Binary Search 👍 we need already sorted array / list.

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

        int start = 0;
        int end = nums.size()-1;

        int mid = start + (end-start)/2;  //if start & end, in case INT_MAX k barabar ya aas pas pahuch jata h to bhi mid confuse n ho

        while(start<=end)
        {
            if(nums[mid]==target)
            {
                return mid;
            }
            else if(nums[mid]<target)
            {
                start = mid +1;
            }
            else
            {
                end = mid-1;
            }
            mid = start + (end-start)/2;
        }

        return -1;
    }
};
```

Ques 1.

https://www.codingninjas.com/studio/problems/first-and-last-position-of-an-element-in-sorted-array_1082549?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2&leftPanelTabValue=PROBLEM

Problem statement

You have been given a sorted array/list ***'arr'*** consisting of ***'n'*** elements. You are also given an integer ***'k'***.

Now, your task is to find the first and last occurrence of 'k' in 'arr'.

Note :

1. If 'k' is not present in the array, then the first and the last occurrence will be -1.
2. 'arr' may contain duplicate elements.

Example:

Input: 'arr' = [0,1,1,5] , 'k' = 1

Output: 1 2

Explanation:
If 'arr' = [0, 1, 1, 5] and 'k' = 1, then the first and last occurrence of 1 will be 1(0 - indexed) and 2.

```
#include<vector>
```

```
using namespace std;
int firstOcc(vector<int>& arr, int n, int key) {

    int s = 0, e = n -1;
    int mid = s + (e - s)/2;
    int ans = -1;
    while(s<=e) {

        if(arr[mid] == key){
            ans = mid;
            e = mid - 1;    // ans k left m jakr check kro khi key already pahle to ni aa chuki h
        }
        else if(key > arr[mid]) {     //Right me jao
            s = mid + 1;
        }
        else if(key < arr[mid]) {       //left me jao
            e = mid - 1;
        }

        mid = s + (e-s)/2;
    }
    return ans;
}

int lastOcc(vector<int>& arr, int n, int key) {

    int s = 0, e = n-1;
    int mid = s + (e-s)/2;
    int ans = -1;
    while(s<=e) {

        if(arr[mid] == key){
            ans = mid;
            s = mid + 1;    // ans k right m jakr check kro khi key already bad m  to ni aa chuki h
        }
        else if(key > arr[mid]) {//Right me jao
            s = mid + 1;
        }
        else if(key < arr[mid]) {//left me jao
            e = mid - 1;
        }

        mid = s + (e-s)/2;
    }
    return ans;
}

pair<int, int> firstAndLastPosition(vector<int>& arr, int n, int k)
{
    pair<int,int> p;
    p.first = firstOcc(arr, n, k);
    p.second = lastOcc(arr, n, k);

    return p;
}
```

Que 2. Total no. of occurrence : (lastOcc - firstOcc) + 1

Que 3.

https://leetcode.com/problems/find-peak-element/ similarly, https://leetcode.com/problems/peak-index-in-a-mountain-array/

Peak of mountains —

You must write an algorithm that runs in `O(log n)` time. In this Question array is ‘partially sorted’

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.
```

Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 5
```

```
Explanation: Your function can return either index number 1 where the peak element is
2, or index number 5 where the peak element is 6.
```

```
class Solution {
public:
   int findPeakElement(vector<int>& a) {
       int s=0;
       int e=a.size()-1;
       int mid = s+(e-s)/2;
       while(s<e)
       {
           if(a[mid]<a[mid+1])
           {
               s=mid+1;
           }
           else
           {
               e=mid;
           }
           mid=s+(e-s)/2;
       }
       return s;

   }
};
```

Solution in O(n) with brute force approach, Question a little different https://leetcode.com/problems/find-the-peaks/

Que 4. Find Pivot in an Sorted & Rotated Array using Binary Search

Que 5. Binary Search in 2d Array

A). https://leetcode.com/problems/search-a-2d-matrix/

```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int rows = matrix.size();
    int cols = matrix[0].size();

    int s = 0, e = rows * cols - 1;

    while(s <= e)
    {
        int mid = s + (e - s)/2;
        int row = mid / cols;
        int col = mid % cols;

        if(matrix[row][col] == target)
            return 1;
        else if(matrix[row][col] < target)
            s = mid + 1;
        else
            e = mid - 1;
    }
    return 0;
```

1	3	5	7
10	11	16	20
23	30	34	60

