1. https://leetcode.com/problems/two-sum/

// brute force approach

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        int i{}, j{};
        for (i=0;i<nums.size();++i) {
            for(j= i+1;j<nums.size();++j) {
                if (nums[i] + nums [j] == target) {
                    return {i,j};
                }
            }
        }
        return {i,j};
    }
};
```

2. https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        int start{0}, end = nums.size()-1, sum{0};
        while (start < end) {
            sum = nums[start] + nums[end];
            if (target  == sum) {
                return {start+1, end+1};
            }
            else if (sum > target) {
                end--;
            }
            else if (sum < target) {
                start++;
            }
        }
        return {start, end};
    }
};
```

3. https://leetcode.com/problems/merge-sorted-array/

4. https://leetcode.com/problems/pascals-triangle/

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

```
        vector<vector<int>> generateAns{};
        for (int i=0;i<numRows;++i){
            std::vector<int> rowVector(i+1);
            rowVector[0] =1;
            rowVector[i] =1;
            for (int j=1;j<i;++j){
                rowVector[j] = generateAns[i-1][j] +
generateAns[i-1][j-1];
            }
            generateAns.push_back(rowVector);
        }
        return (generateAns);
    }
};
```

5. https://leetcode.com/problems/pascals-triangle-ii/

```
class Solution {
public:
    vector<int> getRow(int row) {
        //row starts from 0, means 3rd row will have 4
elements
        vector<int> ans(row+1,1);

        long prev=1;
        for(int j=1;j<=row-1;j++){
            prev = prev * (row-j+1) / j;
            ans[j] = prev;
        }
        return ans;
    }
};
```

6. https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int minPrices {INT_MAX}, maxProfit{};

        for (int i=0;i<prices.size();++i) {
            if (prices[i] - minPrices > maxProfit) {
                maxProfit = prices[i] - minPrices;
            }
            else if(prices[i] < minPrices) {
                minPrices = prices[i];
            }
        }
        return maxProfit;
```

```
    }
};
```

7. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int maxProfit {};

        for (int i=0;i<prices.size()-1;++i) {
            if (prices[i+1] > prices[i]) {
                maxProfit += prices[i+1] - prices[i];
            }
        }
        return maxProfit;
    }
};
```

8. https://leetcode.com/problems/majority-element/

```
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int count{};
        int element{};
        // Using Moore Voting Algorithm
        // TC -> O(n)
        //  SC -> 0(1)
        for(int i=0;i<nums.size();i++)
        {
            if(count==0)
            {
                count=1;
                element=nums[i];
            }
            else if(element==nums[i])
            {
                ++count;
            }
            else
            {
                --count;
            }
        }
        int c=0;
```

```
        for(int i=0;i<nums.size();++i)
        {
            if(nums[i]==element)
            {
                ++c;
            }
        }
        if(c>(nums.size()/2))
        {
            return element;
        }
        return -1;

    }
};
```

9. https://leetcode.com/problems/majority-element-ii/

```
class Solution {
public:
    vector<int> majorityElement(vector<int>& nums) {
    int cnt1 {}, cnt2 {};
    int el1 {}, el2 {};

    // First pass to find potential majority elements.
    for (int i = 0; i < nums.size(); i++) {
        if (cnt1 == 0 && nums[i] != el2) {
            cnt1 = 1;
            el1 = nums[i];
        }
        else if (cnt2 == 0 && nums[i] != el1) {
            cnt2 = 1;
            el2 = nums[i];
        }
        else if (el1 == nums[i]) {
                ++cnt1;
        }
        else if (el2 == nums[i]) {
                ++cnt2;
        }
        else {
            --cnt1; --cnt2;
        }
    }

    std::vector<int> ans{};
    cnt1 = 0, cnt2 = 0;
```

```
    // Second pass to count occurrences of the potential
majority elements.
    for (int i = 0; i < nums.size(); i++) {
        if (el1 == nums[i]) {
            ++cnt1;
        }
        else if (el2 == nums[i]) {
            ++cnt2;
        }
    }

    if (cnt1 > (nums.size() / 3)) {
        ans.push_back(el1);
    }
    if (cnt2 > (nums.size() / 3)) {
        ans.push_back(el2);
    }
        return (ans);
    }
};
```

