EASY :

1. Count Negative Numbers in a Sorted Matrix

```
class Solution {
public:
    int countNegatives(vector<vector<int>>& grid)
    {
        int cnt=0;
        int r=grid.size();
        int c=grid[0].size();
        int i=r-1;
        int j=0;

        while(i>=0 && j<c)
        {
            if(grid[i][j]<0)
            {
                cnt+=j-c;
                i--;
            }
            else
            j++;
        }
        return -cnt;
    }
};
```

2. Peak index in a mountain array

```
class Solution {
public:
    int peakIndexInMountainArray(vector<int>& arr)
    {

        int l=0;
        int r=arr.size()-1;

        while(l<r)
        {
            int mid = l+(r-l)/2;
            if(arr[mid]>arr[mid+1])
            r=mid;
            else
            l=mid+1;
        }
        return l;
    }
};
```

3. Find Smallest Letter Greater Than Target

```
class Solution {
public:
    char nextGreatestLetter(vector<char>& letters, char target)
    {
        int l=0;
        int r=letters.size()-1;

        while(l<=r)
        {
            int mid=l+(r-l)/2;
            if(letters[mid]<=target)
            l=mid+1;
            else
            r=mid-1;
        }

        if(r==letters.size()-1 && letters[r]<=target)
        return letters[0];

        return letters[l];
    }
};
```

MEDIUM :

1. Kth Smallest Element in a Sorted Matrix

```
class Solution {
public:
    int kthSmallest(vector<vector<int>>& matrix, int k)
    {
       int n=matrix.size();
        int l=matrix[0][0];
        int r=matrix[n-1][n-1];
        while(l<r)
        {
            int mid=l+(r-l)/2;
            if(count(mid,matrix,n)<k)
            l=mid+1;
            else
            r=mid;
        }
        return r;
    }
    int count(int m,vector<vector<int>>& mat,int n)
    {
        int r=0;
        int c=n-1;
        int f=0;
        while(r<n && c>=0)
        {
            if(mat[r][c]<=m)
            {
                f+=c+1;
                r++;
            }
            else
            c--;
        }
        return f;
    }
};
```

2. Find the Duplicate

```
class Solution {
public:
    int findDuplicate(vector<int>& nums)
    {
        int slow =nums[0];
        int fast =nums[0];

        do{
            slow = nums[slow];
            fast = nums[nums[fast]];
        }while(fast!=slow);


          fast = nums[0];
        while(slow!=fast)
        {
            slow = nums[slow];
            fast = nums[fast];
        }
        return slow;
    }
};
```

3. Search in 2D Matrix II

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target)
    {
        int n=matrix.size();
        int m=matrix[0].size();
        int i=0;
        int j=m-1;

            while(i<n && j>=0)
            {
                if(matrix[i][j]==target)
                return 1;
                else if(matrix[i][j]>target)
                j--;
                else
                i++;

            }
        return false;
    }
};
```

4. Find peak element

```
class Solution {
public:
    int findPeakElement(vector<int>& nums)
    {
        int l=0;
        int r=nums.size()-1;

        while(l<r)
        {
            int mid = l+(r-l)/2;
            if(nums[mid]>nums[mid+1])
            r=mid;
            else
            l=mid+1;

        }
        return l;
    }
};
```

5. Find first and last position of element in sorted array

```
class Solution {
public:
    vector<int> searchRange(vector<int>& nums, int target) {

        int a=-1;
        int b=-1;
        int n=nums.size();
        int l=0;
        int r=n-1;
        while(l<=r)
        {
            int mid=l+(r-l)/2;

            if(nums[mid]==target)
            {
                a = mid;
                b = mid;
                while(a-1>=0 && nums[a-1]==target )
                a--;
                while(b+1<n && nums[b+1]==target )
                b++;

                return {a,b};
            }

            else if(nums[mid]<target)
            l=mid+1;
            else
            r=mid-1;
        }
        return {a,b};
     }
};
```

6. K-diff pairs in an array

```
class Solution {
public:
    bool binary(vector<int>arr,int n,int l,int r)
{
       while(l<=r)
      {
         int mid=l+(r-l)/2;
         if(arr[mid]==n)
         return 1;
         else if(arr[mid]<n)
         l=mid+1;
         else
         r=mid-1;
     }
     return 0;
}
    int findPairs(vector<int>& nums, int k)
    {
       sort(nums.begin(),nums.end());
        int n=nums.size();
        int c=0;
        int f=0;
         for(int i=0;i<n-1;i++)
          {
             if(nums[i]==nums[i+1])
             {
                 if(k==0 && f==0)
                 c++;
                 f=1;
                 continue;
             }
            int value=k+nums[i];
            if(k==0)
            i++;

            bool temp=binary(nums,value,i,n-1);
            if(temp==1)
            c++;
             if(k==0)
             i--;
             f=0;
        }
    return c;
    }};
```

7. Search in rotated sorted array

```
class Solution {
public:
    int search(vector<int>& nums, int target)
    {
       int n=nums.size();
       int l=0;
        int r=n-1;
        if(n<=2)
        {
            if(n==1)
            return (nums[0]==target)?0:-1;
            else
            return (nums[0]==target)?0:(nums[1]==target)?1:-1;
        }
        while(l<=r)
        {
            int mid=(l+r)/2;

            if(nums[mid]==target)
            return mid;


            if(nums[mid]>=nums[l])
            {
                if(target>=nums[l] && target<=nums[mid])
                r=mid-1;
                else
                l=mid+1;
            }
            else
            {
                if(target>=nums[mid] && target<=nums[r])
                l=mid+1;
                else
                r=mid-1;
            }

        }
        return -1;
    }
};
```