**Assignment Questions 1 Solutions:**

**Qs.1.**

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

unordered\_map<int, int> mp;

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

if (mp.count(target-nums[i])) {

return {mp[target-nums[i]], i};

}

mp[nums[i]] = i;

}

return {};

}

};

**Qs.2.**

class Solution {

public:

int removeElement(vector<int>& nums, int val) {

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

{

if(nums[i]==val)

{ nums.erase(nums.begin()+i); i--;}

}

return nums.size();

}

};

Qs.3.

class Solution {

public:

int searchInsert(vector<int>& nums, int target) {

int pos=0; int k= nums.size()-1;

if(k==0)

{ if(target<(nums[0]))

{ pos=0; }

else if(nums[0]==target)

pos=0;

else

{

pos=1;

}

}

else {

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

if(nums[i]==target)

pos=i;

else if(target<(nums[0]))

{ pos=0; cout<<"p"; }

else if(target<nums[i] && target>nums[i-1])

pos=i;

else if(target>nums[k])

pos=k+1; }

return pos;

}

};

Qs.4.

class Solution {

public:

int binarySearchTar(vector<int>& nums, int begin, int end, int target){ // recursive function

int middle = (begin + end) / 2;

if(nums[middle] == target){ // base case

return middle;

}else if(nums[middle] < target){ // recurse top half

if(end > middle){ // check so we dont go out of range

if(nums[middle + 1] > target){return middle + 1;}

}

return binarySearchTar(nums, middle + 1, end, target);

}else{ // recurse bottom half

if(middle - 1 >= 0){ // check so we dont go out of range

if(nums[middle - 1] < target){return middle ;}

}

return binarySearchTar(nums, begin, middle - 1, target);

}

}

int searchInsert(vector<int>& nums, int target) {

ios\_base::sync\_with\_stdio(false); // makes tests run faster, look it up elsewhere

cin.tie(nullptr);

if(target >= nums[nums.size()-1]){ // check if target > greatest value

if(target == nums[nums.size()-1]){

return nums.size()-1;

}

return nums.size();

}

if(target <= nums[0]){ // check if target lower than smallest

return 0;

}

return binarySearchTar(nums, 0, nums.size()-1, target);

}

};

Qs.5

class Solution {

public:

void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {

int j=0;

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

{

nums1[i+m]=nums2[j];

j++;

}

sort(nums1.begin(),nums1.end());

}

};

Qs.6.

class Solution {

public:

bool containsDuplicate(vector<int>& nums) {

unordered\_set<int> nums\_set(nums.begin(), nums.end());

return nums\_set.size() != nums.size();

}

};

Qs.7.

class Solution {

public:

void moveZeroes(vector<int>& nums) {

int c=0;

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

{

if(nums[i]==0) { c++; nums.erase(nums.begin()+i); i--;}

}

cout<<c;

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

{

nums.push\_back(0);

}

}

};

Qs.8.

int n = nums.size(),sum=0;

unordered\_set<int> s;

int repeated\_number=0;

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

{

if(s.find(nums[i])!=s.end())

{

repeated\_number=nums[i];

}

else

{

s.insert(nums[i]);

sum+=nums[i];

}

}

int k = n\*(n+1)/2;

return {repeated\_number,k-sum};

}