Coding Sheet – Top 50 Questions FOR TCS NQT 2025

Q1-Find sum of elements in a given array.

Input : arr[] = {1, 2, 3}
Output : 6
Explanation: 1 + 2 + 3 = 6

Q2- Given an array arr[] of n elements, write a function to search a given element x in arr[]. – Linear Search

Input: arr[] = {10, 20, 80, 30, 60, 50,110, 100, 130, 170}

x = 110;

Output: Element x is present at index 6

Q3-Given an integer array nums, return true if any value appears **at least twice** in the array, and return false if every element is distinct.(Contains Duplicate -Leetcode)

Input: nums = [1,2,3,1]

Output: true

Q4- Given a non-empty array of integers nums, every element appears *twice* except for one. Find that single one.

Input: nums = [2,2,1]

Output: 1

Q5- Given an array nums of size n, return *the majority element*.

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Input: nums = [3,2,3]

Output: 3

Q6-Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*. (Two Sum - Leetcode)

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Q7-Given an integer x, return true if x is a palindrome, and false otherwise.(Palindrome Number-Leetcode).

Input: x = 121

Output: true

Explanation: 121 reads as 121 from left to right and from right to left.

Q8- Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return *the number of unique elements in* nums.(Remove duplicates from sorted array -Leetcode)

Input: nums = [1,1,2]

Output: 2, nums = [1,2,_]

Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.It does not matter what you leave beyond the returned k (hence they are underscores).

Q9-Given an integer array nums, rotate the array to the right by k steps, where k is non-negative.(Rotate Array -Leetcode)

Input: nums = [1,2,3,4,5,6,7], k = 3

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

Explanation:

rotate 1 steps to the right: [7,1,2,3,4,5,6]

rotate 2 steps to the right: [6,7,1,2,3,4,5]

rotate 3 steps to the right: [5,6,7,1,2,3,4]

Q10- Given an array which may contain duplicates, print all elements and their frequencies.

Input: arr[] = {10,5,10,15,10,5};

Output: 10 3

5 2

15 1

Explanation: 10 occurs 3 times in the array, 5 occurs 2 times in the array

15 occurs 1 time in the array

Q11- Given an array arr[] of length N, The task is to find the maximum and the minimum number in the array.

Input: arr[] = {1, 2, 3, 4, 5}
Output: Maximum is: 5
Minimum is: 1
Explanation: The maximum of the array is 5
and the minimum of the array is 1.

Q12-Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Input: nums = [0,1,0,3,12]

Output: [1,3,12,0,0]

Q13-Given two integer arrays nums1 and nums2, return an array of their intersection Each element in the result must be unique and you may return the result in any order.(Intersection of two arrays- Leetcode)

Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]

Output: [9,4]

Explanation: [4,9] is also accepted.

Q14-Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i].(Product of array except self- Leetcode)

Input: nums = [1,2,3,4]

Output: [24,12,8,6]

Q15- You are given an array prices where prices[i] is the price of a given stock on the i^{th} day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Input: prices = [7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Q16-Given an integer array nums, find a subarray that has the largest product, and return *the product*.(Maximum product subarray-Leetcode)

Input: nums = [2,3,-2,4]

Output: 6

Explanation: [2,3] has the largest product 6.

Q17-Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1. (Binary Search)

Input: nums = [-1,0,3,5,9,12], target = 9

Output: 4

Explanation: 9 exists in nums and its index is 4

Q18- A peak element is an element that is strictly greater than its neighbors.

Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks.

You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.(Find Peak Element – Leetcode)

Input: nums = [1,2,3,1]

Output: 2

Explanation: 3 is a peak element and your function should return the index number 2**.**

Q19- Given an integer array arr, return *the number of distinct bitwise ORs of all the non-empty subarrays of* arr.

The bitwise OR of a subarray is the bitwise OR of each integer in the subarray. The bitwise OR of a subarray of one integer is that integer.A subarray is a contiguous non-empty sequence of elements within an array.(Bitwise ORs of Subarrays) (TCS April 2024)

Input: arr = [1,1,2]

Output: 3

Explanation: The possible subarrays are [1], [1], [2], [1, 1], [1, 2], [1, 1, 2].

These yield the results 1, 1, 2, 1, 3, 3. There are 3 unique values, so the answer is 3.