```
}
```

Time Complexity:

- O(log(rows * cols)) for arrays sorted in both dimensions.
- O(log(rows) + cols) for arrays sorted in one dimension.

B). https://leetcode.com/problems/search-a-2d-matrix-ii/

```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int n = matrix.size();
    int m = matrix[0].size();

    int i = 0, j = m -1;

    while(i < n && j >=0 )
    {


        if(matrix[i][j] == target)
            return 1;
        else if(matrix[i][j] < target)
            i++;
        else
            j--;
    }
    return 0;
}
```

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

#Find minimum element in array in optimize way
#Find minimum element using binary search for partially sorted and rotated sorted array
#Reverse a number in optimize way
#Binary to decimal in optimise way

Que 6. Find pivot element in array where pivot is the index of that element which the continuity of the sorted array like arr = [3, 8, 10, 17, 1] so there pivot is 4 which is index of element 1

```
int getPivot(int arr[], int n)
{
    while(s<e)
    {
        if(arr[mid] >= arr[0])
            {
              s = mid + 1;
            }
            else{
                    e = mid;
            }
     mid = s + (e-s)/2;
    }
}
```

Que 7. Finding sqrt using Binary Search https://leetcode.com/problems/sqrtx/

```
int mySqrt(int n) {
        int s = 0;
    int e = n;
    long long int mid = s + (e-s)/2;
    long long int ans = -1;
    while(s<=e)
    {
        long long int sqaure = mid * mid;
        if(sqaure == n)
        {
            return mid;
        }
        else if(sqaure < n)
        {
            ans = mid;
            s = mid + 1;
        }
        else
        {
            e = mid -1;
        }
        mid = s + (e-s)/2;
    }
    return ans;
    }
```

For finding more precise value of sqrt like sqrt(37) == 6.082

```
double morePrecision(int n,int precision,int tempSol){

double factor =1, ans = tempSol;

for(int i=0;i<precision;i++){
factor = factor /10;

   for(double j = ans; j*j < n; j= j+factor){
                ans = j;
    }
  }
  return ans;
}
```

Que 8. Book Allocation Problem

https://www.codingninjas.com/studio/problems/ayush-and-ninja-test_1097574?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2

```
bool ispossible(int n , int m ,vector<int> time , long long int mid){

          long long int totaltime =0 ;

           long long int days = 1 ;

           for(long long int i = 0 ; i<m ;i++){

           if(totaltime + time[i] <= mid) {

               totaltime += time[i] ;

           }
           else{
               days++;
               if( days>n || time[i]>mid){

                   return false;
               }
               totaltime = time[i];

           }

           }

           return true;
}

long long ayushGivesNinjatest(int n, int m, vector<int> time)

{
```

```
    if(n>m) {
        return -1;
    }
    long long int s = 0 ;
    long long int sum = 0 ;
    for(long long int i=0 ; i<m ; i++) {
        sum += time[i];
    }
    long long int e = sum ;
    long long int mid = s +((e-s)/2) ;
    long long int ans=-1;
    while(s<=e) {
        if(ispossible(n,m,time,mid)) {
            ans = mid ;
            e = mid -1 ;
        }
        else {
            s = mid + 1 ;
        }
        mid = s +((e-s)/2);
    }
     return ans;
}
```

Que 9. Aggressive Cows

https://www.codingninjas.com/studio/problems/aggressive-cows_1082559?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2&leftPanelTabValue=SUBMISSION

```
bool possible(vector<int> &stalls,int k ,int mid){
    int count=1;
    int lastpos = stalls[0];
    for(int i=0;i<stalls.size();i++){
        if(stalls[i] - lastpos >= mid ){
            count++;
            if(count == k){
            return 1;
            }
            lastpos =stalls[i];
        }
    }
    return 0;
}


int aggressiveCows(vector<int> &stalls, int k)
{
   sort(stalls.begin(),stalls.end());
   int s=0;
   int maxi = -1;
   for(int i=0;i<stalls.size();i++){
       maxi=max(maxi,stalls[i]);
   }
   int e = maxi;
   int ans =-1;
   int mid = s +(e-s)/2;
```

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

Que 10. Painter's Algorithm

https://www.codingninjas.com/studio/problems/painter's-partition-problem_1089557?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2&leftPanelTabValue=SUBMISSION

```
bool ispossible(vector<int> &boards,int k ,int mid){
   int pages = 0;
   int Count = 1;
   for(int i=0;i<boards.size();i++){
      if(pages + boards[i] <= mid){
         pages+=boards[i];
      }
      else{
         Count++;
         if(Count > k || boards[i] > mid){
            return 0;
         }
         pages = boards[i];
      }
   }
   return 1;
}

int findLargestMinDistance(vector<int> &boards, int k)
{
   int s =0;
   int sum =0;
   for(int i=0;i<boards.size();i++)
   {
      sum+=boards[i];
   }
   int e = sum;
   int mid = s +(e-s)/2;
   int ans = -1;

