function to perform basic string compression using the counts of repeated characters. For example, "aabcccccaaa" would become "a2b1c5a3."


Array-Rotation.js
String-Compression.js
String-Compression.js > compressString
JavaScript Debug Terminal

```javascript
function compressString(str) {
    if (str.length === 0) return '';

    let compressed = '';
    let count = 1;

    for (let i = 1; i < str.length; i++) {
        if (str[i] === str[i - 1]) {
            count++;
        } else {
            compressed += str[i - 1] + count;
            count = 1;
        }
    }
    compressed += str[str.length - 1] + count;
    return compressed.length < str.length ? compressed : str;
}

console.log(compressString("aabcccccaaa"));
```

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node
 .\String-Compression.js
Debugger attached.
a2b1c5a3
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```

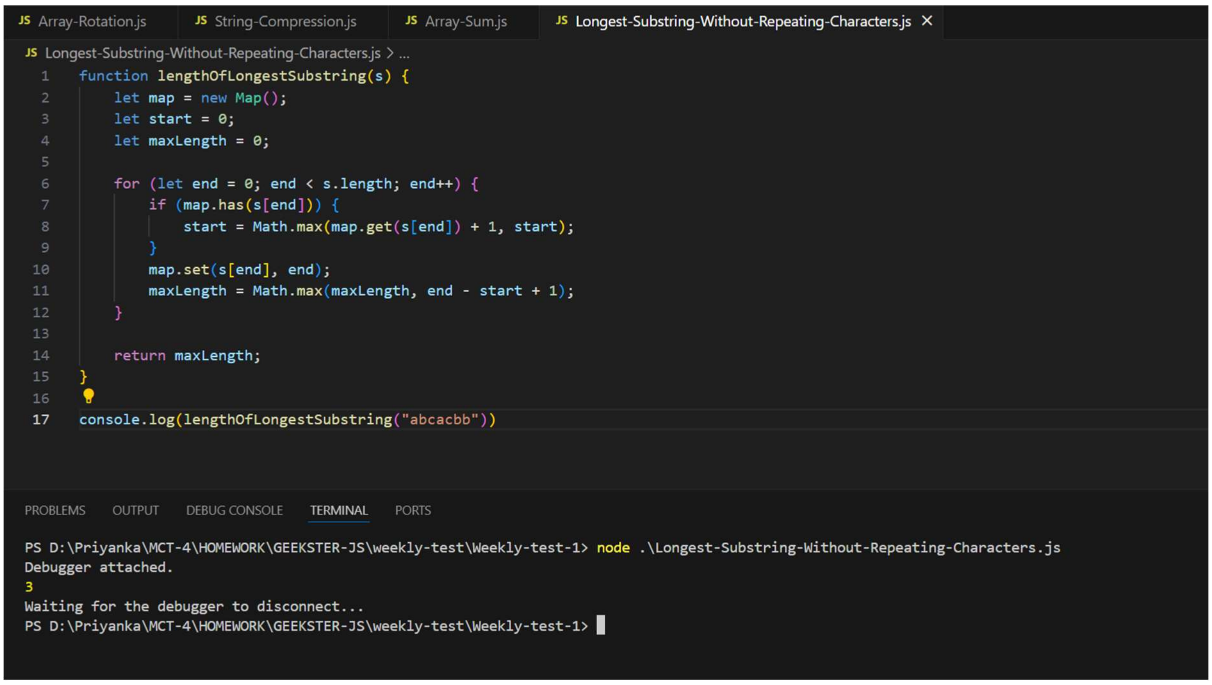
Array Sum: Write an algorithm to find the pair of elements in an array that adds up to a specific target sum.



```javascript
function findPairs(arr, target) {
    const pairs = [];
    const seen = new Set();

    for (const num of arr) {
        const complement = target - num;
        if (seen.has(complement)) {
            pairs.push([complement, num]);
        }
        seen.add(num);
    }

    return pairs;
}

console.log(findPairs([1,2,3,4,5,6,7], 6))
```

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node .\Array-Su
m.js
Debugger attached.
[ [ 2, 4 ], [ 1, 5 ] ]
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```

Longest Substring Without Repeating Characters: Write an algorithm to find the length of the longest substring without repeating characters in a given string.



Array-Rotation.js
String-Compression.js
Array-Sum.js
Longest-Substring-Without-Repeating-Characters.js
Longest-Substring-Without-Repeating-Characters.js > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```javascript
function lengthOfLongestSubstring(s) {
    let map = new Map();
    let start = 0;
    let maxLength = 0;

    for (let end = 0; end < s.length; end++) {
        if (map.has(s[end])) {
            start = Math.max(map.get(s[end]) + 1, start);
        }
        map.set(s[end], end);
        maxLength = Math.max(maxLength, end - start + 1);
    }

    return maxLength;
}

console.log(lengthOfLongestSubstring("abcacbb"))
```

```
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1> node .\Longest-Substring-Without-Repeating-Characters.js
Debugger attached.
3
Waiting for the debugger to disconnect...
PS D:\Priyanka\MCT-4\HOMEWORK\GEEKSTER-JS\weekly-test\Weekly-test-1>
```