Experiment 10

Student Name: Arfan Mohd **UID: 22BCS12329**
Branch: CSE **Section/Group: NTPP 603B**
Semester: 6 **Date of Performance: 18/04/25**
Subject Name: AP Lab 2 **Subject Code:22CSP-351**

1. **Aim**:
   A. Pascal's Triangle
   B. Task Scheduler
   C. Divide Two Integers
   D. Trapping rainwater

2. **Code:**

A.
```
class Solution {
    public List<List<Integer>> generate(int numRows) {
         List<List<Integer>> result = new ArrayList<>();

        for (int i = 0; i < numRows; i++) {
            List<Integer> row = new ArrayList<>();
            row.add(1);

            for (int j = 1; j < i; j++) {
                    row.add(result.get(i - 1).get(j - 1) +
 result.get(i - 1).get(j));
            }

            if (i > 0) {
                row.add(1);
            }

            result.add(row);
        }
```

```
        return result;
    }
}
```

B.

```
class Solution {
    public int leastInterval(char[] tasks, int n) {
         int[] taskCounts = new int[26];
        for (char task : tasks) {
            taskCounts[task - 'A']++;
        }

        Arrays.sort(taskCounts);
        int maxCount = taskCounts[25];
        int maxCountTasks = 1;

        for (int i = 24; i >= 0; i--) {
            if (taskCounts[i] == maxCount) {
                maxCountTasks++;
            } else {
                break;
            }
        }

        int result = Math.max(tasks.length, (maxCount - 1) *
        (n + 1) + maxCountTasks);
        return result;
    }
}
```

C.

```
class Solution {
    public int divide(int dividend, int divisor) {
            if (dividend == Integer.MIN_VALUE && divisor ==
    -1) return Integer.MAX_VALUE;

        long dvd = Math.abs((long) dividend);
        long dvs = Math.abs((long) divisor);
        int result = 0;
```

```
        while (dvd >= dvs) {
            long temp = dvs, multiple = 1;
            while (dvd >= (temp << 1)) {
                temp <<= 1;
                multiple <<= 1;
            }
            dvd -= temp;
            result += multiple;
        }

        return (dividend < 0) == (divisor < 0) ? result : -
result;
        }
    }
```