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

Que 11. More Problems on Binary Search Advance Concepts

https://www.codingninjas.com/studio/problems/search-in-rotated-sorted-array_1082554?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2

https://www.codingninjas.com/studio/problems/cooking-ninjas_1164174?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2

Que 12. Bubble sort https://www.codingninjas.com/studio/problems/bubble-sort_980524?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2

```
void bubbleSort(vector<int>& arr, int n)
{
    for(int i=1;i<n;i++){
        for(int j=0;j<n-i;j++){
            if(arr[j]>arr[j+1]){
                swap(arr[j],arr[j+1]);
            }
        }
    }
}
```

Que 13. Selection Sort https://www.codingninjas.com/studio/problems/selection-sort_981162?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2&leftPanelTabValue=SUBMISSION

```
#include <bits/stdc++.h>
void selectionSort(vector<int>& arr, int n)
{
    for(int i=0;i<n-1;i++)
```

```
    {
        int minIndex = i;
        for(int j = i+1;j<n;j++){
            if(arr[j]<arr[minIndex]){
                minIndex = j;
            }
        }
        swap(arr[i],arr[minIndex]);
    }
}
```

Que 14. Insertion Sort https://www.codingninjas.com/studio/problems/insertion-sort_3155179?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2

```
#include <bits/stdc++.h>
void insertionSort(int n, vector<int> &arr){
    for(int i=1;i<n;i++){
        int temp = arr[i];
        int j= i-1;

        while(arr[j] > temp && j>=0){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = temp;
    }
}
```

Que 15. Reverse Array

```
void reverseArray(vector<int> &arr , int m)
{
    int s=m+1;
    int e=arr.size()-1;

    while(s<e)
    {
        swap(arr[s++],arr[e--]);
    }
}
```

Que 16. Merged Sorted Array https://leetcode.com/problems/merge-sorted-array/

Que 17. Rotate Array https://leetcode.com/problems/rotate-array/

```
class Solution {
public:
    void rotate(vector<int>& nums, int k) {
        reverse(nums.begin(),nums.end());
        reverse(nums.begin(),nums.begin()+ k % nums.size());
        reverse(nums.begin()+k % nums.size(),nums.end());
    }
};
```

Que 18. Practice Questions

https://leetcode.com/problems/check-if-array-is-sorted-and-rotated/discuss/

https://www.codingninjas.com/studio/problems/reverse-the-array_1262298?utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio3

https://www.codingninjas.com/studio/problems/sum-of-two-arrays_893186?utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_4

https://www.codingninjas.com/studio/problems/check-if-the-string-is-a-palindrome_1062633?utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_5

https://www.geeksforgeeks.org/problems/maximum-occuring-character-1587115620/1

https://www.codingninjas.com/studio/problems/replace-spaces_1172172?utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_5

https://leetcode.com/problems/string-compression/

https://leetcode.com/problems/remove-all-adjacent-duplicates-in-string/

https://leetcode.com/problems/remove-all-occurrences-of-a-substring/

https://leetcode.com/problems/permutation-in-string/

https://leetcode.com/problems/unique-number-of-occurrences/

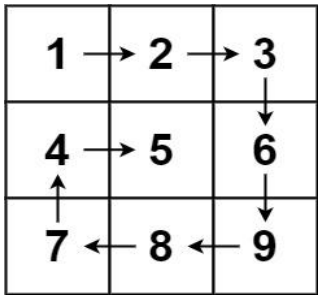
Que 19. Matrix

a. wave


```
vector<int> wavePrint(vector<vector<int>> arr, int nRows, int mCols)
{
   vector<int> v;
   int row=0;
   bool t =1;
   while(mCols>0)
   {
      if(t)
   for(int i=0;i<nRows;i++){
      v.push_back(arr[i][row]);
   }else
   for(int i=nRows-1;i>=0;i--){
      v.push_back(arr[i][row]);
   }
   row++;
   mCols--;
   t = !t;
   }
   return v;
}
```

https://www.codingninjas.com/studio/problems/print-like-a-wave_893268?leftPanelTabValue=SUBMISSION

