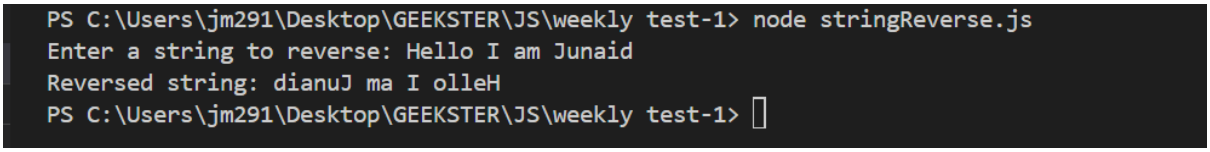
- **String Reversal:** Write a function to reverse a given string in JavaScript without using built-in reverse functions.

Code-

Output-



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

  Code-

```js
const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

function areAnagrams(str1, str2) {
    if (str1.length !== str2.length) {
        return false;
    }

    const charCount = {};

    for (let char of str1) {
        charCount[char] = (charCount[char] || 0) + 1;
    }

    for (let char of str2) {
        if (!charCount[char]) {
            return false;
        }
        charCount[char] -= 1;
    }

    for (let count in charCount) {
        if (charCount[count] !== 0) {
            return false;
        }
    }

    return true;
}

rl.question('Enter the first string: ', (str1) => {
    rl.question('Enter the second string: ', (str2) => {
        if (areAnagrams(str1, str2)) {
            console.log('The strings are anagrams.');
        } else {
            console.log('The strings are not anagrams.');
        }
        rl.close();
    });
});
```

  Output-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> & node anagramCheck.js
Enter the first string: geekster
Enter the second string: stergeek
The strings are anagrams.
PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> & node anagramCheck.js
Enter the first string: junaid
Enter the second string: malik
The strings are not anagrams.
PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1>
```

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

  Code-

```
JS arrayIntersection.js > ...
1   const readline = require('readline');
2
3   const rl = readline.createInterface({
4       input: process.stdin,
5       output: process.stdout
6   });
7
8   function findIntersection(arr1, arr2) {
9       const set1 = new Set(arr1);
10      const set2 = new Set(arr2);
11      const intersection = [];
12
13      set1.forEach(item => {
14          if (set2.has(item)) {
15              intersection.push(item);
16          }
17      });
18
19      return intersection;
20  }
21
22  function parseArrayInput(input) {
23      return input.split(',').map(Number).filter(n => !isNaN(n));
24  }
25
```