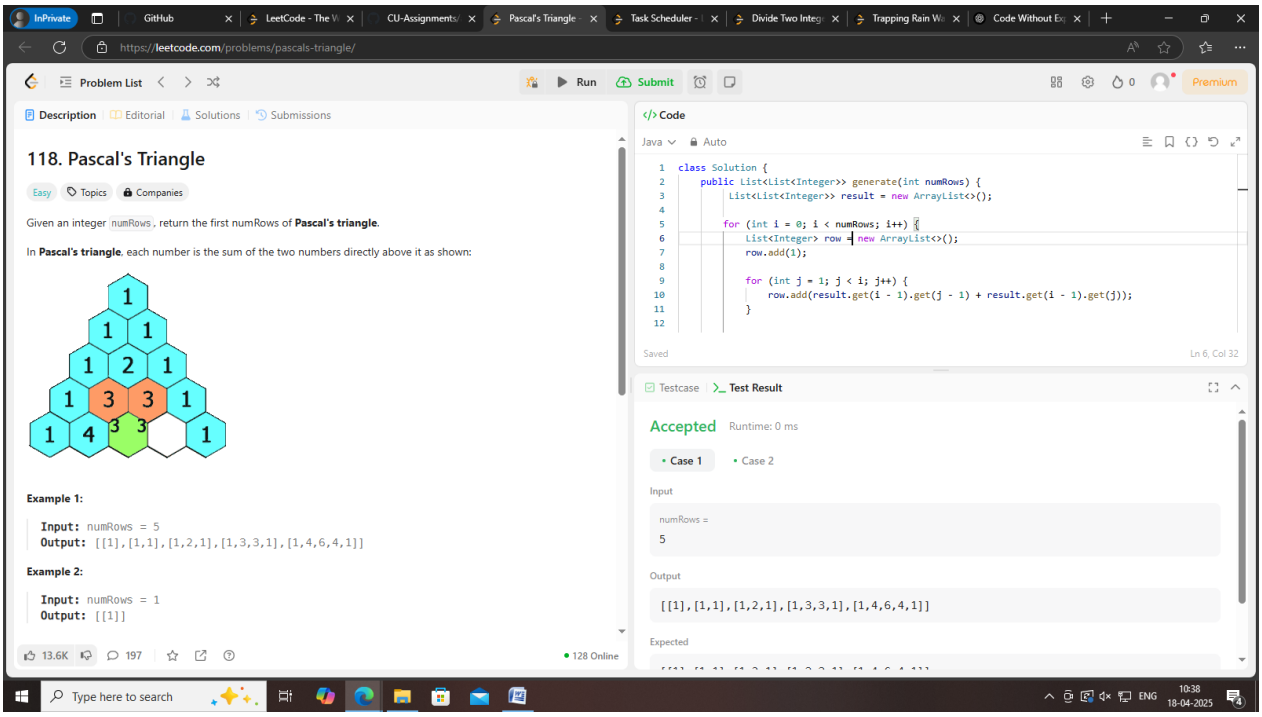
D.

```
class Solution {
    public int trap(int[] height) {
         if (height == null || height.length == 0) return 0;

        int left = 0, right = height.length - 1;
        int leftMax = 0, rightMax = 0;
        int waterTrapped = 0;

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

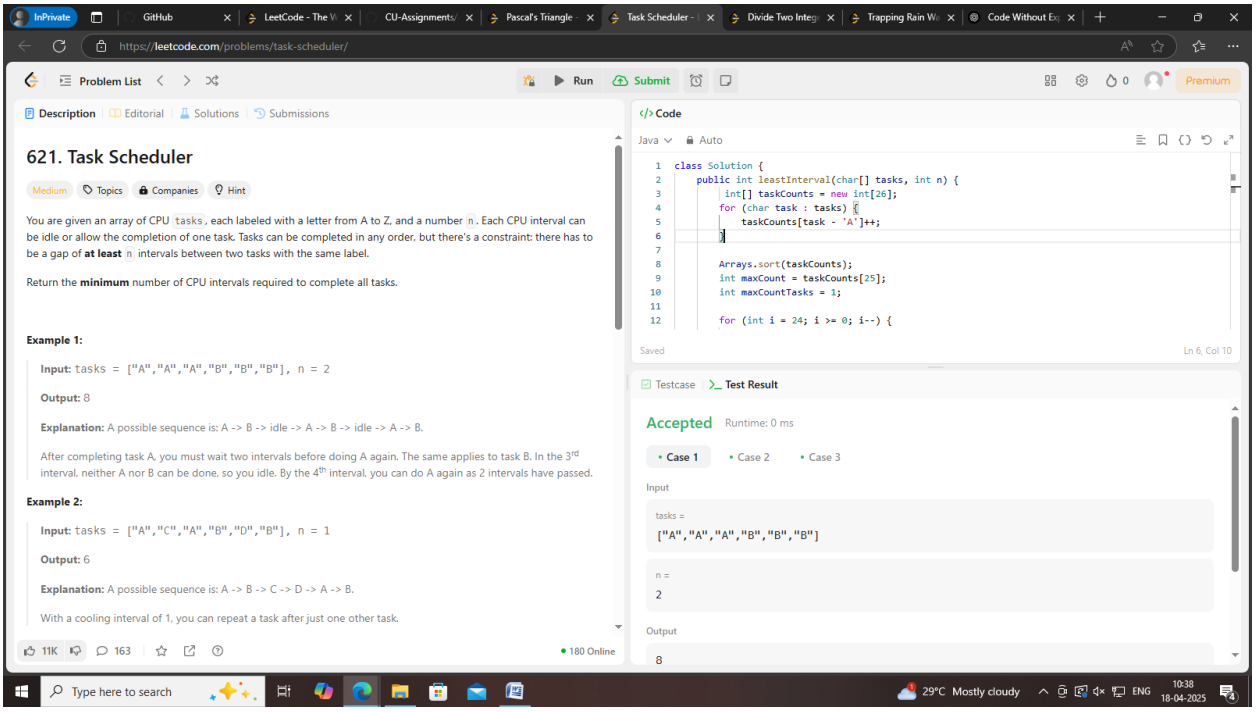
```
                }
                right--;
            }
        }