b. Spiral



https://leetcode.com/problems/spiral-matrix/

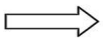
```
class Solution {
public:
   vector<int> spiralOrder(vector<vector<int>>& matrix) {
      int nrow = matrix.size();
      int ncol = matrix[0].size();
```

```
        int count =0;
        int total = nrow*ncol;

        int Startingrow =0;
        int Startingcol =0;
        int Endingrow = nrow-1;
        int Endingcol = ncol -1;

        vector<int> v;

        while(count < total){
            for(int i=Startingcol;count < total && i<=Endingcol;i++){
                v.push_back(matrix[Startingrow][i]);
                count++;
            }
            Startingrow++;
            for(int i = Startingrow;count < total && i<=Endingrow;i++){
                v.push_back(matrix[i][Endingcol]);
                count++;
            }
            Endingcol--;

            for(int i=Endingcol;count < total && i>=Startingcol;i--){
                v.push_back(matrix[Endingrow][i]);
                count++;
            }
            Endingrow--;
            for(int i=Endingrow;count < total && i>=Startingrow;i--){
                v.push_back(matrix[i][Startingcol]);
                count++;
            }

            Startingcol++;

        }
        return v;
    }
};
```

c. Transpose https://leetcode.com/problems/rotate-image/submissions/

```
void rotate(vector<vector<int>>& matrix) {
        // 1. Swap respect to main Diagonal
        //2. reverse each row
        // 3. remember that's all going in place

    int n = matrix.size();

    // Transpose the matrix
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            swap(matrix[i][j], matrix[j][i]);
        }
    }

    // Reverse each row
    for (int i = 0; i < n; i++) {
        reverse(matrix[i].begin(), matrix[i].end());
    }

    }
```

1	2	3
4	5	6
7	8	9



7	4	1
8	5	2
9	6	3

Note : for learning Pointers
https://www.codingninjas.com/studio/guided-paths/pointers

MATHEMATICS

Prime Number

sieve of eratosthenes (the most efficient ways to find all primes smaller than n when n is smaller than 10 million)

```
void SieveOfEratosthenes(int n)            // most suitable for long range
{
  bool prime[n + 1];
   memset(prime, true, sizeof(prime));

   for (int p = 2; p * p <= n; p++) {
      if (prime[p] == true) {
         for (int i = p * p; i <= n; i += p)
            prime[i] = false;
      }
   }

   // Print all prime numbers
   for (int p = 2; p <= n; p++)
      if (prime[p])
         cout << p << " ";
}
```

Time Complexity: O(n*log(log(n)))
Auxiliary Space: O(n)

```
bool isPrime(int num) {                                    //Trial division

   if (num <= 1) {  // Handle cases less than or equal to 1
      return false;
   }
   if (num <= 3) {  // Handle base cases 2 and 3
      return true;
   }
   if (num % 2 == 0 || num % 3 == 0) {  // Check divisibility by 2 and 3
      return false;
   }

   for (int i = 5; i * i <= num; i += 6) {  // Check divisibility by 6k +/- 1
      if (num % i == 0 || num % (i + 2) == 0) {
         return false;
      }
   }

   return true;  // If no divisors found, it's prime
}
```

1. Key Points:
2. Time Complexity: O(sqrt(n)) for each number
3. Suitable for: Small ranges of numbers
4. Not efficient for: Large ranges or frequent primality tests
5. Alternatives for larger ranges: *Sieve of Eratosthenes, Sieve of Sundaram, Miller-Rabin test*

GCD or HCF

1. If a is equal to 0, then the GCD of a and b is b.
2. If b is equal to 0, then the GCD of a and b is a.
3. If a and b are equal, the GCD is a or b.
4. If a > b, call the gcd function recursively with parameters (a – b, b). Or either we can use (a%b, b).
5. Else, call the gcd function recursively with parameters (a, b-a). Or either we can use (a, b%a).

Euclidean Algorithm

Iterative method :

```
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

Recursive method :

```
int gcd(int a, int b) {
    if (b == 0) {
        return a;
    } else {
        return gcd(b, a % b);
    }
}
```

Time Complexity: O(log(max(a, b)))

Binary GCD Algorithm

Good for very long numbers