Q20- Q1-A person has many shoes of different sizes and he wants to arrange them, Calulate the numbers of pairs of shoes.(TCS 2024 April)

I/P- 6 7L 8L 6R 6L 7L 8R

O/P- 2

Q21-A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.(Valid Palindrome - Leetcode)

Input: s = "A man, a plan, a canal: Panama"

Output: true

Explanation: "amanaplanacanalpanama" is a palindrome.

Q22- Given two strings s and t, return true *if* t *is an anagram of* s*, and* false *otherwise*.

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.(Valid Anagram - Leetcode)

Input: s = "anagram", t = "nagaram"

Output: true

Q23- Given a string of lowercase characters from ‘a’ – ‘z’. We need to write a program to print the characters of this string in sorted order.

Input : bbccdefbbaa
Output : aabbbbccdef

Q24- Convert a sentence into its equivalent mobile numeric keypad sequence.(GFG)

Input : HELLO WORLD
Output : 4433555555666096667775553

Q25- Program to count vowels, consonant, digits and special characters in string.

Input : str = "geeks for geeks121"

Output : Vowels: 5

Consonant: 8

Digit: 3

Special Character: 2

Q26- We define the usage of capitals in a word to be right when one of the following cases holds:

- All letters in this word are capitals, like "USA".
- All letters in this word are not capitals, like "leetcode".
- Only the first letter in this word is capital, like "Google".

Given a string word, return true if the usage of capitals in it is right.(Detect Capital – Leetcode)

Q27- Given size of n and list of array elements and we should print if the given element in array is divisible by 3 then replace the element with "Three" and if the element in array is divisible by 5 then replace the element with "Five" if the element divisible by 3 and 5 both then replace the element with "ThreeFive" if the element in the array is not satisfying the above 3 conditions then put the element as it is and print the array.

I/P- N=4

[8,3,9,5]

O/P- 8 Three 9 Five

Q28- You are given a string s consisting of lowercase English letters. A duplicate removal consists of choosing two adjacent and equal letters and removing them.

We repeatedly make duplicate removals on s until we no longer can.

Return *the final string after all such duplicate removals have been made*. It can be proven that the answer is unique.(Remove All Adjacent Duplicates In String -Leetcode)

Input: s = "abbaca"

Output: "ca"

Explanation:

For example, in "abbaca" we could remove "bb" since the letters are adjacent and equal, and this is the only possible move. The result of this move is that the string is "aaca", of which only "aa" is possible, so the final string is "ca".

Q29- A sentence is a string of single-space separated words where each word consists only of lowercase letters.

A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return *a list of all the uncommon words*. You may return the answer in any order.(Uncommon Words from Two Sentences – Leetcode)

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Q30- Given two strings text1 and text2, return *the length of their longest common subsequence*. If there is no common subsequence, return 0.

A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.(Longest Common Subsequence- Leetcode)

Input: text1 = "abcde", text2 = "ace"

Output: 3

Explanation: The longest common subsequence is "ace" and its length is 3.

Q31- Given a 32-bit integer num, return *a string representing its hexadecimal representation*. For negative integers, two's complement method is used.

All the letters in the answer string should be lowercase characters, and there should not be any leading zeros in the answer except for the zero itself.(Convert a Number to Hexadecimal – Leetcode)

Input: num = 26

Output: "1a"

Q32- Given a positive integer num, return true if num is a perfect square or false otherwise.

A perfect square is an integer that is the square of an integer. In other words, it is the product of some integer with itself.(Valid Perfect Square – Leetcode)

Input: num = 16

Output: true

Explanation**:** We return true because 4 * 4 = 16 and 4 is an integer.

Q33- Given an integer n, return true if it is an armstrong number, otherwise return false. an armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits.(TCS April 2024)

I/P-[153, 371, 108]

O/P- [true, true, false]

Q34- Given an integer as input and replace all the '0' with '5' in the integer.