```
25
26  rl.question('Enter the first array (comma-separated values): ', (input1) => {
27      rl.question('Enter the second array (comma-separated values): ', (input2) => {
28          const arr1 = parseArrayInput(input1);
29          const arr2 = parseArrayInput(input2);
30
31          const intersection = findIntersection(arr1, arr2);
32          console.log('Intersection of the two arrays:', intersection);
33
34          rl.close();
35      });
36  });
```

  Output-

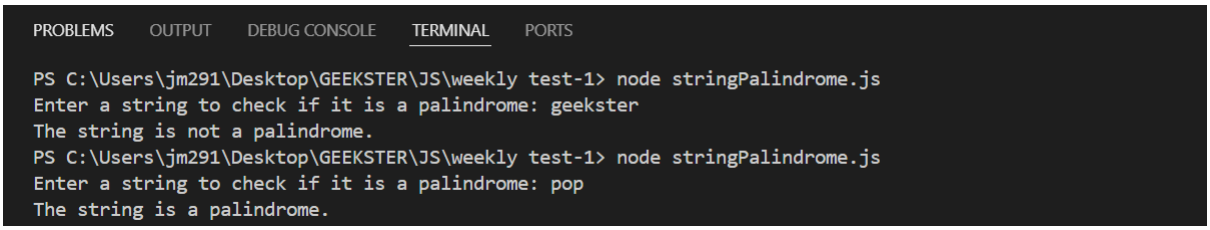
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Code

PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> node arrayIntersection.js
Enter the first array (comma-separated values): 2,5,7,9,4,8,4,8
Enter the second array (comma-separated values): 2,4,3,4,2,5,8,5,9
Intersection of the two arrays: [ 2, 5, 9, 4, 8 ]
PS C:\Users\jm291\Desktop\GEEKSTER\JS\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.

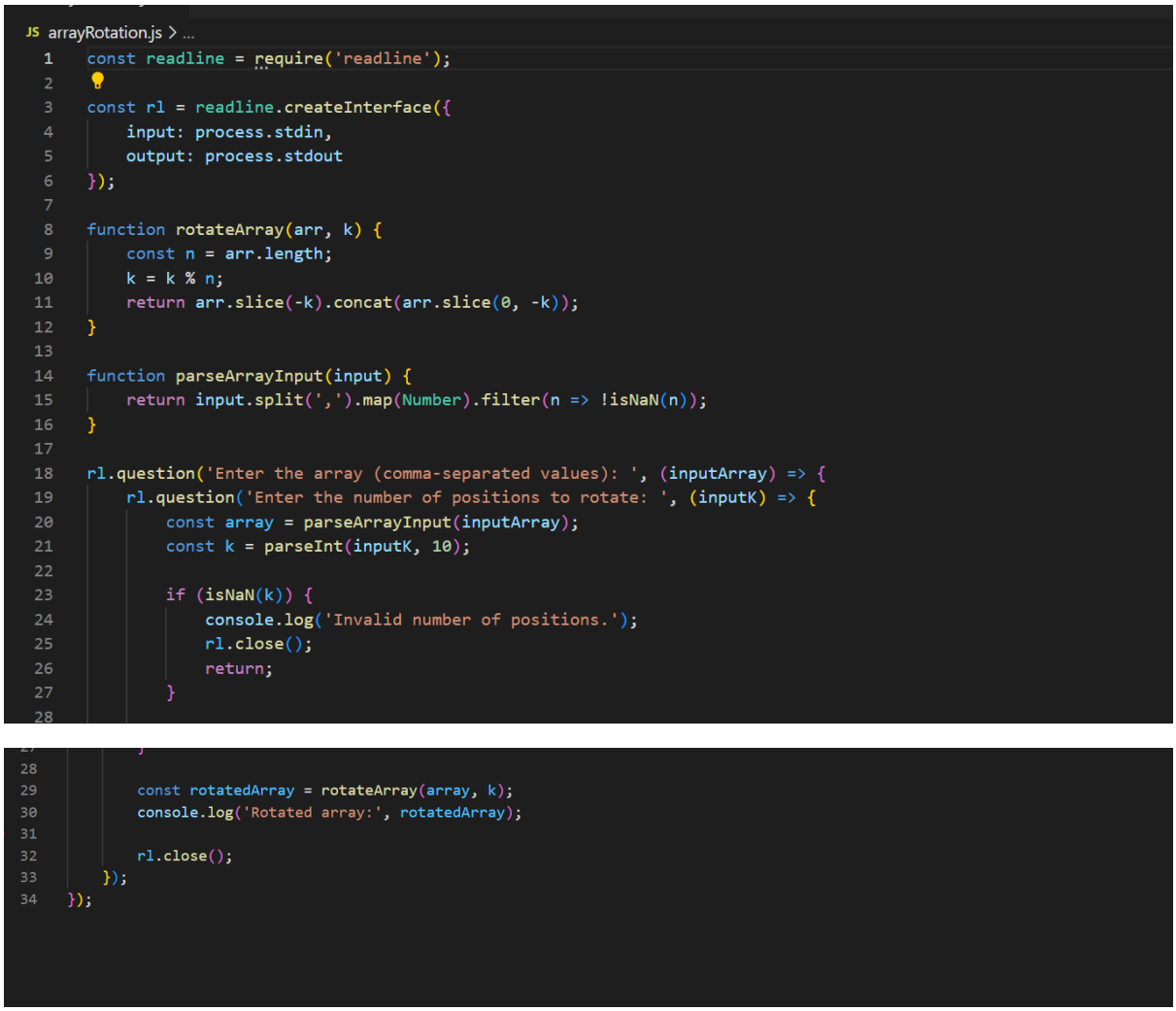
  Code-

  Output-



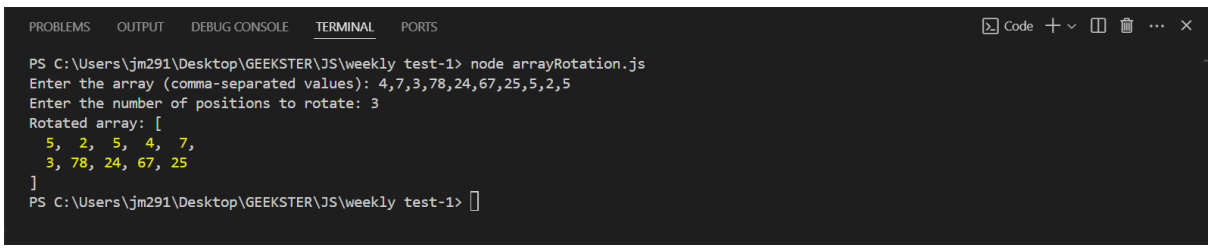
- **Array Rotation:** Implement a function to rotate an array to the right by a specified number of positions.

  Code-



