DSA MOST LIKELY QUESTIONS

1-Set Matrix zeros:

***Problem Statement:****Given a matrix if an element in the matrix is 0 then you will have to set its entire column and row to 0 and then return the matrix.*

void setZeros(vector<vector<int>> &m)

{

    map<int, int> row, col;

    for (int i = 0; i < m.size(); i++)

    {

        for (int j = 0; j < m[0].size(); j++)

        {

            if (m[i][j] == 0)

            {

                row[i]++;

                col[j]++;

            }

        }

    }

    for (auto x : row)

    {

        for (int i = 0; i < m[0].size(); i++)

        {

            m[x.first][i] = 0;

        }

    }

    for (auto x : col)

    {

        for (int i = 0; i < m.size(); i++)

        {

            m[i][x.first] = 0;

        }

    }

}

2-Pascals Triangle:

***Problem Statement:****Given an integer****N****, return the first****N****rows of Pascal’s triangle.*

vector<vector<int>> generate(int numRows)

{

    vector<vector<int>> ans(numRows);

    for (int i = 0; i < numRows; i++)

    {

        ans[i].resize(i + 1);

        ans[i][0] = 1;

        ans[i][i] = 1;

        for (int j = 1; j < i; j++)

        {

            ans[i][j] = ans[i - 1][j - 1] + ans[i - 1][j];

        }

    }

    return ans;

}

3-Next Permutation

***Problem Statement:****Given an array Arr[] of integers, rearrange the numbers of the given array into the lexicographically next greater permutation of numbers.*

*If such an arrangement is not possible, it must rearrange it as the lowest possible order (i.e., sorted in ascending order).*

void nextPermutation(vector<int> &nums)

{

    int i = nums.size() - 2;

    while (i >= 0 && nums[i + 1] <= nums[i])

    {

        i--;

    }

    if (i >= 0)

    {

        int j = nums.size() - 1;

        while (nums[j] <= nums[i])

        {

            j--;

        }

        swap(nums[i], nums[j]);

    }

    reverse(nums.begin() + i + 1, nums.end());

}

4-Kadane’s Algorithm

***Problem Statement****: Given an integer array arr, find the contiguous sub-array (containing at least one number) which  
has the largest sum and return its sum and print the sub-array.*

int Kandane(vector<int> &nums)

{

    int ans = 0;

    int cur = 0;

    int mx = INT\_MIN;

    if (nums.size() == 1)

        return nums[0];

    for (auto x : nums)

    {

        cur += x;

        mx = max(mx, cur);

        if (cur > 0)

            ans = max(ans, cur);

        else

            cur = 0;

    }

    if (ans == 0)

        return mx;

    else

        return ans;

}