Q1:- <https://www.google.com/url?q=https://leetcode.com/problems/two-sum/&sa=D&source=editors&ust=1629897830638000&usg=AOvVaw3FqfvdavvXHppB242wpcX6>

class Solution {

public:

vector<int> twoSum(vector<int>v, int target) {

int n=v.size();

sort(v.begin(),v.end());

vector<int>vec;

int hi=n-1;

int lo=0;

while(lo<hi){

int curr=v[lo]+v[hi];

if(curr==target){

vec.push\_back(lo);

vec.push\_back(hi);

break;

}

else if(curr<target){

lo++;

}

else {

hi--;

}

}

return vec;

}

};

Q2:- <https://www.google.com/url?q=https://leetcode.com/problems/merge-sorted-array/&sa=D&source=editors&ust=1629897830639000&usg=AOvVaw1UvvkOiOPFYk_ZWyRwOKug>

class Solution {

public:

void merge(vector<int>& num1, int m, vector<int>& num2, int n) {

vector<int>result(m+n,0);

int i=0,j=0,k=0;

while(i<m and j<n){

if(num1[i]<num2[j]){

result[k]=num1[i];

i++;k++;

}

else if(num1[i]>num2[j]){

result[k]=num2[j];

j++;

k++;

}

else {

result[k]=num1[i];

k++;

result[k]=num2[j];

i++;

j++;

k++;

}

}

while(i<m){

result[k]=num1[i];

i++;k++;

}

while(j<n){

result[k]=num2[j];

j++;

k++;

}

i=0;

while(i<(m+n)){

num1[i]=result[i];

i++;

}

}

};

Q3:- <https://leetcode.com/problems/move-zeroes/>

class Solution {

public:

void moveZeroes(vector<int>& nums) {

int i=0;

int count=0;

int n=nums.size();

for(i=0;i<n-1;i++){

if(nums[i]==0){

count++;

}

}

for(i=0;i<count;i++){

for(int j=0;j<n-1;j++){

if(nums[j]==0){

swap(nums[j],nums[j+1]);

}

}

}

}

};

Q4:- <https://www.google.com/url?q=https://leetcode.com/problems/squares-of-a-sorted-array/&sa=D&source=editors&ust=1629897830640000&usg=AOvVaw22YtWq40vfaFCby4c9Zvii>

class Solution {

public:

vector<int> sortedSquares(vector<int>& nums) {

int i=0;

int n=nums.size();

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

if(nums[i]<0){

nums[i]=abs(nums[i]);

}

}

sort(nums.begin(),nums.end());

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

nums[i]=nums[i]\*nums[i];

}

return nums;

}

};

Q5:-

<https://leetcode.com/problems/running-sum-of-1d-array/>

class Solution {

public:

vector<int> runningSum(vector<int>& nums) {

vector<int>sum(nums.size(),0);

sum[0]=nums[0];

for(int i=1;i<nums.size();i++){

sum[i]+=sum[i-1]+nums[i];

}

return sum;

}

};

Q6:- <https://leetcode.com/problems/find-pivot-index/>

class Solution {

public:

int pivotIndex(vector<int>& nums) {

int i;

int n=nums.size();

vector<int>leftSum(n,0);

vector<int>rightSum(n,0);

leftSum[0]=nums[0];

for(i=1;i<n;i++){

leftSum[i]=leftSum[i-1]+nums[i];

}

rightSum[n-1]=nums[n-1];

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

rightSum[i]=rightSum[i+1]+nums[i];

}

int count=-1;

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

if(leftSum[i]==rightSum[i]){

count=i;

break;

}

}

return count;

}

};

Q7:- <https://leetcode.com/problems/fibonacci-number/>

class Solution {

public:

int fib(int n) {

if(n==0){

return 0;

}

else if(n==1){

return 1;

}

return (fib(n-1)+fib(n-2));

}

};