Assignment 6

**Question 2**

You are given an m x n integer matrix matrix with the following two properties:

* Each row is sorted in non-decreasing order.
* The first integer of each row is greater than the last integer of the previous row.

Given an integer target, return true *if* target *is in* matrix *or* false *otherwise*.

You must write a solution in O(log(m \* n)) time complexity.

**Input:** matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3

**Output:** true

ANS =

// Time complexity - O(M\*Log(K))

// space complexity - O(1)

// where M rows in the grid.

// where K is the maximum number of columns in any row.

class Solution {

public:

int countNegatives(vector<vector<int>>& grid) {

int n = grid.size();

int ans = 0;

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

int ub = upper\_bound(grid[i].rbegin(),grid[i].rend(),-1) - grid[i].rbegin();

ans += ub;

}

return ans;

}

};

**Question 3**

Given an array of integers arr, return *true if and only if it is a valid mountain array*.

Recall that arr is a mountain array if and only if:

* arr.length >= 3
* There exists some i with 0 < i < arr.length - 1 such that:
  + arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
  + arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

**Example 1:**

**Input:** arr = [2,1]

**Output:**

false

ANS =

class Solution {

public:

bool validMountainArray(vector<int>& arr) {

if(arr.size()<3) return false;

int max = INT\_MIN;

int mountain = -1;

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

if(arr[i]>max){

max=arr[i];

mountain =i;

}

}

if(mountain==0 || mountain==arr.size()-1) return false;

for(int i=0; i<mountain; i++){

if(arr[i]>=arr[i+1]){

return false;

}

}

for(int j=mountain+1; j<arr.size(); j++){

if(arr[j]>=arr[j-1]){

return false;

}

}

return true;

}

};

**Question 4**

Given a binary array nums, return *the maximum length of a contiguous subarray with an equal number of* 0 *and* 1.

**Example 1:**

**Input:** nums = [0,1]

**Output:** 2

**Explanation:**

[0, 1] is the longest contiguous subarray with an equal number of 0 and 1.

ANS =

class Solution {

public int search(int[] nums, int target) {

int si= 0;

int ei = nums.length-1;

while(si<=ei){

int mid = (ei-si)/2 + si;

if(nums[mid] == target){

return mid;

}

else if(nums[mid] > target){

ei = mid-1;

}

else{

si = mid+1;

}

}

return -1;

}

}

**Question 6**

An integer array original is transformed into a **doubled** array changed by appending **twice the value** of every element in original, and then randomly **shuffling** the resulting array.

Given an array changed, return original *if* changed *is a* ***doubled*** *array. If* changed *is not a* ***doubled*** *array, return an empty array. The elements in* original *may be returned in* ***any*** *order*.

**Example 1:**

**Input:** changed = [1,3,4,2,6,8]

**Output:** [1,3,4]

**Explanation:** One possible original array could be [1,3,4]:

* Twice the value of 1 is 1 \* 2 = 2.
* Twice the value of 3 is 3 \* 2 = 6.
* Twice the value of 4 is 4 \* 2 = 8.

Other original arrays could be [4,3,1] or [3,1,4].

ANS =

class Solution {

public:

vector<int> findOriginalArray(vector<int>& changed) {

int n= changed.size();

if (n&1)return {};

vector<int> ans;

map<int,int> mp;

sort(changed.begin(),changed.end(),greater<int>());

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

if (mp.find(changed[i]\*2)!=mp.end()){

if (mp[changed[i]\*2]>0){mp[changed[i]\*2]--;}

ans.push\_back(changed[i]);

if (mp[changed[i]\*2]==0) {mp.erase(changed[i]\*2);}

}

else {mp[changed[i]]++;}

}

for (auto val: mp){

if (val.second!=0)return {};

}

return ans;

}

};

**Question 7**

Given a positive integer n, generate an n x n matrix filled with elements from 1 to n2 in spiral order.

**Example 1:**

**Input:** n = 3

**Output:** [[1,2,3],[8,9,4],[7,6,5]]

ANS =

class Solution {

public List<Integer> spiralOrder(int[][] matrix) {

List<Integer> l1 = new ArrayList<Integer>();

//left

int l=0,b=matrix.length-1,r=matrix[0].length-1,t=0;

while(l<=r || t<=b){

if(t<=b){

for(int i=l;i<=r;i++)

l1.add(matrix[t][i]);

t++;

}

if(l<=r){

for(int i=t;i<=b;i++)

l1.add(matrix[i][r]);

r--;

}

if(t<=b){

for(int i=r;i>=l;i--)

l1.add(matrix[b][i]);

b--;

}

if(l<=r){

for(int i=b;i>=t;i--)

l1.add(matrix[i][l]);

l++;

}

}

return l1;

}

}