10.https://leetcode.com/problems/missing-ranges/

```
class Solution{

public:

    string findMissing(int arr[], int n)

  {

        string strs;

        int l = 0;

        int h = 0;

        // code here

        sort(arr, arr+n);

        for(int i = 0;i <n;i++)

        {

            int h = arr[i] - 1;

            if(h - l > 0)
```

```
            {
                strs += to_string(l) + '-' + to_string(h)+ '
';
            }
            else if(h - l == 0)
            {
                strs+= to_string(l) + ' ';
            }
            else
            {
            }
            l = arr[i] +1;
        }
        if(strs.empty())
        {
            return "-1";
        }
        return strs;
    }
};
```

11.https://leetcode.com/problems/3sum/

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
    std::sort(begin(nums),end(nums));
    vector<vector<int>>ans{};
    set<vector<int>> setUnique{};
    int sum {}, target {};
    for (int i=0;i<nums.size();++i) {
        int j=i+1, k = nums.size()-1;
        while (j<k) {
            sum = nums[i] + nums[j] + nums[k];
```

```
            if (sum == target) {
                setUnique.insert({nums[i], nums[j],
nums[k]});
                ++j; --k;
            }
            else if (sum > target) {
                --k;
            }
            else {
                ++j;
            }
        }
    }
    for (const auto& i: setUnique) {
        ans.push_back(i);
    }
    return ans;
    }
};
```

12.https://leetcode.com/problems/3sum-smaller/

```
class Solution{
        public:
        long long countTriplets(long long arr[], int n, long long sum)
        {
           // Your code goes here
           std::sort(arr,arr+n);
           long long ans = 0;
           for (int i = 0; i < n - 2; i++) {
                int j = i + 1, k = n - 1;
                while (j < k) {
                        if (arr[i] + arr[j] + arr[k] >= sum)
                                --k;
                        else {
```

```
                    ans += (k - j);
                    ++j;
                }
            }
        }
    return ans;
}
};
```

13.https://leetcode.com/problems/3sum-closest/

```cpp
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        sort(nums.begin(),nums.end());
        int sum=0;
        int diff = INT_MAX;
        int ans = INT_MAX;
        for(int i=0;i<nums.size()-1;i++) {
            int j=i+1;
            int k=nums.size()-1;
            while(j<k){
                sum = nums[i]+nums[j]+nums[k];
                if(diff>abs(sum-target)){
                    diff = abs(sum-target);
                    ans = sum;
                }
                if(sum<target)
                    j++;
                else if(sum>target)
                    k--;
                else
                    return ans;
            }
        }
        return ans;
    }
};
```

14.https://leetcode.com/problems/4sum/

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int
target) {
        std::sort(nums.begin(), nums.end());
        std::set<vector<int>> setU;
        int n = nums.size();
        for(int i=0; i<n-3; i++)  {
            for(int j=i+1; j<n-2; j++)  {
                long long sum = target - 0LL - nums[i] -
nums[j];
                int s = j+1, e = n-1;
                while(s < e)    {
                    if(nums[s] + 0LL + nums[e] == sum){
                        setU.insert({nums[i], nums[j],
nums[s], nums[e]});
                        s++;
                        e--;
                    }
                    else if(nums[s] + 0LL + nums[e] >
sum){
                        e--;
                    }
                    else {
                        s++;
                    }
                }
            }
        }
        return  std::vector<std::vector<int>>
(setU.begin(), setU.end());
    }
};
```

15.https://leetcode.com/problems/rotate-image/