



## Experiment 10

**Student Name: Riyan Reyaz**

**UID:22BCS11986**

**Branch: CSE**

**Section/Group: NTPP 603/B**

**Semester: 06**

**Date of Performance: 18/04/2025**

**Subject Name: AP Lab II**

**Subject Code: 22CSP-351**

### 1. Aim:

- a) Pascal's Triangle
- b) Hamming Distance
- c) Task Scheduler

### 2. Source Code:

**a.**

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        vector<vector<int>> ans;

        for (int i = 0; i < numRows; ++i)
            ans.push_back(vector<int>(i + 1, 1));

        for (int i = 2; i < numRows; ++i)
            for (int j = 1; j < ans[i].size() - 1; ++j)
                ans[i][j] = ans[i - 1][j - 1] + ans[i - 1][j];

        return ans;
    }
};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

**b.**

```
class Solution {
public:
    int hammingDistance(int x, int y) {
        int ans = 0;

        while (x > 0 || y > 0) {
            ans += (x & 1) ^ (y & 1);
            x >>= 1;
            y >>= 1;
        }

        return ans;
    }
};
```

**c.**

```
class Solution {
public:
    int leastInterval(vector<char>& tasks, int n) {
        if (n == 0)
            return tasks.size();

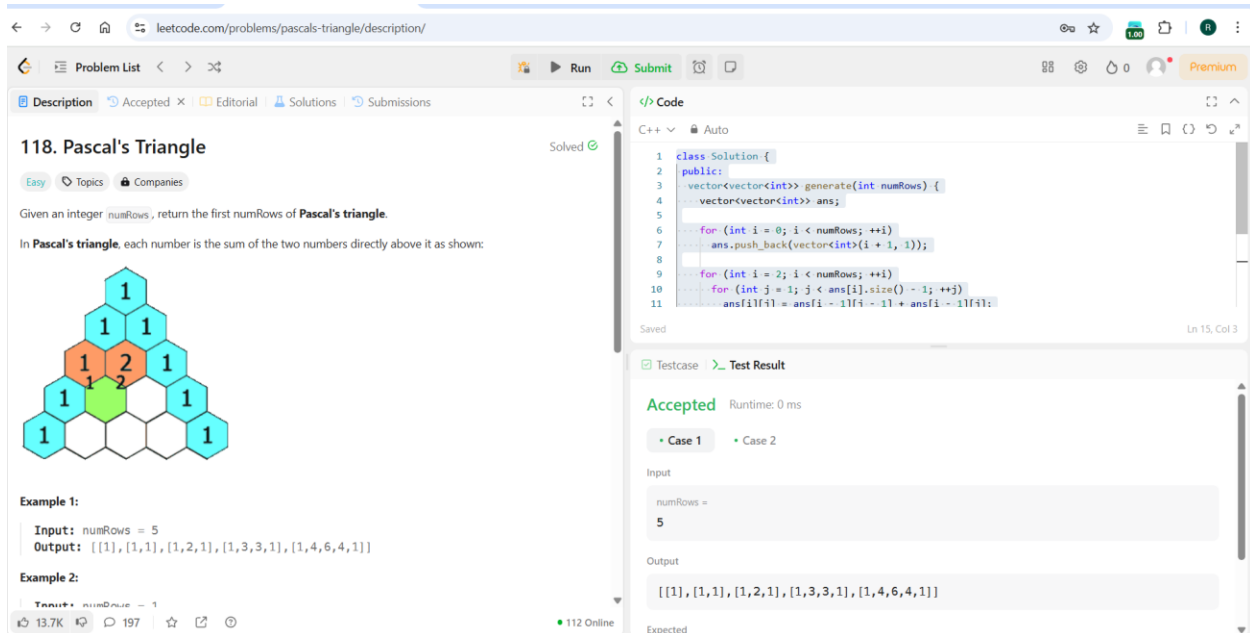
        vector<int> count(26);

        for (const char task : tasks)
            ++count[task - 'A'];

        const int maxFreq = ranges::max(count);
        const int maxFreqTaskOccupy = (maxFreq - 1) * (n + 1);
        const int nMaxFreq = ranges::count(count, maxFreq);
        return max(maxFreqTaskOccupy + nMaxFreq, static_cast<int>(tasks.size()));
    }
};
```

## Screenshot of Outputs:

a.



