*// ^ Set Matrix Zero*

vector<vector<int>> zeroMatrix(vector<vector<int>> &matrix, int n, int m) {

    int col0 = 1;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < m; j++) {

            if (matrix[i][j] == 0) {

                matrix[i][0] = 0;

                if (j != 0)

                    matrix[0][j] = 0;

                else

                    col0 = 0;

            }

        }

    }

    for (int i = 1; i < n; i++) {

        for (int j = 1; j < m; j++) {

            if (matrix[i][j] != 0) {

                if (matrix[i][0] == 0 || matrix[0][j] == 0) {

                    matrix[i][j] = 0;

                }

            }

        }

    }

    if (matrix[0][0] == 0) {

        for (int j = 0; j < m; j++) {

            matrix[0][j] = 0;

        }

    }

    if (col0 == 0) {

        for (int i = 0; i < n; i++) {

            matrix[i][0] = 0;

        }

    }

    return matrix;

}

*// ^ Pascal’s Triangle*

int nCr(int n, int r) {

    long long res = 1;

*// calculating nCr:*

    for (int i = 0; i < r; i++) {

        res = res \* (n - i);

        res = res / (i + 1);

    }

    return res;

}

vector<int> generateRow(int row) {

    long long ans = 1;

    vector<int> ansRow;

    ansRow.push\_back(1); *//inserting the 1st element*

*//calculate the rest of the elements:*

    for (int col = 1; col < row; col++) {

        ans = ans \* (row - col);

        ans = ans / col;

        ansRow.push\_back(ans);

    }

    return ansRow;

}

vector<vector<int>> pascalTriangle(int n) {

    vector<vector<int>> ans;

*//store the entire pascal's triangle:*

    for (int row = 1; row <= n; row++) {

        ans.push\_back(generateRow(row));

    }

    return ans;

}

*// ^ Next Permutation*

vector<int> nextGreaterPermutation(vector<int> &A) {

    int n = A.size(); *// size of the array.*

*// Step 1: Find the break point:*

    int ind = -1; *// break point*

    for (int i = n - 2; i >= 0; i--) {

        if (A[i] < A[i + 1]) {

*// index i is the break point*

            ind = i;

            break;

        }

    }

*// If break point does not exist:*

    if (ind == -1) {

*// reverse the whole array:*

        reverse(A.begin(), A.end());

        return A;

    }

*// Step 2: Find the next greater element*

*//         and swap it with arr[ind]:*

    for (int i = n - 1; i > ind; i--) {

        if (A[i] > A[ind]) {

            swap(A[i], A[ind]);

            break;

        }

    }

*// Step 3: reverse the right half:*

    reverse(A.begin() + ind + 1, A.end());

    return A;

}