```
int gcd(int a, int b) {
    if (a == 0) {
        return b;
    }
    if (b == 0) {
        return a;
    }

    int shift = 0;
    while (!(a & 1) && !(b & 1)) {
```

```
        a >>= 1;
        b >>= 1;
        shift++;
    }

    while (!(a & 1)) {
        a >>= 1;
    }
    while (b != 0) {
        while (!(b & 1)) {
            b >>= 1;
            }
        if (a > b) {
            std::swap(a, b);
        }
        b = b - a;
    }

    return a << shift;
}
```

Time Complexity: O(log(a) + log(b))

Pointer

https://www.codingninjas.com/studio/guided-paths/pointers

Static & Dynamic Memory Allocation, Macros, Inline

https://www.codingninjas.com/studio/guided-paths/basics-of-c/content/118785/offering/1381146

Recursion :

https://leetcode.com/problems/valid-palindrome/

https://leetcode.com/problems/fibonacci-number/

https://takeuforward.org/data-structure/reverse-a-given-array/

https://takeuforward.org/data-structure/factorial-of-a-number-iterative-and-recursive/

https://takeuforward.org/data-structure/sum-of-first-n-natural-numbers/

https://takeuforward.org/recursion/print-n-to-1-using-recursion/

https://takeuforward.org/recursion/print-1-to-n-using-recursion/

https://takeuforward.org/recursion/print-name-n-times-using-recursion/

Above are Basic recursion Problem,

IsSortedArray, Linear Search and Binary Search

https://www.codingninjas.com/studio/problems/binary-search_972?leftPanelTab=0&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_11

Merge Sort :

https://www.codingninjas.com/studio/problems/merge-sort_920442?leftPanelTab=0&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_14

```
void merge(vector<int> &arr,int start,int end,int mid){
    int i = start, j = mid + 1, k = 0;
    vector<int> result(end-start+1,0);

    while(i <= mid && j <= end){
        if(arr[i] < arr[j])
            result[k++] = arr[i++];
        else
            result[k++] = arr[j++];
    }

    while(i <= mid)
        result[k++] = arr[i++];
    while(j <= end)
        result[k++] = arr[j++];

    for(int i = start; i <= end; i++){
        arr[i] = result[i-start];
    }
}


void  MergeSort(vector<int> &arr, int start,int end){
    int mid;
    if(start<end){
        mid = start + (end - start)/2;
        MergeSort(arr,start,mid);
        MergeSort(arr,mid+1,end);
        merge(arr,start,end,mid);
    }
}


void mergeSort(vector < int > & arr, int n) {
    MergeSort(arr,0,arr.size()-1);
}
```

Quick Sort :

https://www.codingninjas.com/studio/problems/quick-sort_5844?utm_source=striver&utm_medium=website&utm_campaign=a_zcoursetuf&leftPanelTab=value=SUBMISSION

```
int partition( int arr[], int s, int e) {

    int pivot = arr[s];
    int cnt = 0;
    for(int i = s+1; i<=e; i++) {
        if(arr[i] <=pivot) {
            cnt++;
        }
    }

    //place pivot at right position
    int pivotIndex = s + cnt;
    swap(arr[pivotIndex], arr[s]);

    int i = s, j = e;

    while(i < pivotIndex && j > pivotIndex) {
```

```
        while(arr[i] <= pivot)
        {
            i++;
        }

        while(arr[j] > pivot) {
            j--;
        }

        if(i < pivotIndex && j > pivotIndex) {
            swap(arr[i++], arr[j--]);
        }

    }

    return pivotIndex;

}

void quickSort(int arr[], int s, int e) {

    //base case
    if(s >= e)
        return ;
    int p = partition(arr, s, e);
    quickSort(arr, s, p-1);
    quickSort(arr, p+1, e);

}
```

Subset

https://leetcode.com/problems/subsets/

```
 void solve(vector<int> nums, vector<int> output, int index, vector<vector<int>> &ans){

        if(index >= nums.size()){
            ans.push_back(output);
            return;
        }

        //exclude
        solve(nums,output,index+1,ans);

        //include
        int element = nums[index];
        output.push_back(element);
        solve(nums,output,index+1,ans);

    }

    vector<vector<int>> subsets(vector<int>& nums) {
        vector<vector<int>> ans;
        vector<int> output;
        int index =0;

        solve(nums,output,index,ans);

        return ans;
    }
```

SubSequence

https://www.codingninjas.com/studio/problems/subsequences-of-string_985087?utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_19

```
void solve(string str, vector<string> &ans, string output,int index){
        if(index >= str.size()){
                if(output.size()>0)
                ans.push_back(output);
                return;
        }

        //exclude
        solve(str,ans,output, index+1);

        //inclusion
        output += str[index];

        solve(str,ans,output, index+1);

}

vector<string> subsequences(string str){

        vector<string> ans;
        string output="";
        int index = 0;
        solve(str,ans,output,index);
        return ans;
}
```

Phone keypad Problem

https://leetcode.com/problems/letter-combinations-of-a-phone-number/