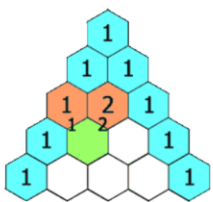
The screenshot shows the LeetCode interface for problem 118, "Pascal's Triangle". The problem description states: "Given an integer `numRows`, return the first `numRows` of Pascal's triangle. In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:". A diagram of Pascal's triangle is provided, showing the first 5 rows. The first row has one element (1). The second row has two elements (1, 1). The third row has three elements (1, 2, 1). The fourth row has four elements (1, 3, 3, 1). The fifth row has five elements (1, 4, 6, 4, 1). The example input is `numRows = 5`, and the output is `[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]`. The code editor shows a C++ solution that generates the triangle using a vector of vectors. The test result is "Accepted" with a runtime of 0 ms.

**118. Pascal's Triangle**

Easy Topics Companies

Given an integer `numRows`, return the first `numRows` of Pascal's triangle.

In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:



**Example 1:**

Input: `numRows = 5`  
Output: `[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]`

**Example 2:**

Input: `numRows = 1`  
Output: `[[1]]`

13.7K 197 112 Online

```

1 class Solution {
2 public:
3     vector<vector<int>> generate(int numRows) {
4         vector<vector<int>> ans;
5
6         for (int i = 0; i < numRows; ++i)
7             ans.push_back(vector<int>(i + 1, 1));
8
9         for (int i = 2; i < numRows; ++i)
10            for (int j = 1; j < ans[i].size() - 1; ++j)
11                ans[i][j] = ans[i - 1][j - 1] + ans[i - 1][j];
12    }
13 }

```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

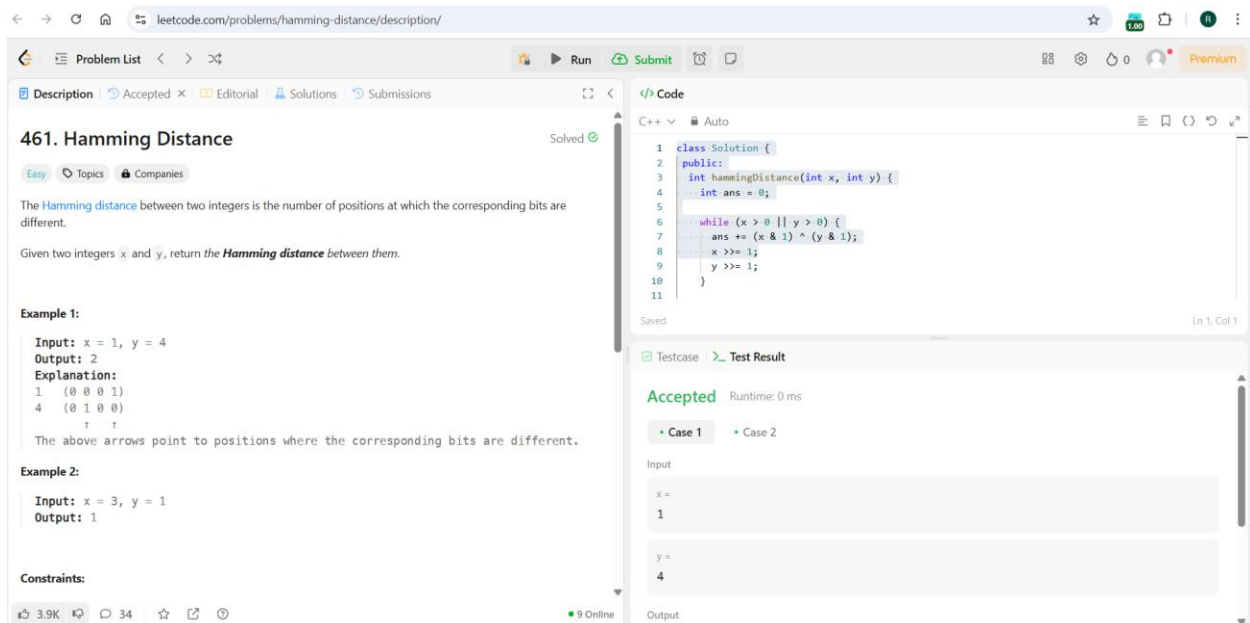
`numRows =`  
`5`

Output

`[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]`

Expected

b.



The screenshot shows the LeetCode interface for problem 461, "Hamming Distance". The problem description states: "The Hamming distance between two integers is the number of positions at which the corresponding bits are different. Given two integers `x` and `y`, return the Hamming distance between them." An example is provided: "Example 1: Input: `x = 1, y = 4`. Output: 2. Explanation: 1 (0 0 0 1), 4 (0 1 0 0). The above arrows point to positions where the corresponding bits are different." The code editor shows a C++ solution that calculates the Hamming distance by counting the number of differing bits. The test result is "Accepted" with a runtime of 0 ms.

