Interview questions:

Notes-
a) This is an open book test and duration of this test is 48 hours.
b) You can use the internet to search for answers and build the code.
c) Please be honest and tell us what you searched and what you remembered.
d) Do not copy paste the code you found off the internet, try to build it and explain in the space given.
e) Do not work in groups. It is an individual activity.
f) If you create a Github profile with the answers to questions – its an added bonus.
g) You can write your code/comments in the word document, convert to PDF and send it across to us.
h) We take cheating very seriously and could have negative repercussions if found out later.
i) Do not use LLM as it is (Ok for learning concepts) and show screenshots of testing results after running through a online compilation tool. .

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.

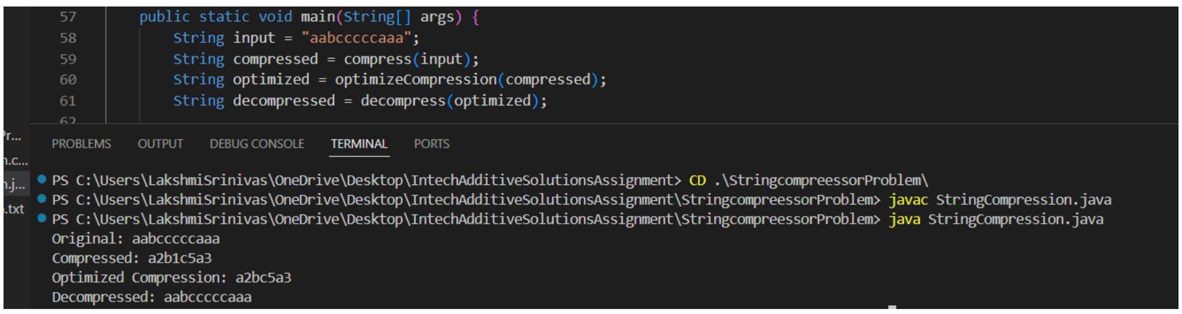
a) String compression
Implement a method to perform string compression. E.g. 'aabcccccaaa' should be a2b1c5a3. The code to implement this is given in the link - https://www.educative.io/answers/string-compression-using-run-length-encoding

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.

CODE:
Programming Language:Java

```java
package StringcompreessorProblem;

public class StringCompression {

    public static String compress(String s) {
        StringBuilder compressed = new StringBuilder();
        int count = 1;

        for (int i = 1; i <= s.length(); i++) {
            if (i == s.length() || s.charAt(i) != s.charAt(i - 1)) {
                compressed.append(s.charAt(i - 1)).append(count);
                count = 1;
            } else {
                count++;
            }
        }
        return compressed.toString();
    }

    public static String optimizeCompression(String s) {
        StringBuilder optimized = new StringBuilder();

        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            int count = 0;

            while (i + 1 < s.length() && Character.isDigit(s.charAt(i
+ 1))) {
                count = count * 10 + (s.charAt(++i) - '0');
            }

            if (count == 1) {
                optimized.append(ch);
            } else {
                optimized.append(ch).append(count);
            }
        }
        return optimized.toString();
    }

    public static String decompress(String s) {
        StringBuilder decompressed = new StringBuilder();

        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            int count = 0;

            while (i + 1 < s.length() && Character.isDigit(s.charAt(i
+ 1))) {
                count = count * 10 + (s.charAt(++i) - '0');
            }
            decompressed.append(String.valueOf(ch).repeat(Math.max(1,
count)));
        }
        return decompressed.toString();
    }

    // Test Cases
    public static void main(String[] args) {
        String input = "aabcccccaaa";
        String compressed = compress(input);
        String optimized = optimizeCompression(compressed);
        String decompressed = decompress(optimized);

        System.out.println("Original: " + input);
        System.out.println("Compressed: " + compressed);
        System.out.println("Optimized Compression: " + optimized);
        System.out.println("Decompressed: " + decompressed);


    }
}
```

OUTPUT:



```
    public static void main(String[] args) {
        String input = "aabcccccaaa";
        String compressed = compress(input);
        String optimized = optimizeCompression(compressed);
        String decompressed = decompress(optimized);
```

```
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment> CD .\StringcompreessorProblem\
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\StringcompreessorProblem> javac StringCompression.java
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\StringcompreessorProblem> java StringCompression.java
Original: aabcccccaaa
Compressed: a2b1c5a3
Optimized Compression: a2bc5a3
Decompressed: aabcccccaaa
```

Explanation:

The program consists of three main functions:
----> First Compressor: compress(String s)
logic for this is : It counts consecutive occurrences of characters and replaces them with char + count.("aabcccccaaa" → "a2b1c5a3)
steps to follow:

Traverse the string.
If the next character is different, append char + count to the result.
Reset the count for the next character.

