



Experiment- 9A

Student Name: Deepanshu

UID: 22BCS16443

Branch: BE-CSE

Section/Group: NTPP 602-A

Semester: 6TH

Date of Performance: 20/03/25

Subject Name: AP Lab-2

Subject Code: 22CSH-352

1. TITLE:

Pascal's Triangle.

2. AIM:

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

3. Algorithm

- **Initialize** the triangle with the first row `[1]`.
- **Iterate** through rows, where each new row is formed by adding adjacent elements from the previous row.
- **Append** the new row to the result and return the triangle after `numRows` iterations.

4. Implemetation/Code

```
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

5. Output:



6. Time Complexity : $O(N^2)$

7. Space Complexity : $O(N^2)$

8. Learning Outcomes:-

- Learn how to build lists iteratively using previous values.
- Understand Pascal's Triangle properties and how each value is derived from previous rows.



Experiment - 9B

Student Name: Deepanshu

UID: 22BCS16443

Branch: BE-CSE

Section/Group: NTPP- 602(A)

Semester: 6TH

Date of Performance: 20/03/25

Subject Name: AP Lab-2

Subject Code: 22CSH-352

1. TITLE:

Number of 1 Bits.

2. AIM:

Given a positive integer n , write a function that returns the number of **set bits** in its **binary** representation (also known as the **Hamming weight**).

3. Algorithm

- **Iterate** while $n > 0$, checking the least significant bit ($n \& 1$).
- **Count the number of 1s** and right shift n ($n \gg= 1$).
- **Return the count** after processing all bits.

4. Implemetation/Code:

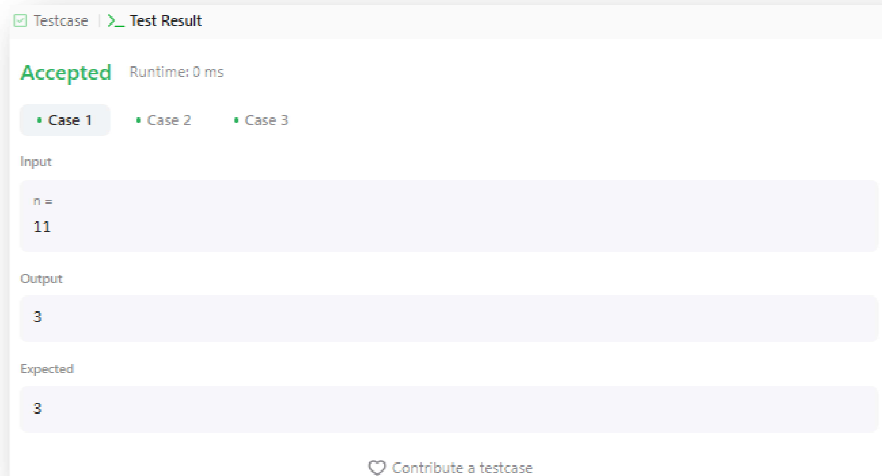
```
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int ans = 0;

        for (int i = 0; i < 32; ++i)
            if ((n >> i) & 1)
                ++ans;

        return ans;
    }
};
```

5. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



6. Time Complexity : $O(\log N)$

7. Space Complexity : $O(1)$

8. Learning Outcomes:-

- Understanding & (AND) and >> (right shift) for efficient bit manipulation
- Counting set bits using simple iteration.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.