```
JS arrayRotation.js > ...
1   const readline = require('readline');
2
3   const rl = readline.createInterface({
4       input: process.stdin,
5       output: process.stdout
6   });
7
8   function rotateArray(arr, k) {
9       const n = arr.length;
10      k = k % n;
11      return arr.slice(-k).concat(arr.slice(0, -k));
12  }
13
14  function parseArrayInput(input) {
15      return input.split(',').map(Number).filter(n => !isNaN(n));
16  }
17
18  rl.question('Enter the array (comma-separated values): ', (inputArray) => {
19      rl.question('Enter the number of positions to rotate: ', (inputK) => {
20          const array = parseArrayInput(inputArray);
21          const k = parseInt(inputK, 10);
22
23          if (isNaN(k)) {
24              console.log('Invalid number of positions.');
25              rl.close();
26              return;
27          }
28
29          const rotatedArray = rotateArray(array, k);
30          console.log('Rotated array:', rotatedArray);
31
32          rl.close();
33      });
34  });
```

  Output-



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Code

PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> node arrayRotation.js
Enter the array (comma-separated values): 4,7,3,78,24,67,25,5,2,5
Enter the number of positions to rotate: 3
Rotated array: [
  5,  2,  5,  4,  7,
  3, 78, 24, 67, 25
]
PS C:\Users\jm291\Desktop\GEEKSTER\JS\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."

  Code-

```
JS stringCompression.js > ...
1   const readline = require('readline');
2
3   const rl = readline.createInterface({
4       input: process.stdin,
5       output: process.stdout
6   });
7
8   function compressString(str) {
9       if (str.length === 0) return '';
10
11      let compressed = '';
12      let count = 1;
13
14      for (let i = 1; i < str.length; i++) {
15          if (str[i] === str[i - 1]) {
16              count++;
17          } else {
18              compressed += str[i - 1] + count;
19              count = 1;
20          }
21      }
22
23      compressed += str[str.length - 1] + count;
24
25      return compressed.length < str.length ? compressed : str;
26  }
27
```

```
26  }
27
28  rl.question('Enter a string to compress: ', (inputString) => {
29      const compressedString = compressString(inputString);
30      console.log('Compressed string:', compressedString);
31      rl.close();
32  });
```

  Output-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Code

PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> node stringCompression.js
Enter a string to compress: jjjuuuuuunnnnnaaaaaiiiiiidd
Compressed string: j3u6n5a5i6d2
PS C:\Users\jm291\Desktop\GEEKSTER\JS\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.

  Code-

```js
const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

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;
}

function parseArrayInput(input) {
    return input.split(',').map(Number).filter(n => !isNaN(n));
}

rl.question('Enter the array (comma-separated values): ', (inputArray) => {
    rl.question('Enter the target sum: ', (inputTarget) => {
        const array = parseArrayInput(inputArray);
        const target = parseInt(inputTarget, 10);

        if (isNaN(target)) {
            console.log('Invalid target sum.');
            rl.close();
            return;
        }

        const pairs = findPairs(array, target);
        if (pairs.length > 0) {
            console.log('Pairs that add up to the target sum:', pairs);
        } else {
            console.log('No pairs found that add up to the target sum.');
        }

        rl.close();
    });
});
```

  Output-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> node arraySum.js
Enter the array (comma-separated values): 3,4,8,33,32,66,-38,13
Enter the target sum: 34
No pairs found that add up to the target sum.
PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> node arraySum.js
Enter the array (comma-separated values): 2,3,4,5,6,7,34,24,2
Enter the target sum: 9
Pairs that add up to the target sum: [ [ 4, 5 ], [ 3, 6 ], [ 2, 7 ], [ 7, 2 ] ]
PS C:\Users\jm291\Desktop\GEEKSTER\JS\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.

  Code-

```javascript
const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

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;
}

rl.question('Enter a string to find the length of the longest substring without repeating characters: ', (inputString) => {
    const result = lengthOfLongestSubstring(inputString);
    console.log('Length of the longest substring without repeating characters:', result);
    rl.close();
});
```

  Output-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1> node longestSubstringWithoutRepeatingCharacters.js
Enter a string to find the length of the longest substring without repeating characters: jsdbiusjsssssddefdckndj
Length of the longest substring without repeating characters: 6
PS C:\Users\jm291\Desktop\GEEKSTER\JS\weekly test-1>
```