----->SecondCompressor: optimizeCompression(String s)
logic:This function removes 1s, making the output smaller.("a2b1c5a3" → "ab2c5a3" (removes "b1" → "b"))
steps to follow :
Traverse the compressed string.
Extract numeric values and ignore 1s.
Reconstruct the optimized string.

-----> Decompressor: decompress(String s)
logic:Takes a compressed string and reconstructs the original string.("ab2c5a3" → "aabbcaaaaa");
steps to follow:
Traverse the string.
Extract numbers after characters and repeat that character accordingly.
Append the expanded result.

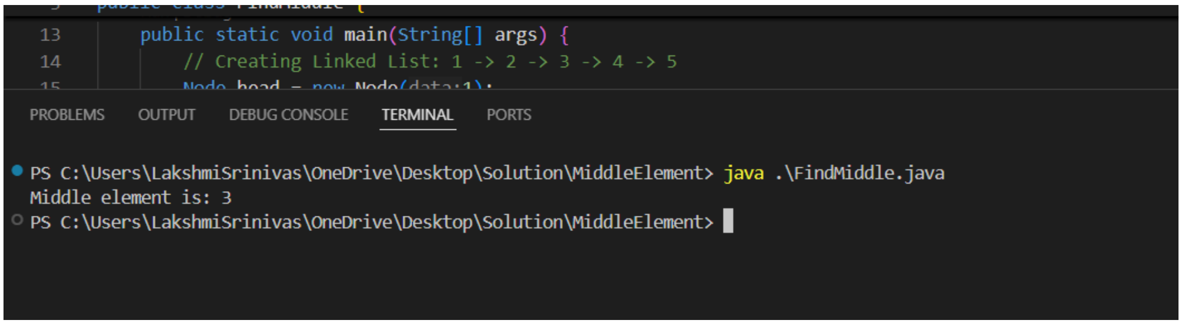
This methods ensures that the compression is efficient and reversible.

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

Code:
Programming Language:Java.

```java
package MiddleElement;

public class FindMiddle {
    static class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            this.next = null;
        }
    }

    public static void main(String[] args) {
        // Creating Linked List: 1 -> 2 -> 3 -> 4 -> 5
        Node head = new Node(1);
        head.next = new Node(2);
        head.next.next = new Node(3);
        head.next.next.next = new Node(4);
        head.next.next.next.next = new Node(5);

        // Find Middle
        Node slow = head;
        Node fast = head;

        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }

        System.out.println("Middle element is: " + slow.data);
    }
}
```

Output:



```java
    public static void main(String[] args) {
        // Creating Linked List: 1 -> 2 -> 3 -> 4 -> 5
```

```
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\Solution\MiddleElement> java .\FindMiddle.java
Middle element is: 3
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\Solution\MiddleElement>
```

Explanation:

First of all we creating the class called findMiddle .And inside the class name we have a

node class in that class we store the data with the help data variable and the next variable is useful to store the reference to the next node in the list.
And next in the main method we create a linkedlist like 1 -> 2 -> 3 -> 4-> 5
We mark the head node with value 1 and we linking the next nodes one by one
At last we initialize the pointers as slow,fast(slow indicates the head where as the fast also starts at head).
We use the while loop and this loop continues until the fast reaches the end of the list.
The slow pointer moves 1 step at a time,where the fast pointer moves 2 steps at a time.
Since fast is moving twice as fast as slow, by the time fast reaches the end ,slow will be at the middle of the list.
After the loop ends, slow will point to the middle element. We print the middle element

