BINARY SEARCH QUESTIONS

01.) IMPLEMENT LOWER BOUND

CODE

```
int lowerBound(vector<int> arr, int n, int x) {
    int low=0;
    int high=n-1;
    int ans=n;

    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid]>=x){
            ans=mid;
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

02.) IMPLEMENT UPPER BOUND

CODE

```
int upperBound(vector<int> &arr, int x, int n){
    int low=0;
    int high=n-1;
    int ans=n;

    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid]>x){
            ans=mid;
            high=mid-1;
        }
        else
        low=mid+1;
    }
    return ans;
}
```

BINARY SEARCH QUESTIONS

03.) SEARCH INSERT POSITION

CODE

Same as lower bound question that is done above

```
int searchInsert(vector<int>& arr, int x)
{
    int n=arr.size();
    int low=0;
    int high=n-1;
    int ans=n;
    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid]>=x){
            ans=mid;
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

BINARY SEARCH QUESTIONS

04.) FLOOR AND CEIL

CODE

```
int floor(vector<int> &a, int n, int x){
    int ans=-1;
    int low=0;
    int high=n-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(a[mid]<=x){
            ans=a[mid];
            low=mid+1;
        }
        else{
            high=mid-1;
        }
    }
    return ans;
}

int ceil(vector<int> a, int n, int x) {
    int low=0;
    int high=n-1;
    int ans=-1;

    while(low<=high){
        int mid=(low+high)/2;
        if(a[mid]>=x){
            ans=a[mid];
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}

pair<int, int> getFloorAndCeil(vector<int> &a, int n, int x) {
    int ans1=floor(a, n, x);
    int ans2=ceil(a, n, x);
    pair<int,int> p=make_pair(ans1, ans2);
    return p;
}
```

BINARY SEARCH QUESTIONS

05.) FIND FIRST AND LAST OCCURENCES IN AN ARRAY

CODE

```
class Solution {
public:
    int lowerBound(vector<int> arr, int n, int x) {
        int low=0;
        int high=n-1;
        int ans=-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(arr[mid]==x){
                ans=mid;
                high=mid-1;
            }
            else if(arr[mid]<x){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return ans;
    }

    int upperBound(vector<int> &arr, int n, int x){
        int low=0;
        int high=n-1;
        int ans=n;

        while(low<=high){
            int mid=(low+high)/2;
            if(arr[mid]>x){
                ans=mid;
                high=mid-1;
            }
            else
            low=mid+1;
        }
        return ans;
    }

    Vector<int> searchRange(vector<int>& nums, int target) {
        int n=nums.size();
        vector<int> ans;
        int lb=lowerBound(nums, n, target);
        ans.push_back(lb);
```

```
        if(lb==-1)
        ans.push_back(-1);
        else
        ans.push_back(upperBound(nums, n, target)-1);
        return ans;
    }
};
```

06.) SEARCH IN ROTATED SORTED ARRAY

CODE

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int low=0;
        int high=nums.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]==target)
            return mid;

            if(nums[low]<=nums[mid]){
                if(nums[low]<=target && target<=nums[mid]){
                    high=mid-1;
                }
                else{
                    low=mid+1;
                }
            }
            else{
                if(nums[mid]<=target && target<=nums[high]){
                    low=mid+1;
                }
                else{
                    high=mid-1;
                }
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

07.) SEARCH IN ROTATED SORTED ARRAY II

CODE

```cpp
class Solution {
public:
    bool search(vector<int>& nums, int target) {
        int low=0;
        int high=nums.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]==target)
            return true;

            // EXTRA CASE ONLY
            if(nums[low]==nums[mid] && nums[mid]==nums[high]){
                low++;
                high--;
                continue;
            }
            if(nums[low]<=nums[mid]){
                if(nums[low]<=target && target<=nums[mid]){
                    high=mid-1;
                }
                else{
                    low=mid+1;
                }
            }
            else{
                if(nums[mid]<=target && target<=nums[high]){
                    low=mid+1;
                }
                else{
                    high=mid-1;
                }
            }
        }
        return false;
    }
};
```

