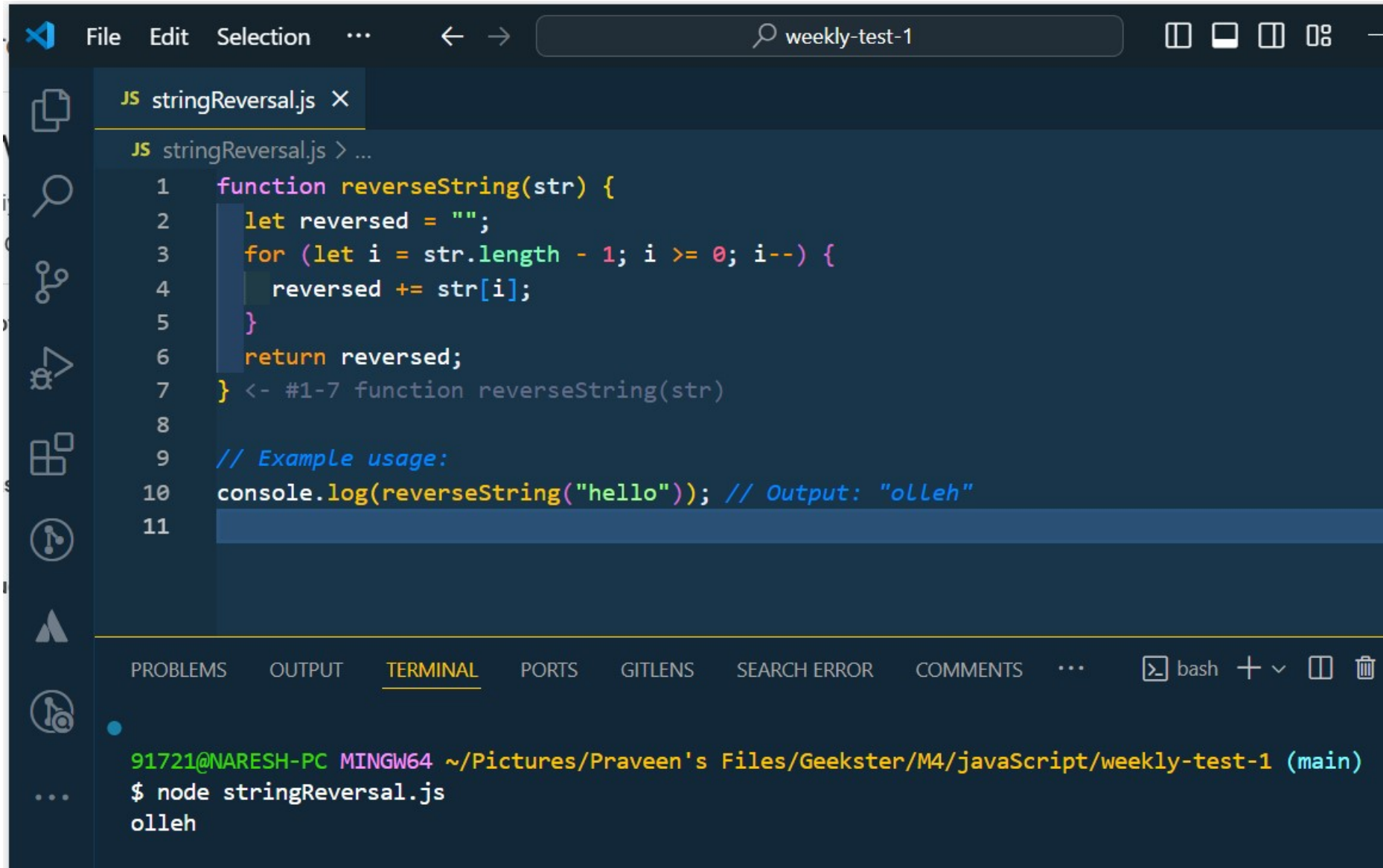


Note : "All the solutions to the questions are provided, and we also have the output of these questions in the terminal."

