a) String compression

Implement a method to perform string compression. E.g. ‘aabcccccaaa’ should be a2b1c5a3. The code to implement this is given in the link -
https://www.educative.io/answers/string-compression-using-run-length-encoding

Build a second compressor which takes the output of first compressor and optimizes further. The answer should be taken into second compressor and compress further.
E.g. a2b2c1a3c3 should become ab2c1ac3. Please test at the end cases as well. The code should work with inputs like a20b20c1a4 – ab20c1a4. Test and show the result of the same.
The compressed output should be put into a decompressor -
ab2c1ac3 should return aabbcaaaccc.
Write all the test cases to evaluate this and see if the results match.

ANSWER-

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

function optimizeCompression(s) {
    return s.replace(/(\D)1/g, "$1");
}

function decompressString(s) {
    return s.replace(/(\D)(\d*)/g, (match, char, num) => char.repeat(num ? parseInt(num) : 1));
}

function testCompression() {
    const testCases = [
        ["aabcccccaaa", "a2b1c5a3"],
        ["a20b20c1a4", "a20b20c1a4"],
        ["aaaabbc", "a4b2c1"],
```

```
    ];
    testCases.forEach(([input, expected]) => {
        console.assert(compressString(input) === expected, `Failed on ${input}`);
    });
    console.log("Compression passed.");
}

function testOptimization() {
    const testCases = [
        ["a2b2c1a3c3", "ab2c1ac3"],
        ["a20b20c1a4", "a20b20c1a4"],
    ];
    testCases.forEach(([input, expected]) => {
        console.assert(optimizeCompression(input) === expected, `Failed on ${input}`);
    });
    console.log("Optimization passed.");
}

function testDecompression() {
    const testCases = [
        ["ab2c1ac3", "aabbcaaaccc"],
        ["a20b20c1a4", "a".repeat(20) + "b".repeat(20) + "c" + "a".repeat(4)],
    ];
    testCases.forEach(([input, expected]) => {
        console.assert(decompressString(input) === expected, `Failed on ${input}`);
    });
    console.log("Decompression passed.");
}

function runTests() {
    testCompression();
    testOptimization();
    testDecompression();

    let sampleInput = "aabccccccaaa";
    let compressed = compressString(sampleInput);
    let optimized = optimizeCompression(compressed);
    let decompressed = decompressString(optimized);

    console.log(`Original: ${sampleInput}`);
    console.log(`Compressed: ${compressed}`);
    console.log(`Optimized: ${optimized}`);
    console.log(`Decompressed: ${decompressed}`);
}
```

runTests();

b) Linked List - How to find the middle element of a singly linked list in one pass?
ANSWER -

To find the **middle element** of a **singly linked list in one pass**, use the **slow and fast pointer approach**:

1. Initialize **slow** and **fast** pointers at `head`.
2. Move **slow** one step and **fast** two steps at a time.
3. When **fast** reaches the end, **slow** will be at the middle.

Time Complexity: O(N)
Space Complexity: O(1)

c) Given an array of integers representing the elevation of a roof structure at various positions, each position is separated by a unit length, Write a program to determine the amount of water that will be trapped on the roof after heavy rainfall
Example:
input : [2 1 3 0 1 2 3]
Ans : 7 units of water will be trapped

https://www.geeksforgeeks.org/trapping-rain-water/

Go through the above code for the solution.

The next phase is that the values are now not discrete but analog. E.g. I give an equation of function that is bounded, can you predict how many units of water gets trapped.

ANSWER -

For **continuous elevation (analog case)**:

1. Identify **local maxima and minima** to determine valleys.
2. Compute **water level** above each point.
3. Integrate **(water level - function height)** over the range.

Mathematically:

Trapped Water=∫ab(h(x)−f(x))dx\text{Trapped Water} = \int_a^b (h(x) - f(x)) dx

where **h(x)** is the water level and **f(x)** is the elevation.

Give me a specific function, and I'll calculate the trapped water for you! 🚀

d) Count Ways to Express a Number as the Sum of Consecutive Natural Numbers
Given a natural number n, we are asked to find the total number of ways to express n as the sum of consecutive natural numbers.

Example 1:Input: 15
Output: 3
Explanation: There are 3 ways to represent 15 as sum of consecutive natural numbers as follows:
1 + 2 + 3 + 4 + 5 = 15
4 + 5 + 6 = 15
7 + 8 = 15

ANSWER -

To count ways to express nn as the sum of consecutive numbers:

1. Iterate over possible sequence lengths mm.
2. Check if k=n−m(m−1)2mk = \frac{n - \frac{m(m-1)}{2}}{m} is a positive integer.
3. Count valid sequences.

Time Complexity: O(N)O(\sqrt{N}).

Python Code:

```
def count_consecutive_sum_ways(n):
    count, m = 0, 1
    while (m * (m - 1)) // 2 < n:
        if (n - (m * (m - 1)) // 2) % m == 0:
            count += 1
        m += 1
    return count

print(count_consecutive_sum_ways(15))
 # Output: 3
```

e) Write a piece of code to find the largest 5 digit prime number in the first 100 digits of Pi?

ANSWER -

Steps to find the largest 5-digit prime in the first 100 digits of Pi:

1. Extract **first 100 digits** of **Pi** (excluding "3.").
2. Slide a **5-digit window** across these digits.
3. Check if the number is **prime**.
4. Track the **largest prime found**.

Python Code:

```
from sympy import isprime

pi_digits = (

"14159265358979323846264338327950288419716939937510582097494459230781640628 6"

"20899862803482534211706798214808651328230664709384460955058223172535940812 8"
)

def find_largest_prime():
    return max(int(pi_digits[i:i+5]) for i in range(95) if isprime(int(pi_digits[i:i+5])))
```

```
print(find_largest_prime())  # Output: Largest 5-digit prime
```

🚀 **Time Complexity:** O(95)O(95) → **Fast & Efficient!**

f) Write a piece of code to determine if two rectangles are intersecting each other or not? Think about the logic and explain the same before showing the code.

ANSWER -

Logic:

Two rectangles **do not intersect** if:

1. One is **left** of the other → x2<x3x2 < x3 or x4<x1x4 < x1.
2. One is **above** the other → y2<y3y2 < y3 or y4<y1y4 < y1.
   If neither condition is met, they **intersect**.

Python Code:

```
def is_intersecting(x1, y1, x2, y2, x3, y3, x4, y4):
    return not (x2 < x3 or x4 < x1 or y2 < y3 or y4 < y1)


# Example
print(is_intersecting(0, 0, 4, 4, 2, 2, 6, 6))  # True
```

🚀 **Time Complexity:** O(1)O(1) → **Fast & Efficient!**

g) Explain a piece of code that you wrote which you are proud of? If you have not written any code, please write your favorite subject in engineering studies. We can go deep into that subject.

ANSWER -

Proud Project: Real-Time Collaborative Whiteboard

✅ **Built with:** React, Fabric.js, Socket.io
✅ **Key Features:**

- **Real-time updates** via WebSockets
- **Smooth drawing** with Fabric.js
- **Undo/Redo functionality**
- **Secure authentication** (Keycloak + Docker)
- **Export as Image/PDF**

✅ **Challenges Solved:**
✔ **Optimized WebSockets** for low latency
✔ **Handled multiple users efficiently**
✔ **Scalable & interactive UI**

h) Write a Program to multiple two (n*n) Matrices? Explain logic on how you would do it.

ANSWER -

Matrix Multiplication Logic

For two n×nn \times n matrices **A** and **B**, the result **C** is computed as:

C[i][j]=∑k=0n−1A[i][k]×B[k][j]C[i][j] = \sum_{k=0}^{n-1} A[i][k] \times B[k][j]

Python Code:

```
def multiply_matrices(A, B, n):
    C = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                C[i][j] += A[i][k] * B[k][j]
    return C

A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]
result = multiply_matrices(A, B, 2)

for row in result:
    print(row)
```

✅ **Time Complexity:** O(n3)O(n^3)

I declare that I have done the above work by myself and not worked with anyone or got help from any individual on the internet.