        return waterTrapped;
    }
}
```

3. Output:

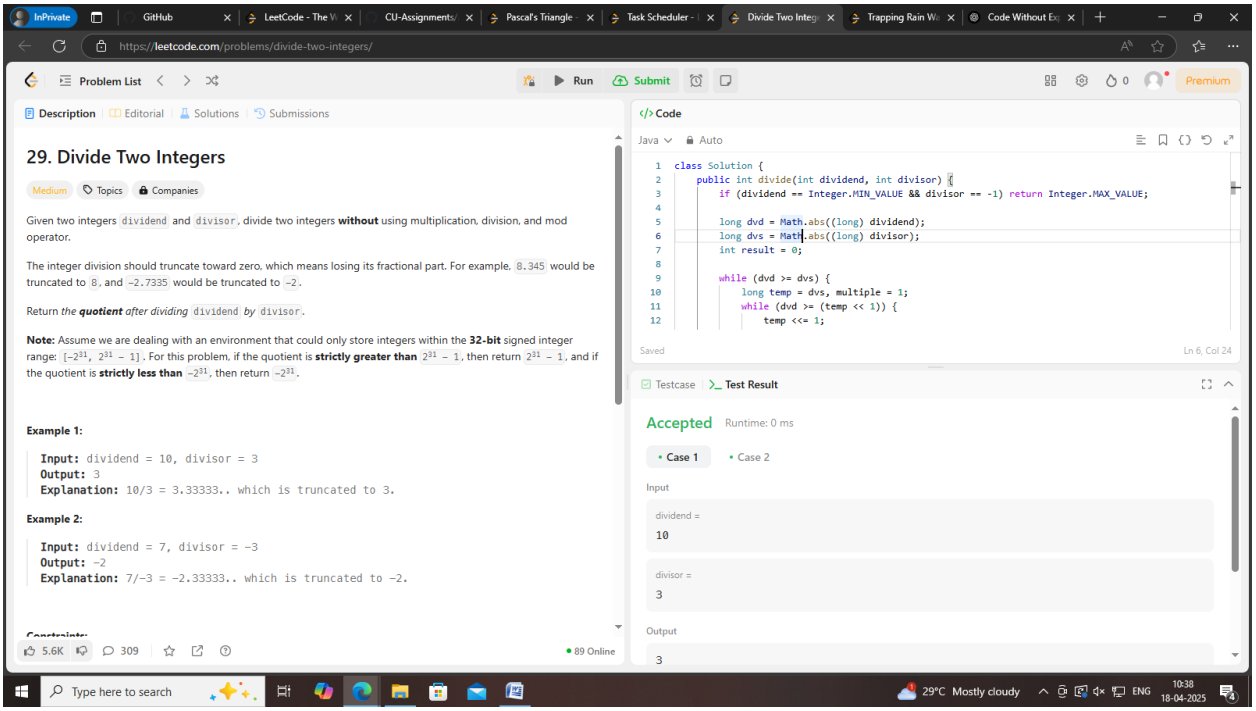
A.



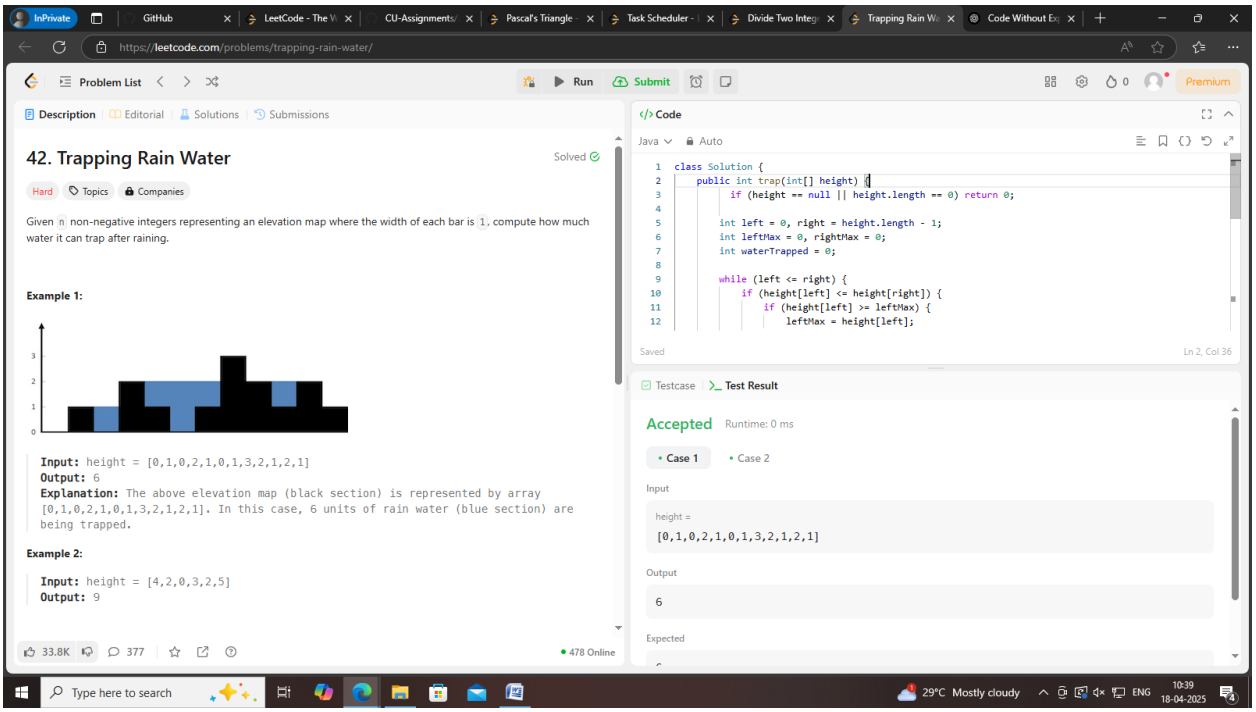
B.


https://leetcode.com/problems/task-scheduler/
Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
621. Task Scheduler
Medium
Topics
Companies
Hint
You are given an array of CPU tasks, each labeled with a letter from A to Z, and a number n. Each CPU interval can be idle or allow the completion of one task. Tasks can be completed in any order, but there's a constraint: there has to be a gap of at least n intervals between two tasks with the same label.
Return the minimum number of CPU intervals required to complete all tasks.
Example 1:
Input: tasks = ["A","A","A","B","B","B"], n = 2
Output: 8
Explanation: A possible sequence is: A -> B -> idle -> A -> B -> idle -> A -> B.
After completing task A, you must wait two intervals before doing A again. The same applies to task B. In the 3rd interval, neither A nor B can be done, so you idle. By the 4th interval, you can do A again as 2 intervals have passed.
Example 2:
Input: tasks = ["A","C","A","B","D","B"], n = 1
Output: 6
Explanation: A possible sequence is: A -> B -> C -> D -> A -> B.
With a cooling interval of 1, you can repeat a task after just one other task.
11K
163
180 Online
Code
Java
Auto
class Solution {
public int leastInterval(char[] tasks, int n) {
int[] taskCounts = new int[26];
for (char task : tasks) {
taskCounts[task - 'A']++;
}
Arrays.sort(taskCounts);
int maxCount = taskCounts[25];
int maxCountTasks = 1;
for (int i = 24; i >= 0; i--) {
Saved
Ln 6, Col 10
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
tasks =
["A","A","A","B","B","B"]
n =
2
Output
8
Type here to search
29°C Mostly cloudy
ENG
10:38
18-04-2025

C.


https://leetcode.com/problems/divide-two-integers/
Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
29. Divide Two Integers
Medium
Topics
Companies
Given two integers dividend and divisor, divide two integers without using multiplication, division, and mod operator.