Input: 102

Output: 152

Explanation: All the digits which are '0' is replaced by '5'

Q35-Given an integer array nums, find three numbers whose product is maximum and return the maximum product.(Maximum Product of Three Numbers- Leetcode)

Input: nums = [1,2,3,4]

Output: 24

Q36- Given N, the number of persons. The task is to arrange N person around a circular table.(GFG)

Input: N = 4

Output: 6

Q37- Given a quadratic equation in the form **ax2 + bx + c**, (Only the values of **a**, **b** and **c** are provided) the task is to find the roots of the equation.(GFG)

Input: a = 1, b = -2, c = 1
Output: Roots are real and same 1

Input : a = 1, b = 7, c = 12
Output: Roots are real and different
-3, -4

Q38- Add two fraction a/b and c/d and print answer in simplest form.(GFG)

Input: 1/2 + 3/2
Output: 2/1

Q39- Given a natural number n, print all distinct divisors of it.(GFG)

Input : n = 10
Output: 1 2 5 10

Q40- Given an integer num, repeatedly add all its digits until the result has only one digit, and return it.(Add Digits -Leetcode)

Input: num = 38

Output: 2

Explanation: The process is

38 --> 3 + 8 --> 11

11 --> 1 + 1 --> 2

Since 2 has only one digit, return it.

Q41- Factorial of a Number Using Recursion.

I/P- 5

O/P-120

Q42- You are climbing a staircase. It takes n steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top? (Climbing Stairs – Leetcode)

Input: n = 3

Output: 3

Explanation: There are three ways to climb to the top.

1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step

Q43- There is a robot on an m x n grid. The robot is initially located at the top-left corner (i.e., grid[0][0]). The robot tries to move to the bottom-right corner (i.e., grid[m - 1][n - 1]). The robot can only move either down or right at any point in time.

Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner.(Unique Paths – Leetcode)(TCS 2024 April)

Input: m = 3, n = 2

Output: 3

Explanation: From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:

1. Right -> Down -> Down

2. Down -> Down -> Right

3. Down -> Right -> Down

Q44- You are given an integer array coins representing coins of different denominations and an integer amount representing a total amount of money.

Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.

You may assume that you have an infinite number of each kind of coin.(Coin Change – Leetcode)

Input: coins = [1,2,5], amount = 11

Output: 3

Explanation: 11 = 5 + 5 + 1

Q45- You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array nums representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***. (House Robber – Leetcode)

Input: nums = [1,2,3,1]

Output: 4

Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).

Total amount you can rob = 1 + 3 = 4.

Q46- Implementation of Bubble Sort.

Q47- Implementation of Selection Sort.

Q48- Implementation of Insertion Sort.

Q49- Implementation of Merge Sort.

Q50- Implementation of Quick Sort.

BONUS QUESTIONS-

Q1- Given the head of a singly linked list, reverse the list, and return *the reversed list*.
(Reverse Linked List- Leetcode)

Input: head = [1,2,3,4,5]

Output: [5,4,3,2,1]

Q2- Given head, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.Return true *if there is a cycle in the linked list*. Otherwise, return false.

(Linked List Cycle - Leetcode)

Input: head = [3,2,0,-4], pos = 1

Output: true

Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

Q3- Given the head of a singly linked list, return *the middle node of the linked list*.

(Middle of Linked List – Leetcode)

Input: head = [1,2,3,4,5]

Output: [3,4,5]

Explanation: The middle node of the list is node 3.

Q4- Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

(Valid Parentheses – Leetcode)

Input: s = "()[]{}"

Output: true

Q5- Next Greater Element I – Using Stack (Leetcode).

Input: nums1 = [4,1,2], nums2 = [1,3,4,2]

Output: [-1,3,-1]

Explanation: The next greater element for each value of nums1 is as follows:

- 4 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.

- 1 is underlined in nums2 = [1,3,4,2]. The next greater element is 3.

- 2 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.

Solution of Coding Sheet -

Answer 1 –

Method 1 -

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


int sum(int arr[], int n)
{
    int sum = 0;


    for (int i = 0; i < n; i++)
        sum += arr[i];

    return sum;
}

int main()
{
    int arr[] = { 12, 3, 4, 15 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "Sum of given array is " << sum(arr, n);
    return 0;
}

```

Method 2 -

```cpp
#include <iostream>
using namespace std;


int sum(int arr[], int n)
{
    // base case
    if (n == 0) {
        return 0;
    }
    else {

        return arr[0] + sum(arr + 1, n - 1);
    }
}
int main()
{
    int arr[] = { 12, 3, 4, 15 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << sum(arr, n);
    return 0;

}
```

JAVA CODE –

```java
int sum(int[] arr, int n) {
    // base case
    if (n == 0) {
        return 0;
    } else {
        return arr[0] + sum(Arrays.copyOfRange(arr, 1, n), n - 1);
    }
}
```

Answer 2-

Method 1-