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/

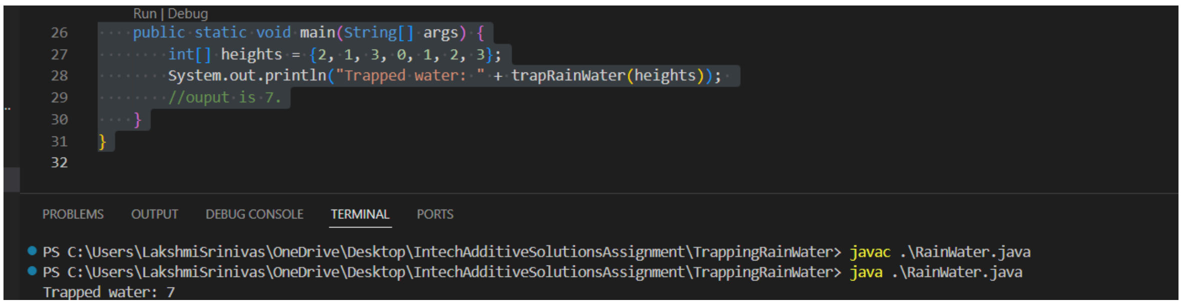
Go through the above code for the solution.

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.

Code:
Programming Language:JAVA

```java
public class RainWater {
    public static int trapRainWater(int[] heights) {
        if (heights == null || heights.length < 3) {
            return 0;
        }

        int left = 0, right = heights.length - 1;
        int leftMax = heights[left], rightMax = heights[right];
        int trappedWater = 0;

        while (left < right) {
            if (leftMax < rightMax) {
                left++;
                leftMax = Math.max(leftMax, heights[left]);
                trappedWater += Math.max(0, leftMax - heights[left]);
            } else {
                right--;
                rightMax = Math.max(rightMax, heights[right]);
                trappedWater += Math.max(0, rightMax -
heights[right]);
            }
        }

        return trappedWater;
    }

    public static void main(String[] args) {
        int[] heights = {2, 1, 3, 0, 1, 2, 3};
        System.out.println("Trapped water: " +
trapRainWater(heights));
        //ouput is 7.
    }
}
```

OUTPUT:



```
    public static void main(String[] args) {
        int[] heights = {2, 1, 3, 0, 1, 2, 3};
        System.out.println("Trapped water: " + trapRainWater(heights));
        //ouput is 7.
    }
}
```

```
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\TrappingRainWater> javac .\RainWater.java
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\TrappingRainWater> java .\RainWater.java
Trapped water: 7
```

Explanation:

how i solve this problem:

1.Use two pointers (left and right) starting at both ends.

--> Keep track of maximum heights on both sides (leftMax and rightMax).

--> Move the pointer with the smaller height:

-->If leftMax is smaller, move left forward and calculate trapped water at left.

-->If rightMax is smaller, move right backward and calculate trapped water at right.

-->Keep adding trapped water until the pointers meet.

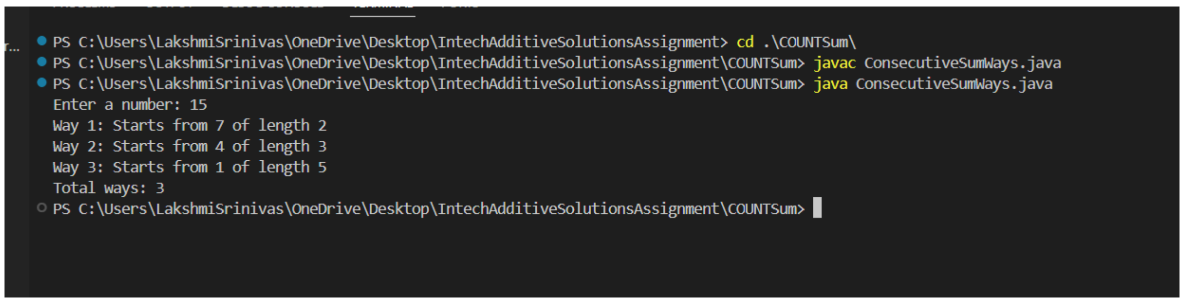
d) Count Ways to Express a Number as the Sum of Consecutive Natural Numbers
Given a natural number n, we are asked to find the total number of ways to express n as the sum of consecutive natural numbers.