```
 void solve(string digits, string output, int index, vector<string> &ans, string mapping[]){

        //Base case
        if(index >= digits.size()){
            ans.push_back(output);
            return;
        }

        int element = digits[index] -'0';
        string value = mapping[element];

        for(int i=0;i<value.size();i++){
            output.push_back(value[i]);
            solve(digits,output,index+1,ans,mapping);
            output.pop_back();
        }
    }
public:
    vector<string> letterCombinations(string digits) {
        vector<string > ans;
        string output="";
        int index = 0;

        if(digits.size() == 0){
            return ans;
        }

        string mapping[10] = { "" , "" , "abc" , "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
        solve(digits,output,index,ans,mapping);
```

```
        return ans;
    }
```

Permutation of String

https://leetcode.com/problems/permutations/

```
void solve(vector<int> nums, vector<vector<int>>& ans, int index){

        //base case
        if(index >= nums.size())
        {
            ans.push_back(nums);
            return ;
        }

        for(int i = index;i<nums.size();i++){
            swap(nums[index],nums[i]);
            solve(nums,ans,index+1);
            //backtracking
            swap(nums[index],nums[i]);
        }

    }
public:
    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans;
        int index =0 ;

        solve(nums,ans,index);
        return ans;
    }
```

Rat in a maze Problem

https://www.geeksforgeeks.org/problems/rat-in-a-maze-problem/1

```
class Solution{
    private:
    bool isSafe(int x, int y, int n, vector<vector<int>> visited, vector<vector<int>> &m){
        if((x >= 0 && x < n) && (y >= 0 && y < n) && visited[x][y] == 0 && m[x][y] == 1 ){
            return 1;
        }
        return 0;
    }
    void solve(vector<vector<int>> &m, int n, vector<string> & ans, int x, int y,vector<vector<int>> visited,string path){

        //you have reached x,y here

        //base case
        if(x == n-1 && y == n-1){
            ans.push_back(path);
            return;
        }

        visited[x][y] = 1;

        //4 choices - D, L, R, U
```

```
    //Down
    int newx = x+1;
    int newy = y;
    if(isSafe(newx, newy, n, visited, m)){
        path.push_back('D');
        solve(m,n,ans,newx,newy,visited,path);
        path.pop_back();
    }

    //left
    newx = x;
    newy = y-1;
    if(isSafe(newx, newy, n, visited, m)){
        path.push_back('L');
        solve(m,n,ans,newx,newy,visited,path);
        path.pop_back();
    }

    //Right
    newx = x;
    newy = y+1;
    if(isSafe(newx, newy, n, visited, m)){
        path.push_back('R');
        solve(m,n,ans,newx,newy,visited,path);
        path.pop_back();
    }

    //UP
    newx = x-1;
    newy = y;
    if(isSafe(newx, newy, n, visited, m)){
        path.push_back('U');
        solve(m,n,ans,newx,newy,visited,path);
        path.pop_back();
    }

    visited[x][y] = 0;
}
public:
vector<string> findPath(vector<vector<int>> &m, int n) {
    vector<string> ans;

    if(m[0][0] == 0){
        return ans;
    }

    int srcx = 0;
    int srcy = 0;

    vector<vector<int>> visited = m;

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            visited[i][j] = 0;
        }
    }

    string path = "";
    solve(m,n,ans,srcx,srcy,visited,path);
    sort(ans.begin(),ans.end());
```

```
        return ans;
    }
};
```

Time Complexity and Space Complexity of Recursive Algorithm

https://www.codingninjas.com/studio/guided-paths/competitive-programming/content/126222/offering/1476042

OOPS and its Concepts

https://www.codingninjas.com/studio/guided-paths/basics-of-c/content/118817/offering/1382190

Linked List

https://www.codingninjas.com/studio/guided-paths/data-structures-algorithms?source=youtube&campaign=YouTube_CodestudioLovebabbar23rdJan&utm_source=youtube&utm_medium=affiliate&utm_campaign=YouTube_CodestudioLovebabbar23rdJan

Questions

Reverse a linked list

https://www.codingninjas.com/studio/problems/reverse-the-singly-linked-list_799897?source=youtube&campaign=Lovebabbarcodestudio_24thJan&utm_source=youtube&utm_medium=affiliate&utm_campaign=Lovebabbarcodestudio_24thJan&leftPanelTabValue=PROBLEM

```
Node* reverseLinkedList(Node* head) {
    if (head == NULL || head->next == NULL)
        return head;