BINARY SEARCH QUESTIONS

8.) FIND MINIMUM IN ROTATED SORTED ARRAY

CODE

```
class Solution {
public:
    int findMin(vector<int>& nums) {
        int low=0;
        int high=nums.size()-1;
        int ans=INT_MAX;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[low]<=nums[mid]){
                ans=min(ans, nums[low]);
                low=mid+1;
            }
            else{
                high=mid-1;
                ans=min(ans, nums[mid]);
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

9.) FIND MINIMUM IN ROTATED SORTED ARRAY II

CODE

```
class Solution {
public:
    int findMin(vector<int>& nums) {
        int low=0;
        int high=nums.size()-1;
        int ans=INT_MAX;

        while(low<=high){
            int mid=(low+high)/2;
            // EXTRA CASE ONLY
            if(nums[low]==nums[mid] && nums[mid]==nums[high]){
                ans=min(ans, nums[low]);
                low++;
                high--;
                continue;
            }
            if(nums[low]<=nums[mid]){
                ans=min(ans, nums[low]);
                low=mid+1;
            }
            else{
                high=mid-1;
                ans=min(ans, nums[mid]);
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

10.) HOW MANY TIMES ARRAY IS ROTATED

CODE

Slight variation in find minimum in sorted array code

```
#include <bits/stdc++.h>
int findMin(vector<int>& nums) {
    int low=0;
    int high=nums.size()-1;
    int ans=INT_MAX;
    int index=-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(nums[low]<=nums[high]){
            if(nums[low]<ans){
                index=low;
                ans=nums[low];
            }
            break;
        }
        if(nums[low]<=nums[mid]){
            if(nums[low]<ans){
                index=low;
                ans=nums[low];
            }
            low=mid+1;
        }
        else{
            if(nums[mid]<ans){
                index=mid;
                ans=nums[mid];
            }
            high=mid-1;
        }
    }
    return index;
}

int findKRotation(vector<int> &arr){
    int ans=findMin(arr);
    return ans;
}
```

BINARY SEARCH QUESTIONS

11.) SINGLE ELEMENT IN SORTED ARRAY

CODE

```
class Solution {
public:
    int singleNonDuplicate(vector<int>& nums) {
        if(nums.size()==1)
        return nums[0];

        if(nums[0]!=nums[1])
        return nums[0];

        if(nums[nums.size()-1]!=nums[nums.size()-2])
        return nums[nums.size()-1];

        int low=1;
        int high=nums.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]!=nums[mid+1] && nums[mid]!=nums[mid-1])
            return nums[mid];
            if((mid%2==1) && nums[mid-1]==nums[mid]
                || (mid%2==0) && nums[mid]==nums[mid+1]){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

12.) FIND PEAK ELEMENT

CODE

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int n=nums.size();
        if(n==1)
        return 0;
        if(nums[0]>nums[1])
        return 0;
        if(nums[n-1]>nums[n-2])
        return n-1;

        int low=1;
        int high=n-2;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]>nums[mid-1] && nums[mid]>nums[mid+1]){
                return mid;
            }
            else if(nums[mid]>nums[mid-1]){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

13.) PEAK INDEX IN A MOUNTAIN ARRAY

CODE

```
class Solution {
public:
    int peakIndexInMountainArray(vector<int>& nums) {
        int n=nums.size();
        if(n==1)
        return 0;
        if(nums[0]>nums[1])
        return 0;
        if(nums[n-1]>nums[n-2])
        return n-1;

        int low=1;
        int high=n-2;

        while(low<=high){
            int mid=(low+high)/2;
            if(nums[mid]>nums[mid-1] && nums[mid]>nums[mid+1]){
                return mid;
            }
            else if(nums[mid]>nums[mid-1]){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return -1;
    }
};
```

BINARY SEARCH QUESTIONS

14.) FIND SQRT

CODE

```
class Solution {
public:
    int mySqrt(int x) {
        if(x==0)
        return 0;

        int low=1;
        int high=x;
        int ans=INT_MIN;

        while(low<=high){
            int mid=low+((high-low)/2);
            if(mid>x/mid){
                high=mid-1;
            }
            else if(mid<=x/mid){
                ans=max(ans, mid);
                low=mid+1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