The integer division should truncate toward zero, which means losing its fractional part. For example, 8.345 would be truncated to 8, and -2.7335 would be truncated to -2.
Return the quotient after dividing dividend by divisor.
Note: Assume we are dealing with an environment that could only store integers within the 32-bit signed integer range: [-2^31, 2^31 - 1]. For this problem, if the quotient is strictly greater than 2^31 - 1, then return 2^31 - 1, and if the quotient is strictly less than -2^31, then return -2^31.
Example 1:
Input: dividend = 10, divisor = 3
Output: 3
Explanation: 10/3 = 3.33333.. which is truncated to 3.
Example 2:
Input: dividend = 7, divisor = -3
Output: -2
Explanation: 7/-3 = -2.33333.. which is truncated to -2.
5.6K
309
89 Online
Code
Java
Auto
class Solution {
public int divide(int dividend, int divisor) {
if (dividend == Integer.MIN_VALUE && divisor == -1) return Integer.MAX_VALUE;
long dvd = Math.abs((long) dividend);
long dvs = Math.abs((long) divisor);
int result = 0;
while (dvd >= dvs) {
long temp = dvs, multiple = 1;
while (dvd >= (temp << 1)) {
temp <<= 1;
Saved
Ln 6, Col 24
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
dividend =
10
divisor =
3
Output
3
Type here to search
29°C Mostly cloudy
ENG
10:38
18-04-2025

D.


https://leetcode.com/problems/trapping-rain-water/
Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
42. Trapping Rain Water
Solved
Hard
Topics
Companies
Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.
Example 1:
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.
Example 2:
Input: height = [4,2,0,3,2,5]
Output: 9
33.8K
377
478 Online
Code
Java
Auto
class Solution {
public int trap(int[] height) {
if (height == null || height.length == 0) return 0;
int left = 0, right = height.length - 1;
int leftMax = 0, rightMax = 0;
int waterTrapped = 0;
while (left <= right) {
if (height[left] <= height[right]) {
if (height[left] >= leftMax) {
leftMax = height[left];
Saved
Ln 2, Col 36
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
height =
[0,1,0,2,1,0,1,3,2,1,2,1]
Output
6
Expected
Type here to search
29°C Mostly cloudy
ENG
10:39
18-04-2025