```cpp
int linearSearchLoop(int arr[], int size, int key) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == key)
            return i; // Return the index if the key is found
    }
    return -1; // Return -1 if the key is not found
}

```

Method 2-

```
int linearSearchRecursion(int arr[], int size, int key, int index = 0) {
    if (index >= size)
        return -1; // Return -1 if the key is not found
    if (arr[index] == key)
        return index; // Return the index if the key is found
    return linearSearchRecursion(arr, size, key, index + 1); // Recur for the next element
}
```

JAVA-

```
int linearSearchRecursion(int[] arr, int size, int key, int index) {
    if (index >= size)
        return -1; // Return -1 if the key is not found
    if (arr[index] == key)
        return index; // Return the index if the key is found
    return linearSearchRecursion(arr, size, key, index + 1); // Recur for the next element
}
```

Answer 3-

Method 1-

```
bool containsDuplicate(vector<int>& nums) {
        bool flag = false;
        for(int i =0;i<nums.size();i++){
            for(int j=i+1;j<nums.size();j++){
                if(nums[i] == nums[j]) return true;
            }
        }
        return flag;
    }
```

JAVA-

```
    boolean containsDuplicate(int[] nums) {
    for (int i = 0; i < nums.length; i++) {
        for (int j = i + 1; j < nums.length; j++) {
            if (nums[i] == nums[j]) return true;
        }
    }
    return false;
}
```

Method 2-

```
13      bool containsDuplicate(vector<int>& nums) {
14          return nums.size() > set<int>(nums.begin(),nums.end()).size();
15      }
```

Answer 4-

Method 1-

```
1   int singleNumber(vector<int>& nums) {
2           unordered_map<int,int> a;
3           for(auto x: nums)
4               a[x]++;
5           for(auto z:a)
6               if(z.second==1)
7                   return z.first;
8           return -1;
9       }
```

Method 2-

```
11  int singleNumber(vector<int>& nums) {
12          int ans=0;
13          for(auto x:nums)
14          ans^=x;
15          return ans;
16      }
```

JAVA-

```
18  int singleNumber(int[] nums) {
19      int ans = 0;
20      for (int x : nums) {
21          ans ^= x;
22      }
23      return ans;
24  }
25
```

Answer 5-

Method 1-

```cpp
int majorityElement(vector<int>& nums) {
        int n = nums.size();
        unordered_map<int, int> m;

        for(int i = 0; i < n; i++){
            m[nums[i]]++;
        }
        n = n/2;
        for(auto x: m){
            if(x.second > n){
                return x.first;
            }
        }
        return 0;
    }
```

Method 2-

```cpp
    //Moore Voting Algorithm
    int majorityElement(vector<int>& nums) {
        int count = 0;
        int candidate = 0;

        for (int num : nums) {
            if (count == 0) {
                candidate = num;
            }

            if (num == candidate) {
                count++;
            } else {
                count--;
            }
        }

        return candidate;
    }
```

JAVA –

```java
    int majorityElement(int[] nums) {
    int count = 0;
    int candidate = 0;

    for (int num : nums) {
        if (count == 0) {
            candidate = num;
        }

        if (num == candidate) {
            count++;
        } else {
            count--;
        }
    }

    return candidate;
}
```

Answer 6 -

Method 1-

```cpp
vector<int> twoSum(vector<int>& nums, int target) {
    int n = nums.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (nums[i] + nums[j] == target) {
                return {i, j};
            }
        }
    }
    return {}; // No solution found
```

Method 2-

```cpp
vector<int> twoSum(vector<int>& nums, int target) {
    unordered_map<int, int> numMap;
    int n = nums.size();

    // Build the hash table
    for (int i = 0; i < n; i++) {
        numMap[nums[i]] = i;
    }

    // Find the complement
    for (int i = 0; i < n; i++) {
        int complement = target - nums[i];
        if (numMap.count(complement) && numMap[complement] != i) {
            return {i, numMap[complement]};
        }
    }

    return {}; // No solution found
}
```

JAVA-

```java
int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> numMap = new HashMap<>();
    int n = nums.length;

    // Build the hash map
    for (int i = 0; i < n; i++) {
        numMap.put(nums[i], i);
    }