15.) FIND THE Nth ROOT OF AN INTEGER

CODE

```cpp
#include <bits/stdc++.h>
using namespace std;

int func(int mid, int n, int m) {
  long long ans = 1;
  for (int i = 1; i <= n; i++) {
      ans = ans * mid;
      if (ans > m) return 2;
  }
  if (ans == m) return 1;
  return 0;
}

int NthRoot(int n, int m) {
  int low = 1, high = m;
  while (low <= high) {
      int mid = (low + high) / 2;
      int midN = func(mid, n, m);
      if (midN == 1) {
          return mid;
      } else if (midN == 0) {
          low = mid + 1;
      } else {
          high = mid - 1;
      }
  }
  return -1;
}
```

BINARY SEARCH QUESTIONS

16.) KOKO EATING BANANAS

CODE

```
class Solution {
public:
    long long func(vector<int> &p, int mid){
        long long sum=0;
        for(int i=0;i<p.size();i++){
            sum=sum+ceil((double)p[i]/(double)mid);
        }
        return sum;
    }

    int minEatingSpeed(vector<int>& p, int h) {
        long long int low=1;
        long long high=*max_element(p.begin(), p.end());
        int ans=-1;

        while(low<=high){
            long long int mid=low+((high-low)/2);
            long long int tot_hr=func(p, mid);
            if(tot_hr<=h){
                ans=mid;
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

17.) MINIMUM NUMBER OF DAYS TO MAKE M BOUQUETS

CODE

```
class Solution {

public:
    bool func(vector<int> &b, int day, int m, int k){
        int count=0;
        int num=0;
        for(int i=0;i<b.size();i++){
            if(b[i]<=day){
                count++;
            }
            else{
                num=num+(count/k);
                count=0;
            }
        }
        num=num+(count/k);
        return num>=m;
    }

    int minDays(vector<int>& b, int m, int k) {
        if(b.size()/k<m)
        return -1;

        int low=*min_element(b.begin(), b.end());
        int high=*max_element(b.begin(), b.end());
        int ans=INT_MAX;

        while(low<=high){
            int mid=(low+high)/2;
            if(func(b, mid, m, k)==true){
                ans=mid;
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

18.) MAXIMUM CANDIES ALLOCATED TO K CHILDREN

CODE

```
class Solution {
public:
    bool func(vector<int> &c, int mid, long long k){
        long long int count=0;
        for(int i=0;i<c.size();i++){
            count=count+(c[i]/mid);
        }
        return count>=k;
    }

    int maximumCandies(vector<int>& c, long long k) {
        int low=1;
        int high=*max_element(c.begin(), c.end());

        while(low<=high){
            int mid=low+((high-low)/2);
            if(func(c, mid, k)){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return high;
    }
};
```

BINARY SEARCH QUESTIONS

19.) FIND THE SMALLEST DIVISOR GIVEN A THRESHOLD VALUES

CODE

```
class Solution {
public:
    int func(vector<int> &nums, int mid){
        int count=0;
        for(int i=0;i<nums.size();i++){
            count=count+ceil((double)nums[i]/(double)mid);
        }
        return count;
    }

    int smallestDivisor(vector<int>& nums, int th) {
        int low=1;
        int high=*max_element(nums.begin(), nums.end());

        while(low<=high){
            int mid=(low+high)/2;
            if(func(nums, mid)<=th){
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

20.) CAPACITY TO SHIP PACKAGES WITHIN D DAYS

CODE

```
class Solution {
public:
    int func(vector<int> &w, int cap){
        int day=1;
        int load=0;
        for(int i=0;i<w.size();i++){
            if(w[i]+load>cap){
                day=day+1;
                load=w[i];
            }
            else{
                load=load+w[i];
            }
        }
        return day;
    }

    int shipWithinDays(vector<int>& w, int days) {
        int low=*max_element(w.begin(), w.end());
        int high=accumulate(w.begin(), w.end(), 0);
        while(low<=high){
            int mid=(low+high)/2;
            int day=func(w, mid);
            if(day<=days)
            high=mid-1;
            else
            low=mid+1;
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

21.) Kth MISSING POSITIVE NUMBERS

CODE

```
class Solution {
public:
    int findKthPositive(vector<int>& arr, int k) {
        int low=0;
        int high=arr.size()-1;

        while(low<=high){
            int mid=(low+high)/2;
            int miss=arr[mid]-(mid+1);
            if(miss<k)
            low=mid+1;
            else
            high=mid-1;
        }
        return k+high+1;
    }
};
```