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

Code:
Programming Language:JAVA

```
package COUNTSum;

import java.util.Scanner;

public class ConsecutiveSumWays {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = sc.nextInt();

        int c = 0;

        // this is the formula
        for (int k = 2; k * (k - 1) / 2 < n; k++) {
            int num= n - k * (k - 1) / 2;
            if (num % k == 0) {
                int l = num / k;
                if (l> 0) {
                    c++;
                    System.out.println("Way " + c + ": Starts from "
+ l + " of length " + k);
                }
            }
        }

        System.out.println("Total ways: " + c);
        sc.close();
    }
}
```

OUTPUT:



```
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment> cd .\COUNTSum\
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\COUNTSum> javac ConsecutiveSumWays.java
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\COUNTSum> java ConsecutiveSumWays.java
Enter a number: 15
Way 1: Starts from 7 of length 2
Way 2: Starts from 4 of length 3
Way 3: Starts from 1 of length 5
Total ways: 3
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\COUNTSum>
```

Explanation:

The logic is based on a simple formula

n = k * x + k * (k - 1) / 2

We want to find valid values of number of terms and starting number which satisfy this equation.
To do this, for every possible starting from 2, we calculate:

x = (n - k*(k-1)/2) / k

If x is a positive integer, then it is a valid sequence.

The loop continues until k*(k-1)/2 becomes greater than or equal to n because after that, no solution is possible.

For example, for n = 15, the valid sequences are: 1 + 2 + 3 + 4 + 5 = 15
4 + 5 + 6 = 15
7 + 8 = 15
Total ways = 3

In short, the approach is to iterate possible lengths of the sequence (k), compute the starting number x, and check if the sequence is valid.

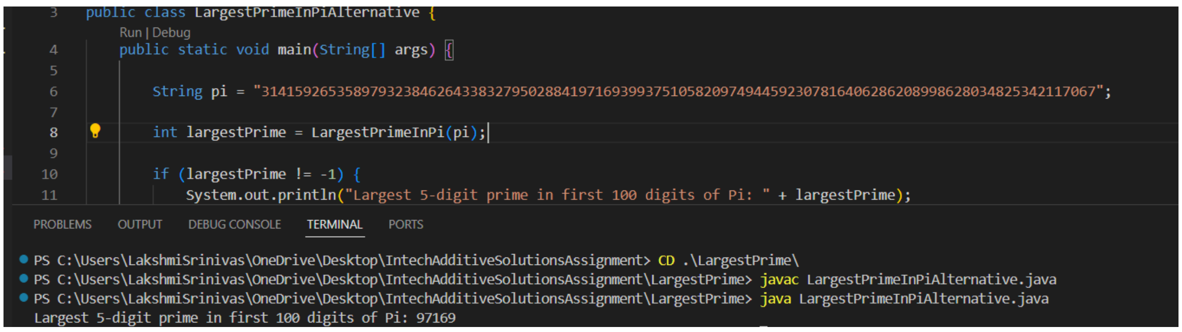
e) Write a piece of code to find the largest 5 digit prime number in the first 100 digits of Pi?

Code:
Programming Language:JAVA

```
package LargestPrime;

public class LargestPrimeInPiAlternative {
    public static void main(String[] args) {

        String pi =
"31415926535897932384626433832795028841971693993751058209749445923078
16406286208998628034825342117067";

        int largestPrime = LargestPrimeInPi(pi);

        if (largestPrime != -1) {
            System.out.println("Largest 5-digit prime in first 100
digits of Pi: " + largestPrime);
        } else {
            System.out.println("No 5-digit prime found.");
        }
    }

    // Main logic function
    public static int LargestPrimeInPi(String pi) {
        int maxPri = -1;

        for (int i = 0; i <= pi.length() - 5; i++) {
            int num = Integer.parseInt(pi.substring(i, i + 5));
            if (isPrime(num) && num > maxPri) {
                maxPri = num;
```

```java
                }
            }
            return maxPri;
        }

        // Prime check function
        public static boolean isPrime(int k) {
            if (k < 2) return false;
            if (k == 2 || k == 3) return true;
            if (k % 2 == 0 || k% 3 == 0) return false;
            for (int i = 5; i * i <= k; i += 6) {
                if (k % i == 0 || k % (i + 2) == 0) return false;
            }
            return true;
        }
}
```