**461. Hamming Distance**

Easy Topics Companies

The Hamming distance between two integers is the number of positions at which the corresponding bits are different.

Given two integers `x` and `y`, return the Hamming distance between them.

**Example 1:**

Input: `x = 1, y = 4`  
Output: 2  
Explanation:  
1 (0 0 0 1)  
4 (0 1 0 0)  
The above arrows point to positions where the corresponding bits are different.

**Example 2:**

Input: `x = 3, y = 1`  
Output: 1

**Constraints:**

0 ≤ `x`, `y` ≤ 2<sup>31</sup> - 1

3.9K 34 9 Online

```

1 class Solution {
2 public:
3     int hammingDistance(int x, int y) {
4         int ans = 0;
5
6         while (x > 0 || y > 0) {
7             ans += (x & 1) ^ (y & 1);
8             x >>= 1;
9             y >>= 1;
10        }
11    }
12 }

```

Accepted Runtime: 0 ms

Case 1 Case 2

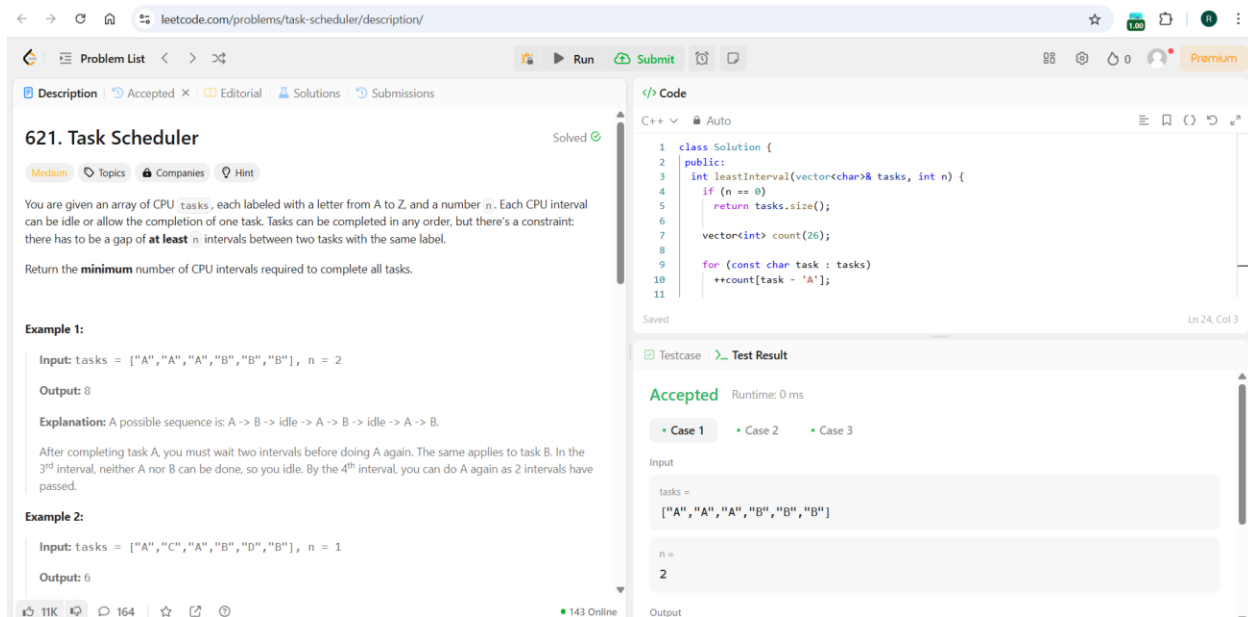
Input

`x =`  
`1`

`y =`  
`4`

Output

C.



The screenshot shows the LeetCode interface for problem 621, "Task Scheduler". The problem description states: "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 shows input tasks = ["A","A","A","B","B","B"], n = 2, output 8, with explanation: "A possible sequence is: A -> B -> idle -> A -> B -> idle -> A -> B." Example 2 shows input tasks = ["A","C","A","B","D","B"], n = 1, output 6. The code editor shows a C++ solution: 

```
1 class Solution {
2 public:
3     int leastInterval(vector<char>& tasks, int n) {
4         if (n == 0)
5             return tasks.size();
6         vector<int> count(26);
7         for (const char task : tasks)
8             ++count[task - 'A'];
9     }
10 }
11
```

 The test result shows "Accepted" with runtime 0 ms for Case 1.

### 3) Learning Outcomes:

- 1) Learned about various miscellaneous problems.



# **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.