BINARY SEARCH QUESTIONS

22.) AGGRESSIVE COWS

CODE

```
bool ispossible(vector<int> &stalls, int k, int mid) {
    int cowcount = 1;
    int lastpos = stalls[0];

    for (int i = 0; i < stalls.size(); i++) {
        if (stalls[i] - lastpos >= mid) {
            cowcount++;
            if (cowcount == k) {
                return true;
            }
            lastpos = stalls[i];
        }
    }
    return false;
}

int aggressiveCows(vector<int> &stalls, int k) {
    sort(stalls.begin(), stalls.end());
    int s = 0;
    int e = *max_element(stalls.begin(), stalls.end());
    int ans = -1;

    while (s <= e) {
        int mid = s + (e - s) / 2;
        if (ispossible(stalls, k, mid)) {
            ans = mid;
            s = mid + 1;
        } else {
            e = mid - 1;
        }
    }
    return ans;
}
```

BINARY SEARCH QUESTIONS

23.) MAGNETIC FORCE BETWEEN TWO BALLS

CODE

Same as aggressive cow questions

```cpp
class Solution {
public:
    bool ispossible(vector<int> &stalls, int k, int mid) {
        int cowcount = 1;
        int lastpos = stalls[0];

        for (int i = 0; i < stalls.size(); i++) {
            if (stalls[i] - lastpos >= mid) {
                cowcount++;
                if (cowcount == k) {
                    return true;
                }
                lastpos = stalls[i];
            }
        }
        return false;
    }

    int maxDistance(vector<int>& stalls, int k) {
        sort(stalls.begin(), stalls.end());
        int s = 0;
        int e = *max_element(stalls.begin(), stalls.end());
        int ans = -1;

        while (s <= e) {
            int mid = s + (e - s) / 2;
            if (ispossible(stalls, k, mid)) {
                ans = mid;
                s = mid + 1;
            } else {
                e = mid - 1;
            }
        }
        return ans;
    }
};
```

BINARY SEARCH QUESTIONS

24.) ALLOCATE BOOKS

CODE

```
#include <bits/stdc++.h>
using namespace std;

int func(vector<int> &arr, int pages){
    int stud=1;
    long long pageStud=0;

    for(int i=0;i<arr.size();i++){
        if(pageStud+arr[i]<=pages){
            pageStud=pageStud+arr[i];
        }
        else{
            stud++;
            pageStud=arr[i];
        }
    }
    return stud;
}

int findPages(vector<int>& arr, int n, int m) {
    if(m>n)
    return -1;

    int low=*max_element(arr.begin(), arr.end());
    int high=accumulate(arr.begin(), arr.end(), 0);

    while(low<=high){
        int mid=(low+high)/2;
        int students=func(arr, mid);
        if(students>m)
        low=mid+1;
        else
        high=mid-1;
    }
    return low;
}
```

BINARY SEARCH QUESTIONS

25.) SPLIT ARRAY LARGEST SUM

CODE

Same as ALLOCATE BOOKS

```
class Solution {
public:
    int func(vector<int> &arr, int pages){
        int stud=1;
        long long pageStud=0;

        for(int i=0;i<arr.size();i++){
            if(pageStud+arr[i]<=pages){
                pageStud=pageStud+arr[i];
            }
            else{
                stud++;
                pageStud=arr[i];
            }
        }
        return stud;
    }

    int splitArray(vector<int>& arr, int m) {
        int n=arr.size();
        if(m>n)
        return -1;

        int low=*max_element(arr.begin(), arr.end());
        int high=accumulate(arr.begin(), arr.end(), 0);

        while(low<=high){
            int mid=(low+high)/2;
            int students=func(arr, mid);
            if(students>m)
            low=mid+1;
            else
            high=mid-1;
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

26.) PAINTER'S PARTITION PROBLEM

CODE

Same as ALLOCATED BOOKS

```cpp
#include <bits/stdc++.h>
using namespace std;

int func(vector<int> &arr, int pages){
    int stud=1;
    long long pageStud=0;

    for(int i=0;i<arr.size();i++){
        if(pageStud+arr[i]<=pages){
            pageStud=pageStud+arr[i];
        }
        else{
            stud++;
            pageStud=arr[i];
        }
    }
    return stud;
}

int findLargestMinDistance(vector<int> &arr, int m)
{
    int n=arr.size();
    if(m>n)
    return -1;

    int low=*max_element(arr.begin(), arr.end());
    int high=accumulate(arr.begin(), arr.end(), 0);

    while(low<=high){
        int mid=(low+high)/2;
        int students=func(arr, mid);
        if(students>m)
        low=mid+1;
        else
        high=mid-1;
    }
    return low;