1.) String Reversal :



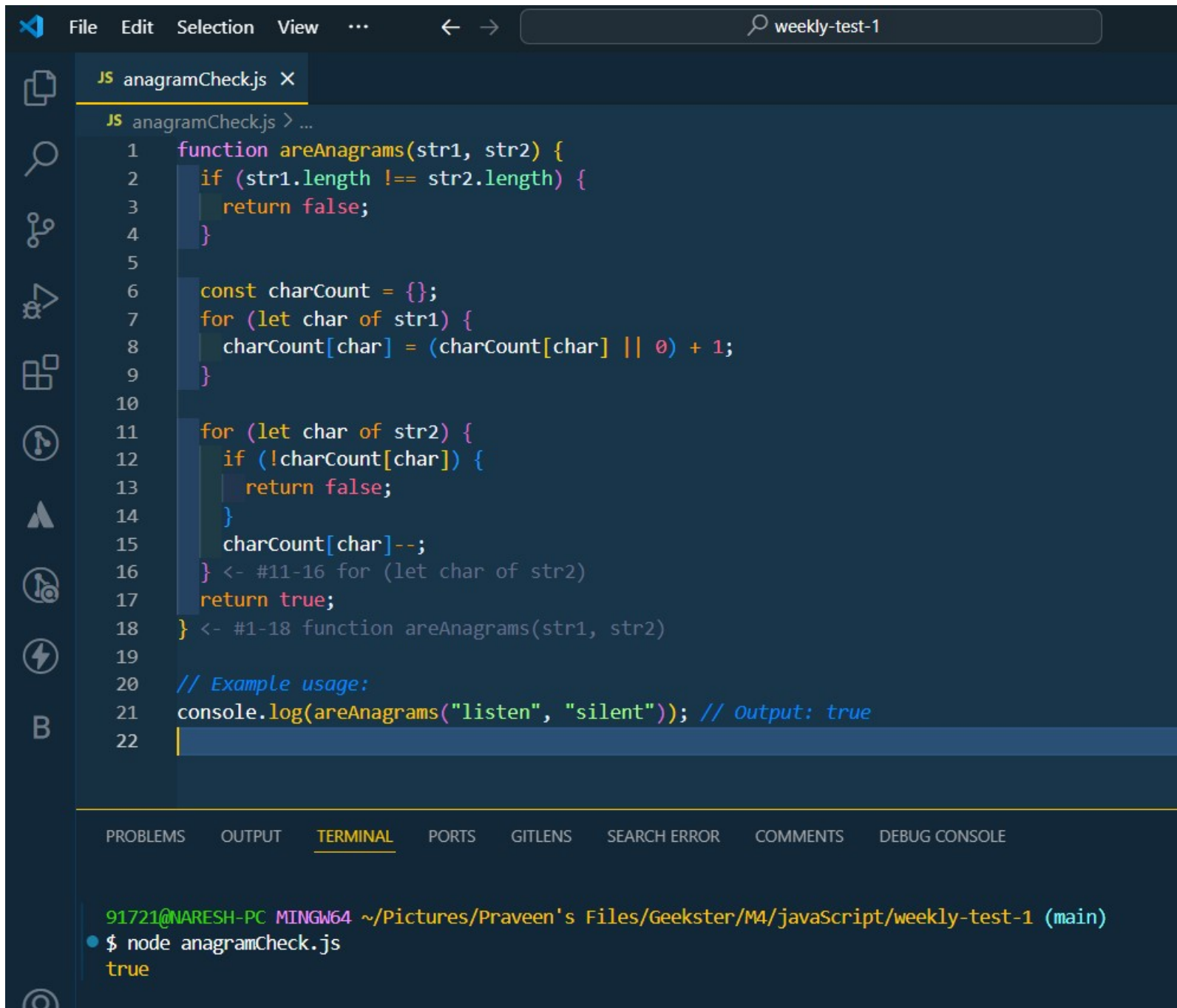
The screenshot shows the Visual Studio Code editor interface. The top bar includes the menu (File, Edit, Selection, ...), navigation arrows, a search bar with the text 'weekly-test-1', and window management icons. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, Extensions, and Testing. The main editor area displays the file 'stringReversal.js' with the following JavaScript code:

```
1 function reverseString(str) {  
2   let reversed = "";  
3   for (let i = str.length - 1; i >= 0; i--) {  
4     reversed += str[i];  
5   }  
6   return reversed;  
7 } <- #1-7 function reverseString(str)  
8  
9 // Example usage:  
10 console.log(reverseString("hello")); // Output: "olleh"  
11
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
91721@NARESH-PC MINGW64 ~/Pictures/Praveen's Files/Geekster/M4/javascript/weekly-test-1 (main)  
$ node stringReversal.js  
olleh
```

2.) Anagram Check :



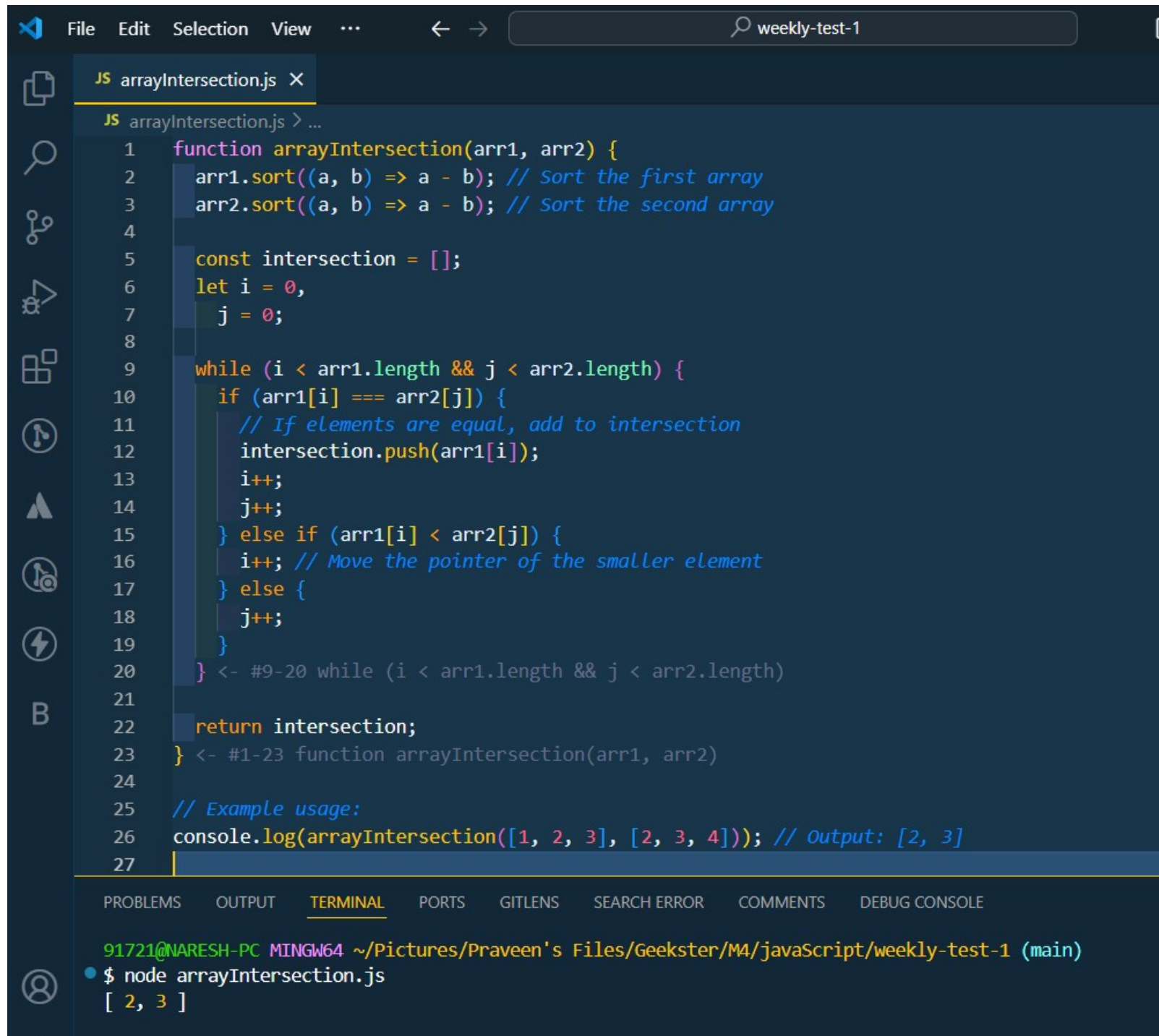
The image shows a code editor with a dark theme. The top bar includes a menu (File, Edit, Selection, View, ...), navigation arrows, and a search bar containing 'weekly-test-1'. The editor window displays a JavaScript file named 'anagramCheck.js'. The