Github

1. String Compression

Problem Statement:
Implement a method to perform string compression. E.g. ‘aabcccccaaa’ should be compressed as ‘a2b1c5a3’. Build a second compressor to optimize further. The compressed output should be put into a decompressor.

Solution (JavaScript):

```
function compressString(str) {
    let compressed = "";
    let count = 1;
    for (let i = 0; i < str.length; i++) {
        if (str[i] === str[i + 1]) {
            count++;
        } else {
            compressed += str[i] + count;
            count = 1;
        }
    }
    return compressed;
}

function decompressString(str) {
    let decompressed = "";
    for (let i = 0; i < str.length; i += 2) {
        decompressed += str[i].repeat(parseInt(str[i + 1]));
    }
    return decompressed;
}

console.log(compressString("aabcccccaaa")); // Output: a2b1c5a3
console.log(decompressString("a2b1c5a3")); // Output: aabcccccaaa
```

2. Find the Middle Element of a Singly Linked List in One Pass

Solution (JavaScript):

```
class ListNode {
    constructor(value) {
        this.value = value;
        this.next = null;
    }
}

function findMiddle(head) {
    let slow = head, fast = head;
    while (fast !== null && fast.next !== null) {
        slow = slow.next;
        fast = fast.next.next;
    }
    return slow.value;
}
```

3. Trapping Rain Water Problem

Solution (JavaScript):

```
function trapRainWater(height) {
    let left = 0, right = height.length - 1;
    let leftMax = 0, rightMax = 0, water = 0;

    while (left < right) {
        if (height[left] < height[right]) {
            height[left] >= leftMax ? (leftMax = height[left]) :
water += leftMax - height[left];
            left++;
        } else {
            height[right] >= rightMax ? (rightMax = height[right]) :
water += rightMax - height[right];
            right--;
        }
    }
    return water;
}
```

4. Count Ways to Express a Number as the Sum of Consecutive Natural Numbers

Solution (JavaScript):

```
function countWays(n) {
    let count = 0;
    for (let start = 1; start <= n; start++) {
        let sum = 0;
        for (let j = start; sum < n; j++) {
            sum += j;
            if (sum === n) count++;
        }
    }
    return count;
}
```

5. Find the Largest 5-Digit Prime in the First 100 Digits of Pi

Solution (JavaScript):

```
function isPrime(num) {
    if (num < 2) return false;
    for (let i = 2; i * i <= num; i++) {
        if (num % i === 0) return false;
    }
    return true;
}

function findLargestPrimeInPi() {
    const piDigits = "1415926535... (100 digits)";
    let largestPrime = -1;
    for (let i = 0; i <= piDigits.length - 5; i++) {
        let num = parseInt(piDigits.substring(i, i + 5));
        if (isPrime(num) && num > largestPrime) largestPrime = num;
    }
    return largestPrime;
}
```

6. Check if Two Rectangles Intersect

Solution (JavaScript):

```
function doRectanglesIntersect(r1, r2) {
    let [x1, y1, x2, y2] = r1;
    let [a1, b1, a2, b2] = r2;
    return !(x2 < a1 || a2 < x1 || y2 < b1 || b2 < y1);
}
```

7. Social Media Application (MERN Stack)

- **Built a social media app with MERN Stack**
- Features: **Login, Signup, Like, Comment, Dark Mode**
- Used **React (Frontend), Node.js & Express.js (Backend), MongoDB (Database)**
- Implemented **JWT Authentication & WebSocket for real-time updates**

Example: **Like Post API (Node.js & Express.js)**

```
router.put("/like/:id", auth, async (req, res) => {
    try {
        const post = await Post.findById(req.params.id);
        if (!post) return res.status(404).json({ message: "Post not
found" });

        const alreadyLiked = post.likes.includes(req.user.id);
        if (alreadyLiked) {
            post.likes = post.likes.filter((id) => id !==
req.user.id);
        } else {
            post.likes.push(req.user.id);
        }

        await post.save();
        res.json(post);
    } catch (error) {
        res.status(500).json({ message: "Server Error" });
    }
});
```

8. Matrix Multiplication (n × n Matrices)

Solution (JavaScript):

```
function multiplyMatrices(A, B) {
    let n = A.length;
    let C = Array.from({ length: n }, () => Array(n).fill(0));

    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) {
            for (let k = 0; k < n; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return C;
}
```