}
```

BINARY SEARCH QUESTIONS

27.) FIND Kth SMALLEST PAIR DISTANCE

CODE

```
class Solution {
public:
    bool isPossible(vector<int>& nums, int k, int mid) {
        int count = 0;
        int left = 0;

        for (int right = 0; right < nums.size(); right++) {
            while (nums[right] - nums[left] > mid) {
                left++;
            }
            count += right - left;
        }
        return count >= k;
    }

    int smallestDistancePair(vector<int>& nums, int k) {
        sort(nums.begin(), nums.end());
        int low = 0, high = nums.back() - nums.front();

        while (low < high) {
            int mid = low + (high - low) / 2;
            if (isPossible(nums, k, mid)) {
                high = mid;
            } else {
                low = mid + 1;
            }
        }
        return low;
    }
};
```

BINARY SEARCH QUESTIONS

28.) MEDIAN OF TWO SORTED ARRAYS

CODE 01

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& a, vector<int>& b) {
        int n1=a.size();
        int n2=b.size();

        int i=0;
        int j=0;
        int n=(n1+n2);

        int ind2=n/2;
        int ind1=ind2-1;

        int count=0;
        int ind1el=-1;
        int ind2el=-1;

        while(i<n1 && j<n2){
            if(a[i]<b[j]){
                if(count==ind1)
                ind1el=a[i];
                if(count==ind2)
                ind2el=a[i];
                count++;
                i++;
            }
            else{
                if(count==ind1)
                ind1el=b[j];
                if(count==ind2)
                ind2el=b[j];
                count++;
                j++;
            }
        }
        while(i<n1){
            if(count==ind1)
            ind1el=a[i];
            if(count==ind2)
            ind2el=a[i];
            count++;
            i++;
        }
```

```
        while(j<n2){
            if(count==ind1)
            ind1el=b[j];
            if(count==ind2)
            ind2el=b[j];
            count++;
            j++;
        }

        if(n%2==1)
        return ind2el;
        return (double)((double)(ind1el+ind2el)/2.0);
    }
};
```

CODE 02

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& a, vector<int>& b) {
        int n1=a.size();
        int n2=b.size();

        if(n1>n2)
        return findMedianSortedArrays(b, a);

        int low=0;
        int high=n1;

        int left=(n1+n2+1)/2;
        int n=n1+n2;

        while(low<=high){
            int mid1=(low+high)>>1;
            int mid2=left-mid1;

            int l1=INT_MIN;
            int l2=INT_MIN;
            int r1=INT_MAX;
            int r2=INT_MAX;

            if(mid1<n1)
            r1=a[mid1];
            if(mid2<n2)
            r2=b[mid2];
            if(mid1-1>=0)
            l1=a[mid1-1];
```

```
            if(mid2-1>=0)
            l2=b[mid2-1];

            if(l1<=r2 && l2<=r1){
                if(n%2==1)
                return max(l1, l2);
                return ((double)(max(l1, l2)+min(r1, r2)))/2.0;
            }
            else if(l1>r2)
            high=mid1-1;
            else
            low=mid1+1;
        }
        return 0;
    }
};
```

BINARY SEARCH QUESTIONS

29.) Kth ELEMENT OF TWO SORTED ARRAYS

CODE

```
#include <bits/stdc++.h>
int kthElement(vector<int> &a, vector<int> &b, int n1, int n2, int k){
    if(n1>n2)
    return kthElement(b, a, n2, n1, k);

    int low=max(k-n2, 0);
    int high=min(k, n1);

    int left=k;
    int n=n1+n2;

    while(low<=high){
        int mid1=(low+high)>>1;
        int mid2=left-mid1;

        int l1=INT_MIN;
        int l2=INT_MIN;
        int r1=INT_MAX;
        int r2=INT_MAX;

        if(mid1<n1)
        r1=a[mid1];
        if(mid2<n2)
        r2=b[mid2];
        if(mid1-1>=0)
        l1=a[mid1-1];
        if(mid2-1>=0)
        l2=b[mid2-1];

        if(l1<=r2 && l2<=r1){
            return max(l1, l2);
        }
        else if(l1>r2)
        high=mid1-1;
        else
        low=mid1+1;
    }
    return 0;