code defines a function 'areAnagrams' that checks if two strings are anagrams by comparing their lengths and character counts. Below the function, there is an example usage: 'console.log(areAnagrams("listen", "silent")); // Output: true'. The bottom panel shows the 'TERMINAL' tab with the command '\$ node anagramCheck.js' and its output 'true'.

```
JS anagramCheck.js X
JS anagramCheck.js > ...
1 function areAnagrams(str1, str2) {
2   if (str1.length !== str2.length) {
3     return false;
4   }
5
6   const charCount = {};
7   for (let char of str1) {
8     charCount[char] = (charCount[char] || 0) + 1;
9   }
10
11   for (let char of str2) {
12     if (!charCount[char]) {
13       return false;
14     }
15     charCount[char]--;
16   } <- #11-16 for (let char of str2)
17   return true;
18 } <- #1-18 function areAnagrams(str1, str2)
19
20 // Example usage:
21 console.log(areAnagrams("listen", "silent")); // Output: true
22
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS SEARCH ERROR COMMENTS DEBUG CONSOLE

```
91721@NARESH-PC MINGW64 ~/Pictures/Praveen's Files/Geekster/M4/javascript/weekly-test-1 (main)
• $ node anagramCheck.js
true
```

3.) Array Intersection :



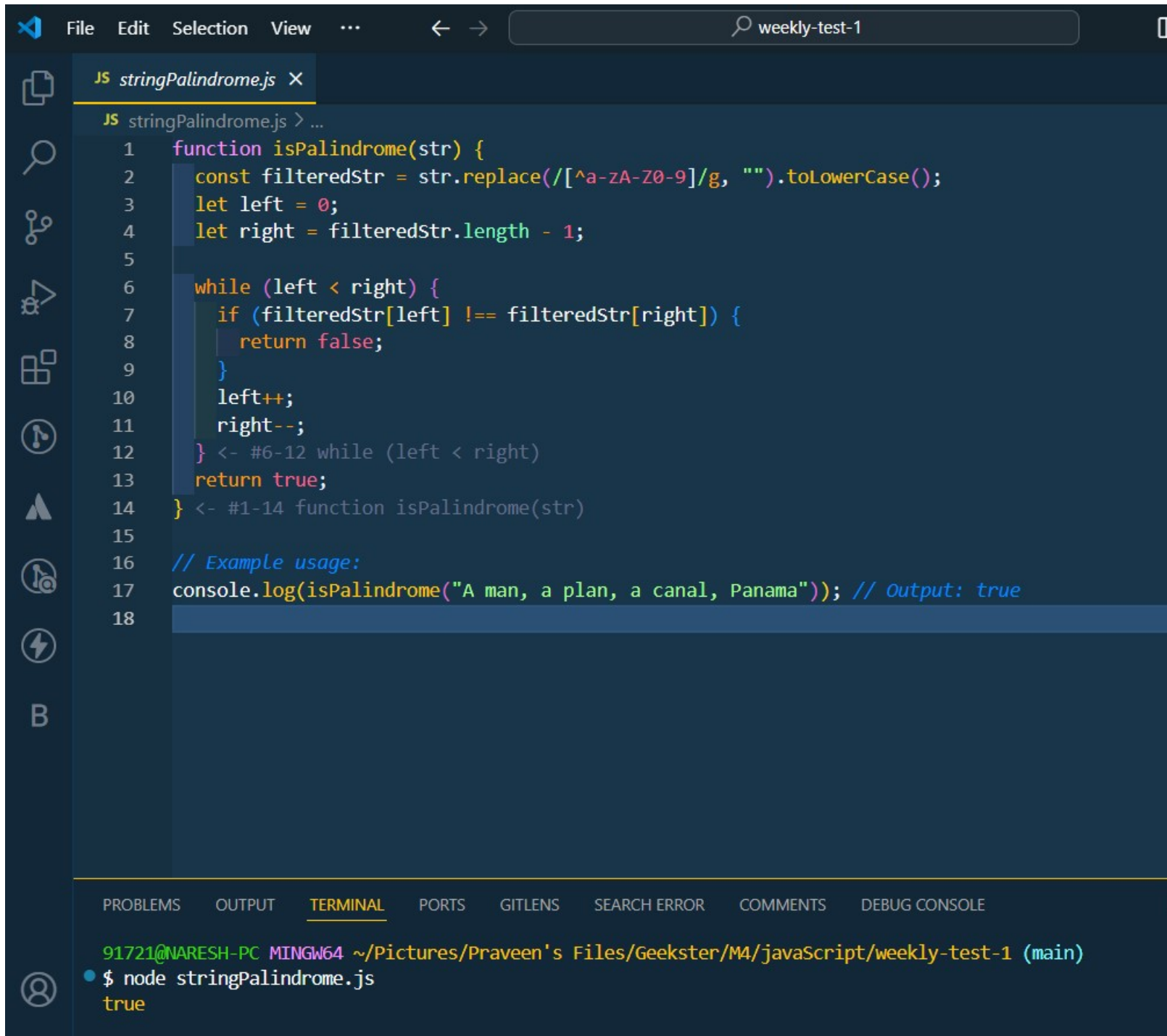