Output:



```
public class LargestPrimeInPiAlternative {
    Run | Debug
    public static void main(String[] args) {

        String pi = "3141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067";

        int largestPrime = LargestPrimeInPi(pi);

        if (largestPrime != -1) {
            System.out.println("Largest 5-digit prime in first 100 digits of Pi: " + largestPrime);
```

```
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment> CD .\LargestPrime\
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\LargestPrime> javac LargestPrimeInPiAlternative.java
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\LargestPrime> java LargestPrimeInPiAlternative.java
Largest 5-digit prime in first 100 digits of Pi: 97169
```

Explanation:

We are trying to find the largest 5-digit prime number present in the first 100 digits of Pi.
Pi digits are stored as a String because:
If we store them as an int or double, Java cannot store so many digits correctly.
We want to extract 5 digits at a time substring() works easily on String.

We use a for loop to go from i = 0 to i = 95.
Why till pi.length() - 5? Because we want to take 5 digits at a time,so last index should be 95 (as 95 + 5 = 100).

Inside the loop: pi.substring(i, i + 5) → takes 5 digits starting from index i,then we convert

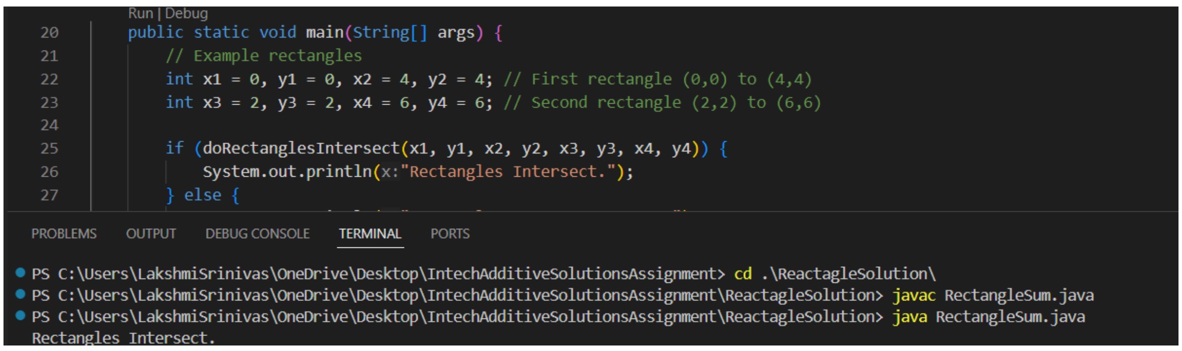
that string into an integer using Integer.parseInt().
And then we check the num variable is prime or not if it is prime return it
If the 5-digit number is prime and larger than the previous largest , we update maxPi.
After checking all possible 5-digit numbers , we print the largest prime found.

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.

Code:
Programming Language:JAVA

```
package ReactagleSolution;

public class RectangleSum {
    // Function to check if two rectangles intersect
    public static boolean doRectanglesIntersect(int x1, int y1, int
x2, int y2,
                                                int x3, int y3, int
x4, int y4) {
        // Check if one rectangle is completely to the left or right
of the other
        if (x2 < x3 || x4 < x1) {
            return false;
        }

        // Check if one rectangle is completely above or below the
other
        if (y2 < y3 || y4 < y1) {
            return false;
        }

        return true; // Rectangles overlap
    }

    public static void main(String[] args) {
        // Example rectangles
        int x1 = 0, y1 = 0, x2 = 4, y2 = 4; // First rectangle (0,0)
to (4,4)
        int x3 = 2, y3 = 2, x4 = 6, y4 = 6; // Second rectangle (2,2)
to (6,6)

        if (doRectanglesIntersect(x1, y1, x2, y2, x3, y3, x4, y4)) {
            System.out.println("Rectangles Intersect.");
        } else {
            System.out.println("Rectangles Do Not Intersect.");
        }
    }
}
```

