TWO POINTER BEST QUESTIONS

1.) SUBARRAY SUM EQUALS K

```
class Solution {
public:
    int subarraySum(vector<int>& nums, int goal) {
        int ans=0;
        int psum=0;
        unordered_map<int,int> mpp;
        mpp[0]=1;
        for(auto it:nums){
            psum=psum+it;
            if(mpp.find(psum-goal)!=mpp.end()){
                ans=ans+mpp[psum-goal];
            }
            mpp[psum]++;
        }
        return ans;
    }
};
```

2.) BINARY SUBARRAYS WITH SUM

FIRST APPROACH

```
class Solution {
public:
    int numSubarraysWithSum(vector<int>& nums, int goal) {
        long long int i=0;
        long long int j=0;
        long long int sum=0;
        long long int ans=0;
        while(j<nums.size()){
            sum=sum+nums[j];
            while(i<=j && sum>goal){
                sum=sum-nums[i];
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
};
```

SECOND APPROACH

```
class Solution {
public:
    int func(vector<int>& nums, int goal) {
        long long int i=0;
        long long int j=0;
        long long int ans=0;
        long long int sum=0;
        while(j<nums.size()){
            sum=sum+nums[j];
            while(i<=j && sum>goal){
                sum=sum-nums[i];
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
    int numSubarraysWithSum(vector<int>& nums, int goal) {
        return func(nums,goal)-func(nums,goal-1);
    }
};
```

3.) TWO SUM

CODE

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int,int> mp;
        vector<int> ans;
        for(int i=0;i<nums.size();i++){
            int rest=target-nums[i];
            if(mp.find(rest)!=mp.end()){
                ans.push_back(i);
                ans.push_back(mp[rest]);
                return ans;
            }
            mp[nums[i]]=i;
        }
        return {};
    }
};
```

4.) 3 SUM

CODE

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());
        for(int i=0;i<nums.size();i++){
            if(i>0 && nums[i]==nums[i-1])
            continue;

            int j=i+1;
            int k=nums.size()-1;
            while(j<k){
                int sum=nums[i]+nums[j]+nums[k];
                if(sum<0){
                    j++;
                }
                else if(sum>0){
                    k--;
                }
                else{
                    vector<int> temp={nums[i], nums[j], nums[k]};
                    ans.push_back(temp);
                    j++;
                    k--;
                    while(j<k && nums[j]==nums[j-1])
                    j++;
                    while(j<k && nums[k]==nums[k+1])
                    k--;
                }
            }
        }
        return ans;
    }
};
```

5.) 4 SUM

CODE

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());
        for(int i=0;i<nums.size();i++){
            if(i>0 && nums[i]==nums[i-1])
            continue;

            for(int j=i+1;j<nums.size();j++){
                if(j!=i+1 && nums[j]==nums[j-1])
                continue;

                int k=j+1;
                int l=nums.size()-1;
                while(k<l){
                    long long int sum=nums[i];
                    sum+=nums[j];
                    sum+=nums[k];
                    sum+=nums[l];
                    if(sum==target){
                        vector<int> temp={nums[i], nums[j], nums[k], nums[l]};
                        ans.push_back(temp);
                        k++;
                        l--;
                        while(k<l && nums[k]==nums[k-1])
                        k++;
                        while(k<l && nums[l]==nums[l+1])
                        l--;
                    }
                    else if(sum<target){
                        k++;
                    }
                    else{
                        l--;
                    }
                }
            }
        }
        return ans;
    }
};
```

6.) 4 SUM II

CODE

```
class Solution {
public:
    int fourSumCount(vector<int>& A, vector<int>& B, vector<int>& C,
vector<int>& D) {
        map<int,int> mp;
        for(int i:A){
            for(int j:B){
                int sum=i+j;
                mp[-sum]++;
            }
        }
        int count=0;
        for(int k:C){
            for(int l:D){
                int sum=k+l;
                count=count+mp[sum];
            }
        }
        return count;
    }
};
```

7.) TWO SUM II – INPUT ARRAY IS SORTED

CODE

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

8.) 3 SUM CLOSEST

CODE

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        sort(nums.begin(), nums.end());
        int diff=INT_MAX;
        int ans=0;

        for(int i=0;i<nums.size();i++){
            int first=nums[i];
            int s=i+1;
            int e=nums.size()-1;

            while(s<e){
                if(first+nums[s]+nums[e]==target){
                    return target;
                }
                else if(abs(first+nums[s]+nums[e]-target)<diff){
                    diff=abs(first+nums[s]+nums[e]-target);
                    ans=first+nums[s]+nums[e];
                }
                if(first+nums[s]+nums[e]>target)
                e--;
                else
                s++;
            }
        }
        return ans;
    }
};
```

9.) MINIMIZE MAXIMUM PAIR SUM IN ARRAY

CODE

```
class Solution {
public:
    int minPairSum(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int ans=INT_MIN;
        for(int i=0;i<nums.size()/2;i++){
            ans=max(ans, nums[i]+nums[nums.size()-1-i]);
        }
        return ans;
    }
};
```