The image shows a code editor with a dark theme. The top bar includes a search icon and the text 'weekly-test-1'. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The main editor area displays a JavaScript file named 'arrayIntersection.js' with the following code:

```
1 function arrayIntersection(arr1, arr2) {  
2   arr1.sort((a, b) => a - b); // Sort the first array  
3   arr2.sort((a, b) => a - b); // Sort the second array  
4  
5   const intersection = [];  
6   let i = 0,  
7       j = 0;  
8  
9   while (i < arr1.length && j < arr2.length) {  
10    if (arr1[i] === arr2[j]) {  
11      // If elements are equal, add to intersection  
12      intersection.push(arr1[i]);  
13      i++;  
14      j++;  
15    } else if (arr1[i] < arr2[j]) {  
16      i++; // Move the pointer of the smaller element  
17    } else {  
18      j++;  
19    }  
20  }  
21  
22  return intersection;  
23 }  
24  
25 // Example usage:  
26 console.log(arrayIntersection([1, 2, 3], [2, 3, 4])); // Output: [2, 3]  
27
```

Below the code editor is a terminal window with the following output:

```
91721@NARESH-PC MINGW64 ~/Pictures/Praveen's Files/Geekster/M4/javascript/weekly-test-1 (main)  
$ node arrayIntersection.js  
[ 2, 3 ]
```

4.) String Palindrome :

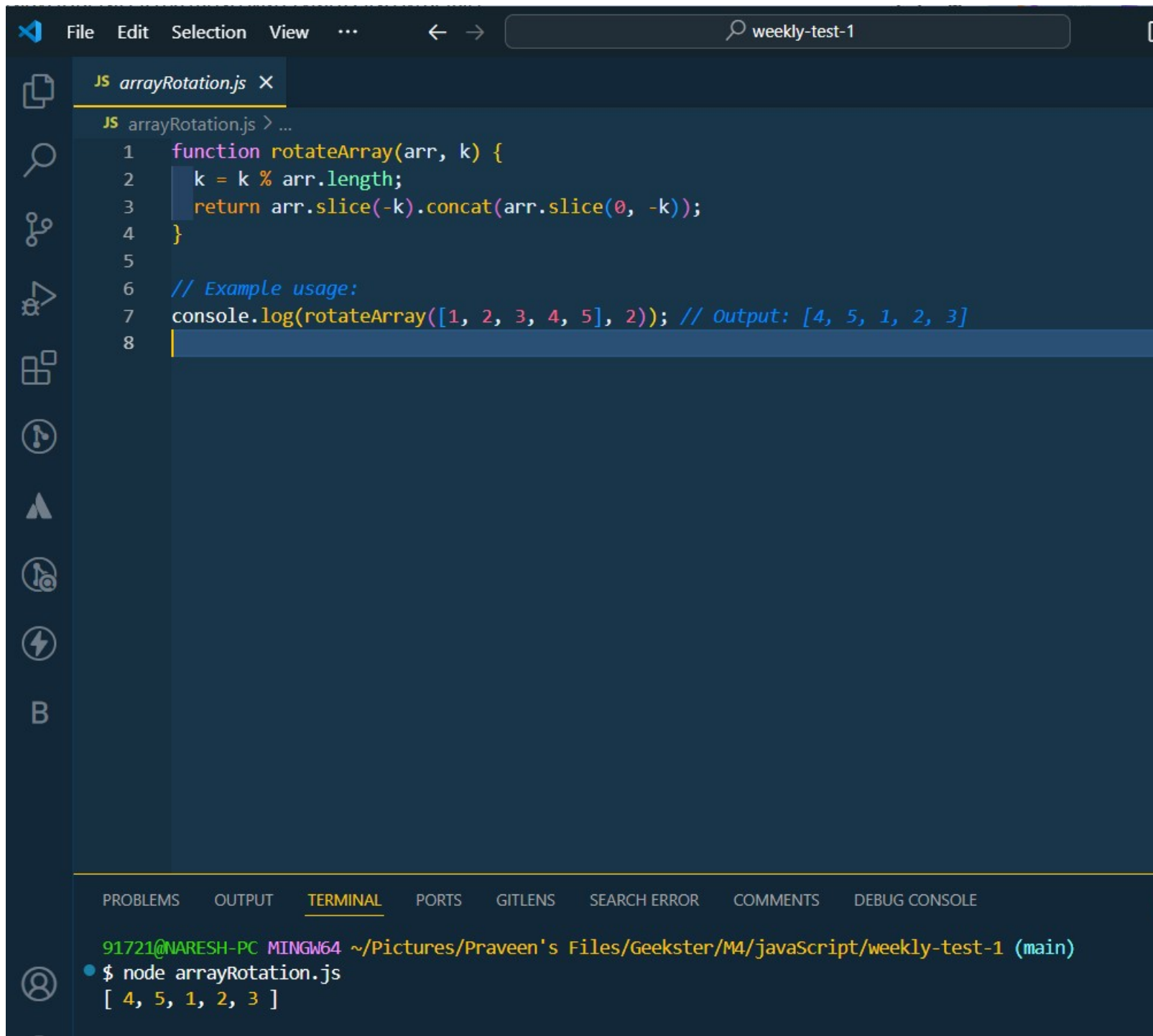


The image shows a VS Code editor window with a file named `stringPalindrome.js` open. The code defines a function `isPalindrome` that checks if a string is a palindrome. It filters out non-alphanumeric characters and converts the string to lowercase. Then, it uses two pointers, `left` and `right`, to compare characters from both ends of the string. If they don't match, it returns `false`; otherwise, it returns `true`. An example usage is provided, showing the function returning `true` for the string "A man, a plan, a canal, Panama".

```
JS stringPalindrome.js > ...
1  function isPalindrome(str) {
2      const filteredStr = str.replace(/[^a-zA-Z0-9]/g, "").toLowerCase();
3      let left = 0;
4      let right = filteredStr.length - 1;
5
6      while (left < right) {
7          if (filteredStr[left] !== filteredStr[right]) {
8              return false;
9          }
10         left++;
11         right--;
12     } <- #6-12 while (left < right)
13     return true;
14 } <- #1-14 function isPalindrome(str)
15
16 // Example usage:
17 console.log(isPalindrome("A man, a plan, a canal, Panama")); // Output: true
18
```

The terminal at the bottom shows the command `$ node stringPalindrome.js` being executed, resulting in the output `true`.