Output:



```
Run | Debug
public static void main(String[] args) {
    // Example rectangles
    int x1 = 0, y1 = 0, x2 = 4, y2 = 4; // First rectangle (0,0) to (4,4)
    int x3 = 2, y3 = 2, x4 = 6, y4 = 6; // Second rectangle (2,2) to (6,6)

    if (doRectanglesIntersect(x1, y1, x2, y2, x3, y3, x4, y4)) {
        System.out.println(x:"Rectangles Intersect.");
    } else {
```

```
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment> cd .\ReactagleSolution\
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\ReactagleSolution> javac RectangleSum.java
PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\ReactagleSolution> java RectangleSum.java
Rectangles Intersect.
```

Explanation:

The function checks if one rectangle is completely to the left/right or completely above/below the other.
If either condition is true, the rectangles do not overlap (return false).
Otherwise, they must be overlapping, so we return true.
The main method tests this logic with sample rectangles and prints whether they intersect or not.

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.

CODE:

One piece of code I am proud of is the Spotify Clone project I built using HTML, CSS, JavaScript, and Bootstrap.

The project replicates the core UI and basic functionality of the Spotify music player.
Users can:
See a list of songs with album art.
Play/Pause songs.
Navigate to next/previous songs.
Seek through the progress bar.

What I implemented:

In this project, I used the following:

HTML & Bootstrap to design a clean and responsive layout.

CSS for custom styling and making it visually close to Spotify.

JavaScript's Audio API to handle:

Play/Pause toggle.

Next/Previous track functionality.

Dynamic updating of progress bar and current song info.

Event listeners to capture user actions.

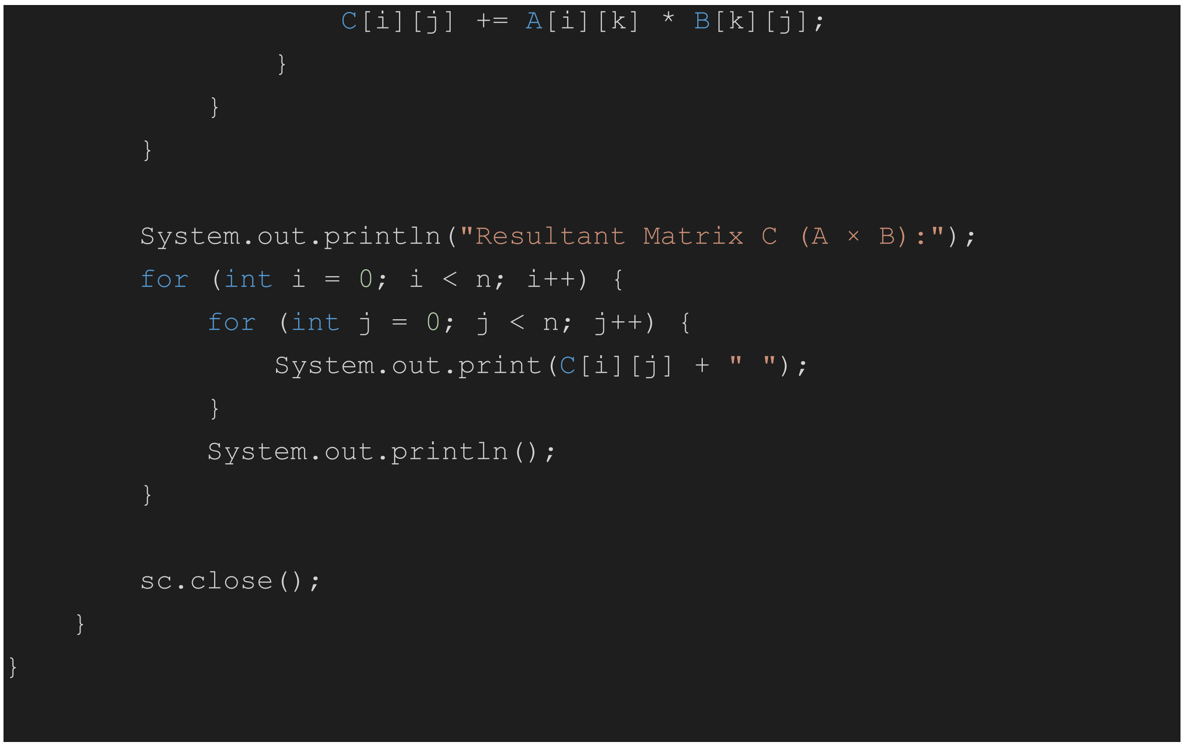
Why am I proud of this project?
Because this project helped me understand:
DOM Manipulation in-depth.
How to work with real-time audio events.
Handling state changes like play, pause, and song switching.
Making the UI responsive and interactive using JavaScript and Bootstrap.