   Node* curr = head;
   Node* prev = NULL;
   Node* next = NULL;

    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }

    return prev;
}
```

Above logic Using recursion

```
void reverse(Node* &head, Node* curr, Node* prev){

//base case
if(curr == NULL){
head = prev;
return;
}

Node* forward = curr -> next;
reverse(head, forward, curr);
curr -> next = prev;
}

Node* reverseLinkedList(Node *head){
Node* curr = head;
Node* prev = NULL;
reverse(head, curr, prev);
return head;
}
```

Reverse a Linked-List Using Recursion

```
Node* reverse(LinkedListNode<int>* head){
```

```
//base case
    if(head == NULL || head-> next == NULL){
        return head;
    }
    Node* chotaHead = reverse(head -> next);
    head -> next -> next = head;
    head -> next = NULL;

    return chotaHead;
}

Node* reverseLinkedList(Node* head){
    return reverse( head);
}
```

Time Complexity for above code : O(n)
Space Complexity for above code : O(n)

Reverse Doubly LinkedList (H/w solution)

```
Node* reverseDoublyLL(Node* head){
if(head==NULL || head->next==NULL)
return head;

Node* temp=head;
Node* store=NULL;
While(temp!=NULL){
//Swapping the addresses
store=temp->prev;
temp->prev=temp->next;
temp->next=store;
temp=temp->prev;}
return store->prev;
}
```

Middle of Linked List

https://www.codingninjas.com/studio/problems/middle-of-linked-list_973250?source=youtube&campaign=Lovebabbarcodestudio_24thJan&utm_source=youtube&utm_medium=affiliate&utm_campaign=Lovebabbarcodestudio_24thJan&leftPanelTabValue=SUBMISSION

Approach 1,

```
int getLength(Node* head){
int len = 0;
while(head != NULL) {
    len++;
    head = head->next;
}
return len;
}

Node* findMiddle(Node* head) {
  int len = getLength(head);
  int ans = (len / 2);

  Node* temp = head;
```

```
 int cnt = 0;

 while (cnt < ans) {
  temp = temp->next;
  cnt++;
 }
 return temp;
}
```

Approach 2

```
Node *findMiddle(Node *head) {
   if(head == NULL || head->next == NULL){
      return head;
   }

   if(head->next->next == NULL){
      return head->next;
   }

   Node* fast = head->next;
   Node* slow = head;

   while(fast != NULL){
      fast = fast->next;
      if(fast != NULL){
         fast = fast->next;
      }

      slow = slow->next;
   }

   return slow;
}
```

Reverse a Liked list in k-Group

https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa3NfVTQzWUNqNG0tOc1TySldLSXF3ZGdQaFBod3xBQ3JtcGtvWkNUFlrTWs5SHNpbDU0M1Vab2MwaWp2NemFxcElfVDNBcUNrTmMQ2U0aXUzRGlNSC03QlJYcHN2REVNMHJBZkRYVURMWU5Ge2EtZDp2eDFrM2doRE4tdWtpaGc5SGNPX1gtam1qY3pKNkMxOEtKYw&q=https%3A%2F%2Fbit.ly%2F3r1rCIP&v=fOvfdmQL0

```
Node* kReverse(Node* head, int k) {
    Node* temp = head;
   int len=0;
   while(temp != NULL){
      temp = temp->next;
      len++;
   }
   // Base Case
   if(len<k || len==0){
    return head;
   }

   //step1: reverse first k nodes
   Node* next = NULL;
   Node* curr = head;
   Node* prev = NULL;
   int count= 0;

   while( curr != NULL && count < k ) {
```

```
        next = curr -> next;
        curr -> next = prev;
        prev = curr;
        curr = next;
        count++;
    }

    //step2: Recursion dekhlega aage ka
    if(next != NULL) {
        head -> next = kReverse(next,k);
    }

    //step3: return head of reversed list
    return prev;
}
```

Time Complexity for above code : O(n)
Space Complexity for above code : O(n)

Check Circularly LinkedList

https://www.codingninjas.com/studio/problems/circularly-linked_1070232?source=youtube&campaign=Love babbar_codestudio_29Sept&utm_source=youtube&utm_medium=affiliate&utm_campaign=Love babbar_codestudio_29Sept&leftPanelTabValue=PROBLEM