5.) Array Rotation :

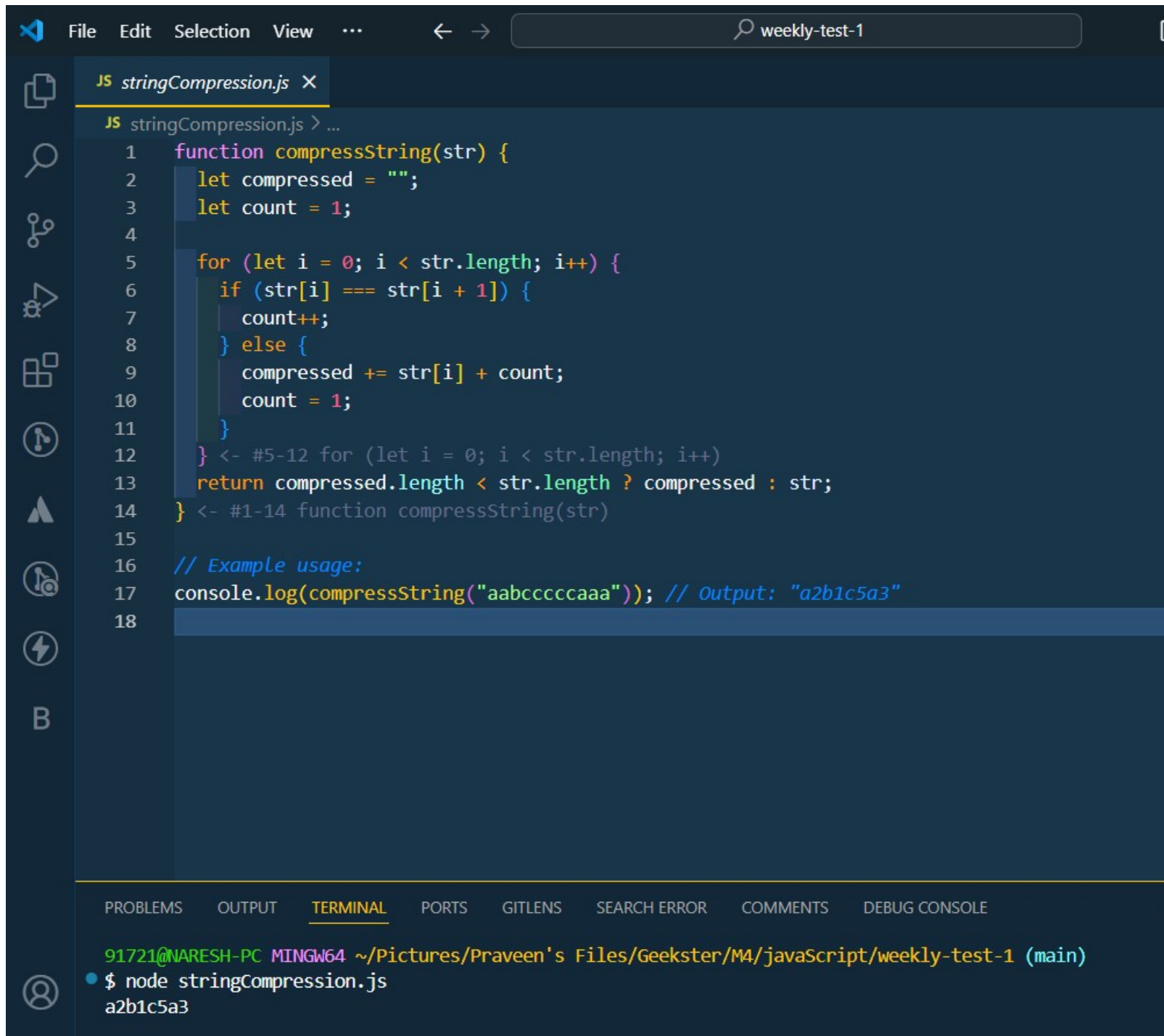


The image shows a Visual Studio Code editor window with a file named `arrayRotation.js`. The code defines a function `rotateArray` that rotates an array by `k` positions. The function uses `arr.slice(-k).concat(arr.slice(0, -k))` to achieve the rotation. An example usage is provided, showing the array `[1, 2, 3, 4, 5]` being rotated by 2 positions to produce `[4, 5, 1, 2, 3]`.

```
JS arrayRotation.js X
JS arrayRotation.js > ...
1 function rotateArray(arr, k) {
2   k = k % arr.length;
3   return arr.slice(-k).concat(arr.slice(0, -k));
4 }
5
6 // Example usage:
7 console.log(rotateArray([1, 2, 3, 4, 5], 2)); // Output: [4, 5, 1, 2, 3]
8
```

The terminal at the bottom shows the command `node arrayRotation.js` being executed, resulting in the output `[4, 5, 1, 2, 3]`.