It was the first time I connected UI + Logic + Media, and seeing it all come together gave me confidence to build real-world projects.

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

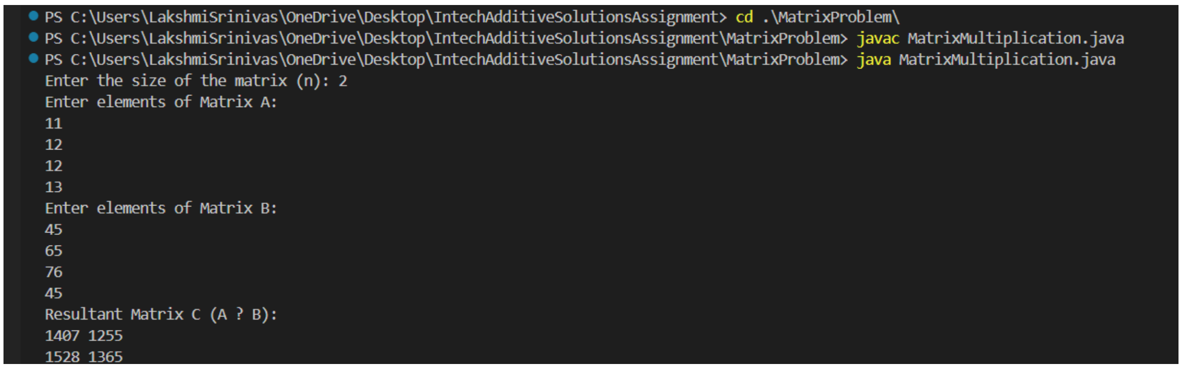
Code:
Programming Language:JAVA

```
package MatrixProblem;
import java.util.Scanner;

public class MatrixMultiplication {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        System.out.print("Enter the size of the matrix (n): ");
        int n = sc.nextInt();


        int[][] A = new int[n][n];
        int[][] B = new int[n][n];
        int[][] C = new int[n][n];

        System.out.println("Enter elements of Matrix A:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                A[i][j] = sc.nextInt();
            }
        }


        System.out.println("Enter elements of Matrix B:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                B[i][j] = sc.nextInt();
            }
        }


        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                for (int k = 0; k < n; k++) {
```



```java
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }

        System.out.println("Resultant Matrix C (A × B):");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(C[i][j] + " ");
            }
            System.out.println();
        }

        sc.close();
    }
}
```

OUTPUT:



```
● PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment> cd .\MatrixProblem\
● PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\MatrixProblem> javac MatrixMultiplication.java
● PS C:\Users\LakshmiSrinivas\OneDrive\Desktop\IntechAdditiveSolutionsAssignment\MatrixProblem> java MatrixMultiplication.java
  Enter the size of the matrix (n): 2
  Enter elements of Matrix A:
  11
  12
  12
  13
  Enter elements of Matrix B:
  45
  65
  76
  45
  Resultant Matrix C (A ? B):
  1407 1255
  1528 1365
```

Explanation:

If you have two matrices:

Matrix A of size n × n

Matrix B of size n × n

The product matrix C = A × B will also be of size n × n.

STEPS:

Take input of Matrix A and Matrix B.

Initialize a result matrix C of size n × n with all zeros.

Apply 3 nested loops:

Loop for row i of Matrix A.

Loop for column j of Matrix B.

Loop for k to perform multiplication and summation.

Print the resultant Matrix C.

In short:

Outer Loop: Go through each row i of Matrix A.

Middle Loop: Go through each column j of Matrix B.

Inner Loop: Multiply elements of row i in A with column j in B and sum them up.

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.