}
```

BINARY SEARCH QUESTIONS

30.) ROW WITH MAXIMUM 1's

CODE

```
class Solution {
public:
    int lowerBound(vector<int> &arr, int n, int x) {
        int low=0;
        int high=n-1;
        int ans=n;
        sort(arr.begin(), arr.end());

        while(low<=high){
            int mid=(low+high)/2;
            if(arr[mid]>=x){
                ans=mid;
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return ans;
    }

    vector<int> rowAndMaximumOnes(vector<vector<int>>& mat) {
        int count_max=0;
        int index=0;

        for(int i=0;i<mat.size();i++){
            int m=mat[i].size();
            int count=m-lowerBound(mat[i], m, 1);
            if(count>count_max){
                count_max=count;
                index=i;
            }
        }
        return {index, count_max};
    }
};
```

BINARY SEARCH QUESTIONS

31.) SEARCH IN 2D MATRIX I

CODE

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& mat, int target) {
        int n=mat.size();
        int m=mat[0].size();

        int low=0;
        int high=(n*m)-1;

        while(low<=high){
            int mid=(low+high)/2;
            int row=mid/m;
            int col=mid%m;

            if(mat[row][col]==target)
            return true;
            else if(mat[row][col]<target)
            low=mid+1;
            else
            high=mid-1;
        }
        return false;
    }
};
```

BINARY SEARCH QUESTIONS

32.) SEARCH IN 2D MATRIX II

CODE

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& mat, int target) {
        int n=mat.size();
        int m=mat[0].size();

        int row=0;
        int col=m-1;

        while(row<n && col>=0){
            if(mat[row][col]==target)
            return true;
            else if(mat[row][col]<target)
            row++;
            else
            col--;
        }
        return false;
    }
};
```

BINARY SEARCH QUESTIONS

33.) FIND PEAK ELEMENT II

CODE

```
class Solution {
public:
    vector<int> findPeakGrid(vector<vector<int>>& mat) {
        int m = mat.size();
        int n = mat[0].size();
        int low = 0;
        int high = n - 1;

        while (low <= high) {
            int mid = low + (high - low) / 2;
            int maxRow = 0;

            for (int i = 0; i < m; i++) {
                if (mat[i][mid] > mat[maxRow][mid]) {
                    maxRow = i;
                }
            }

            bool isLeftValid = false;
            bool isRightValid = false;

            if (mid - 1 >= 0 && mat[maxRow][mid] < mat[maxRow][mid - 1]) {
                isLeftValid = true;
            }

            if (mid + 1 < n && mat[maxRow][mid] < mat[maxRow][mid + 1]) {
                isRightValid = true;
            }

            if (!isLeftValid && !isRightValid) {
                return {maxRow, mid};
            }

            if (isLeftValid) {
                high = mid - 1;
            }
            else {
                low = mid + 1;
            }
        }
        return {-1, -1};
    }
};
```

BINARY SEARCH QUESTIONS

34.) MEDIAN OF A ROW WISE SORTED MATRIX

CODE

```
class Solution {
public:
    int countLessEqual(vector<vector<int>>& matrix, int mid) {
        int count = 0;
        int rows = matrix.size();
        int cols = matrix[0].size();

        for (int i = 0; i < rows; ++i) {
            count += upper_bound(matrix[i].begin(), matrix[i].end(), mid)
                                - matrix[i].begin();
        }
        return count;
    }

    int findMedian(vector<vector<int>>& matrix) {
        int rows = matrix.size();
        int cols = matrix[0].size();

        int low = INT_MAX, high = INT_MIN;
        for (int i = 0; i < rows; ++i) {
            low = min(low, matrix[i][0]);
            high = max(high, matrix[i][cols - 1]);
        }

        int desired = (rows * cols + 1) / 2;

        while (low < high) {
            int mid = low + (high - low) / 2;
            if (countLessEqual(matrix, mid) < desired) {
                low = mid + 1;
            } else {
                high = mid;
            }
        }

        return low;
    }
};
```

THANK YOU !