```
bool isCircularList(Node* head)
//empty list
    if (head == NULL) {
        return true;
    }
    Node* temp = head -> next;
    while(temp |= NULL && temp != head) {
        temp = temp —> next;
        if(temp == head) {
            return true;
        }
    return false;
}
```

Some more approaches..
i. If there is loop in LL than it must be a circular LL as well, Using slow and fast pointer(floyds algo),
ii. Save data as NULL and when we get by traversing data == NULL so it is circular
iii. We can make map<node*, bool> of taking the trace of visited nodes.

Detect loop

```
Node* floydDetectLoop(Node* head) {

    if(head == NULL)
        return NULL;

    Node* slow = head;
    Node* fast = head;

    while(slow != NULL && fast !=NULL) {

        fast = fast -> next;
        if(fast != NULL) {
            fast = fast -> next;
        }
```

```
    slow = slow -> next;

    if(slow == fast) {
      return slow;
    }
  }

  return NULL;

}
```

Note : take detail description about why floyd works

Remove loop https://www.codingninjas.com/studio/problems/interview-shuriken-42-detect-and-remove-loop_2410491?leftPanelTab=0%3Fsource%3Dyoutube&campaign=YouTube_codestudio_lovebabbar28thjan

```
Node *removeLoop(Node *head)
{
  if(head == NULL){
    return NULL;
  }

  Node *slow = head, *fast = head;

  while(slow != NULL && fast != NULL){
    fast = fast->next;
    if(fast != NULL){
      fast = fast->next;
    }
    slow = slow->next;

    if(slow == fast){
      slow->next = NULL;
      break;
    }
  }

  return head;
}
```

https://www.codingninjas.com/studio/problems/unique-sorted-list_2420283?leftPanelTab=0%3Fsource%3Dyoutube&campaign=YouTube_Lovebabbar29thJan&leftPanelTabValue=SUBMISSION

```
Node * uniqueSortedList(Node * head) {
        //empty List
  if(head == NULL)
    return NULL;

  //non empty list
  Node* curr = head;

  while(curr != NULL) {

    if( (curr -> next != NULL) && curr -> data == curr -> next -> data) {
      Node* next_next = curr ->next -> next;
      Node* nodeToDelete = curr -> next;
      delete(nodeToDelete);
      curr -> next = next_next;
    }
    else //not equal
    {
      curr = curr -> next;
    }
  }
```

```
    return head;
}
```

https://www.codingninjas.com/studio/problems/remove-duplicates-from-unsorted-linked-list_1069331?source=youtube&campaign=YouTube_Lovebabbar29th_Jan&leftPanelTabValue=SUBMISSION

```
#include<bits/stdc++.h>
Node *removeDuplicates(Node *head)
{
    Node* curr = head;
    Node* prev = NULL;
    unordered_map<int, int> visited;

    while(curr != NULL){
        if(!visited[curr->data]){
            visited[curr->data] = 1;
            prev = curr;
            curr = curr -> next;
        }
        else{
            prev -> next = curr -> next;
            delete curr;
        }
        curr = prev -> next;
    }
    return head;
}
```

Note : Vector – upper_bound and lower_bound

```
#include <algorithm> // for lower_bound, upper_bound and sort
#include <iostream>
#include <vector> // for vector

using namespace std;

int main()
{
        // Note that the array is sorted
        int gfg[] = { 5, 5, 5, 6, 6, 6, 7, 7 };

        vector<int> v(gfg, gfg + 8); // 5 5 5 6 6 6 7 7

        vector<int>::iterator lower, upper;
        lower = lower_bound(v.begin(), v.end(), 6);       // 3
        upper = upper_bound(v.begin(), v.end(), 6);       // 6

        cout << "lower_bound for 6 at index "
             << (lower - v.begin()) << '\n';
        cout << "upper_bound for 6 at index "
             << (upper - v.begin()) << '\n';

        return 0;
}
```

⇒ **i).** so simply if **target is not presented** in array than it returns **next index** in lower and upper bound both cases,

Like in above example **target == 4,** than result **index is 1** for both cases and if **target == 8** than **index is 8.**

ii). target is presented in array than in above example answer is mentioned.
Above stl function complexity is **log n.**

Using Binary Search implementing Upper and lower Bound

```
int lowerBound(const std::vector<int>& arr, int target) {
    int left = 0;
    int right = arr.size();

    while (left < right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid;
        }
    }

    return left; // Return the index of the first element not less than target
}


int upperBound(const std::vector<int>& arr, int target) {
    int left = 0;
    int right = arr.size();

    while (left < right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] <= target) {
            left = mid + 1;
        } else {
            right = mid;
        }
    }

    return left; // Return the index of the first element greater than target
}
```