```
/*
https://leetcode.com/problems/two-sum/
*/

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

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

class Solution {
public:
    vector<int> twoSum(vector<int>& numbers, int target) {
        vector<int> output{};
        int size = (numbers.size()-1);
        int i{}, j{size},sum {};
        output.clear();

        while(i<j){
            sum=numbers[i]+numbers[j];
            std::cout << sum << std::endl;
            if (sum == target) {
                output.push_back(i+1);
                output.push_back(j+1);
            }
            if (sum<target){
                std::cout << "If \n";
                i++;
            } else {
                std::cout << "Else \n";
                j--;
            }
        }
        return {output};
    }
};

/*
https://leetcode.com/problems/merge-sorted-array/description/
*/
```

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        for (int i=0;i<n;++i){
            nums1[m+i] = nums2[i];
        }
        for (const auto& i:nums1) {
            std::cout << i << " ";
        }
        std::cout << std::endl;
        std::sort(nums1.begin(),nums1.end());
    }
};

/*
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);
    }
};

/*
https://leetcode.com/problems/pascals-triangle-ii/description/
*/

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;
    }
};
```

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

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

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

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

class Solution {
public:
    int maxProfit(vector<int>& prices) {
        if (prices.size() <= 1)
            return (0);
        int maxProfit{};

        for (int i=0;i<=prices.size()-2;++i){
            if (prices[i+1] > prices[i]) {
                maxProfit += prices[i+1] - prices[i];
            }
        }
        return (maxProfit);
    }
};

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

class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int returnValue{};
        int majorityElement = nums.size()/2;
        std::unordered_map<int, int> uMp{};

        for (int i=0;i<nums.size();++i){
            uMp[nums[i]]++;
        }
        for (const auto& i : uMp){
            if (i.second > majorityElement) {
                returnValue = i.first;
            }
        }
        return (returnValue);
    }
};

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

class Solution {
public:
    vector<int> majorityElement(vector<int>& nums) {
        std::vector<int> returnValue{};
        int majorityElement = nums.size()/3;

        std::unordered_map<int, int> uMp{};

        for (int i=0;i<nums.size();++i){
            uMp[nums[i]]++;
        }

        for (const auto& i : uMp){
            if (i.second > majorityElement) {
                //returnValue = i.first;
                returnValue.push_back(i.first);
            }
        }
        return (returnValue);

    }
};


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


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

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

        for (const auto& i: sPush) {
            threeSumOutput.push_back(i);
        }
        return {threeSumOutput};
    }
};

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


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


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++) //fixing 1 pointer
        {
            int j=i+1;
            int k=nums.size()-1;
            while(j<k)  // using 2 pointer logic
            {
                sum = nums[i]+nums[j]+nums[k];
                if(diff>abs(sum-target)) //checking with absolute difference as negative numbers are also prese
nt in the vector
                {
                    diff = abs(sum-target);
                    ans = sum;
                }
                if(sum<target)
                    j++;
                else if(sum>target)
                    k--;
                else
                    return ans; // reduces runtime significantly
            }
        }
        return ans;
    }
};

/*
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());
    }
};

/*
https://leetcode.com/problems/rotate-image/
*/
```