6.) String Compression :

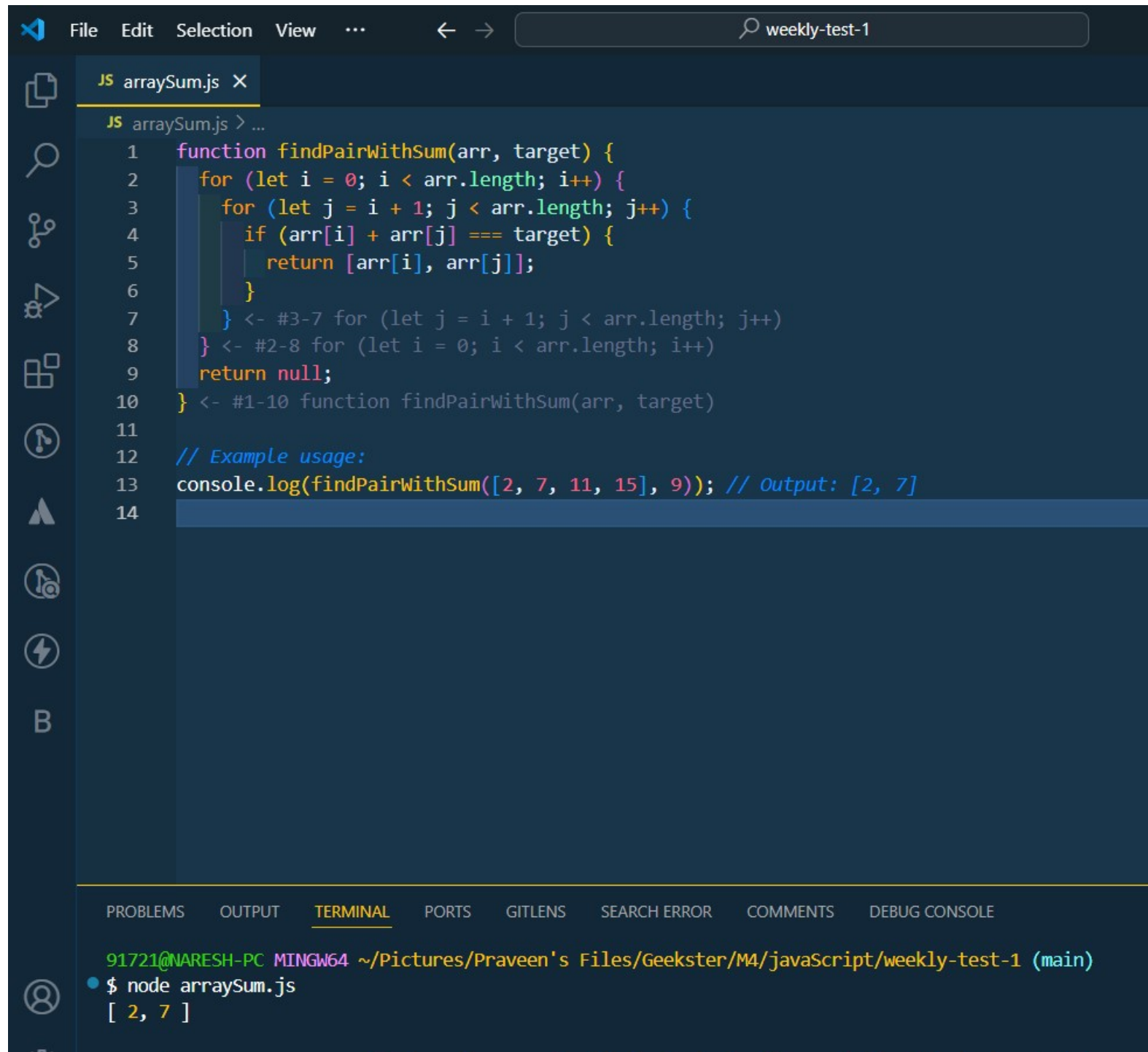


The image shows a VS Code editor window with a file named `stringCompression.js` open. The code defines a function `compressString` that takes a string `str` and returns a compressed version. The function iterates through the string, counting consecutive identical characters. For example, "aabcccccaaa" is compressed to "a2b1c5a3".

```
JS stringCompression.js X
JS stringCompression.js > ...
1 function compressString(str) {
2   let compressed = "";
3   let count = 1;
4
5   for (let i = 0; i < str.length; i++) {
6     if (str[i] === str[i + 1]) {
7       count++;
8     } else {
9       compressed += str[i] + count;
10      count = 1;
11    }
12  } <- #5-12 for (let i = 0; i < str.length; i++)
13  return compressed.length < str.length ? compressed : str;
14 } <- #1-14 function compressString(str)
15
16 // Example usage:
17 console.log(compressString("aabcccccaaa")); // Output: "a2b1c5a3"
18
```

The terminal at the bottom shows the command `$ node stringCompression.js` being executed, resulting in the output `a2b1c5a3`.

7.) Array Sum :

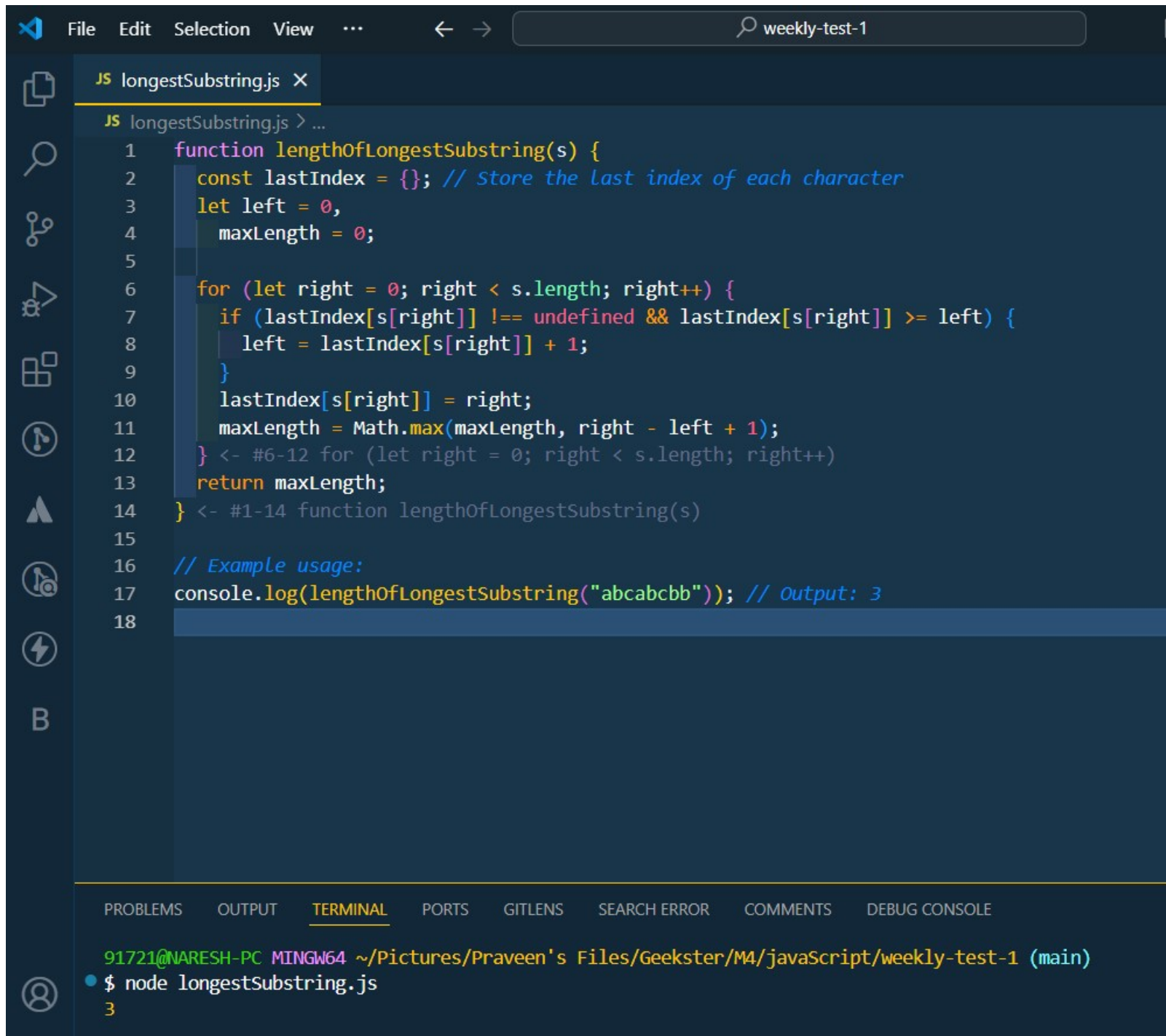


```
JS arraySum.js X
JS arraySum.js > ...
1 function findPairWithSum(arr, target) {
2   for (let i = 0; i < arr.length; i++) {
3     for (let j = i + 1; j < arr.length; j++) {
4       if (arr[i] + arr[j] === target) {
5         return [arr[i], arr[j]];
6       }
7     } <- #3-7 for (let j = i + 1; j < arr.length; j++)
8   } <- #2-8 for (let i = 0; i < arr.length; i++)
9   return null;
10 } <- #1-10 function findPairWithSum(arr, target)
11
12 // Example usage:
13 console.log(findPairWithSum([2, 7, 11, 15], 9)); // Output: [2, 7]
14
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS SEARCH ERROR COMMENTS DEBUG CONSOLE

```
91721@NARESH-PC MINGW64 ~/Pictures/Praveen's Files/Geekster/M4/javascript/weekly-test-1 (main)
$ node arraySum.js
[ 2, 7 ]
```

8.) Longest Substring Without Repeating Characters :



```
JS longestSubstring.js X
JS longestSubstring.js > ...
1  function lengthOfLongestSubstring(s) {
2      const lastIndex = {}; // Store the last index of each character
3      let left = 0,
4          maxLength = 0;
5
6      for (let right = 0; right < s.length; right++) {
7          if (lastIndex[s[right]] !== undefined && lastIndex[s[right]] >= left) {
8              left = lastIndex[s[right]] + 1;
9          }
10         lastIndex[s[right]] = right;
11         maxLength = Math.max(maxLength, right - left + 1);
12     } <- #6-12 for (let right = 0; right < s.length; right++)
13     return maxLength;
14 } <- #1-14 function lengthOfLongestSubstring(s)
15
16 // Example usage:
17 console.log(lengthOfLongestSubstring("abcabcbb")); // Output: 3
18
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS SEARCH ERROR COMMENTS DEBUG CONSOLE

```
91721@NARESH-PC MINGW64 ~/Pictures/Praveen's Files/Geekster/M4/javascript/weekly-test-1 (main)
$ node longestSubstring.js
3
```