    // Find the complement
    for (int i = 0; i < n; i++) {
        int complement = target - nums[i];
        if (numMap.containsKey(complement) && numMap.get(complement) != i) {
            return new int[]{i, numMap.get(complement)};
        }
    }

    return new int[]{}; // No solution found
}
```

Answer 7-

```
bool isPalindrome(int x) {
        if (x < 0) {
            return false;
        }

        long long reversed = 0;
        long long temp = x;

        while (temp != 0) {
            int digit = temp % 10;
            reversed = reversed * 10 + digit;
            temp /= 10;
        }

        return (reversed == x);
    }
```

JAVA-

```java
    public boolean isPalindrome(int x) {
    if (x < 0) {
        return false;
    }

    long reversed = 0;
    int temp = x;

    while (temp != 0) {
        int digit = temp % 10;
        reversed = reversed * 10 + digit;
        temp /= 10;
    }

    return reversed == x;
}

```

Answer 8-

```cpp
int removeDuplicates(vector<int>& nums) {
        int j = 1;
        for(int i = 1; i < nums.size(); i++){
            if(nums[i] != nums[i - 1]){
                nums[j] = nums[i];
                j++;
            }
        }
        return j;
    }
```

JAVA –

```java
    public int removeDuplicates(int[] nums) {
    int j = 1;
    for (int i = 1; i < nums.length; i++) {
        if (nums[i] != nums[i - 1]) {
            nums[j] = nums[i];
            j++;
        }
    }
    return j;
}
```

Answer 9-

```cpp
void rotate(vector<int>& nums, int k) {
        int n = nums.size();
        k = k % n;
        vector<int> rotated(n);

        for (int i = 0; i < n; i++) {
            rotated[(i + k) % n] = nums[i];
        }

        for (int i = 0; i < n; i++) {
            nums[i] = rotated[i];
        }
    }
```

JAVA –

```java
    public void rotate(int[] nums, int k) {
    int n = nums.length;
    k = k % n;
    int[] rotated = new int[n];

    for (int i = 0; i < n; i++) {
        rotated[(i + k) % n] = nums[i];
    }

    for (int i = 0; i < n; i++) {
        nums[i] = rotated[i];
    }
}
```

Answer 10-

Method 1-

```cpp
void Frequency(int arr[], int n)
{
    unordered_map<int, int> map;

    for (int i = 0; i < n; i++)
        map[arr[i]]++;

    // Traverse through map and print frequencies
    for (auto x : map)
        cout << x.first << " " << x.second << endl;
}
```

Method 2-

```cpp
void countFreq(int arr[], int n)
{
    vector<bool> visited(n, false);

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

        // Skip this element if already processed
        if (visited[i] == true)
            continue;

        // Count frequency
        int count = 1;
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                visited[j] = true;
                count++;
            }
        }
        cout << arr[i] << " " << count << endl;
    }
}
```

JAVA-

```java
import java.util.Arrays;

public void countFreq(int[] arr, int n) {
    boolean[] visited = new boolean[n];
    Arrays.fill(visited, false);

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

        // Skip this element if already processed
        if (visited[i]) {
            continue;
        }

        // Count frequency
        int count = 1;
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                visited[j] = true;
                count++;
            }
        }
        System.out.println(arr[i] + " " + count);
    }
}
```

Answer 11-

```cpp
void LargestAndSmallest(int arr[], int n)

{

    int small, large;

    small = large = arr[0];

    for(int i = 1; i < n ;i++){

        if(arr[i] < small) small = arr[i];

        if(arr[i] > large)

            large = arr[i];

    }

    cout<<"Smallest Number: "<<small<<"\n";

    cout<<"Largest Number: "<<large<<"\n";

}
```

JAVA-

```java
public void LargestAndSmallest(int[] arr, int n) {
    int small, large;

    small = large = arr[0];

    for (int i = 1; i < n; i++) {
        if (arr[i] < small) small = arr[i];

        if (arr[i] > large) large = arr[i];
    }

    System.out.println("Smallest Number: " + small);
    System.out.println("Largest Number: " + large);
}
```

Answer 12-

```cpp
void moveZeroes(std::vector<int>& nums) {
    int i = 0;  // Pointer to track the position of the next non-zero element

    for (int j = 0; j < nums.size(); j++) {
        if (nums[j] != 0) {
            std::swap(nums[i], nums[j]);  // Swap using std::swap
            i++;
        }
```

JAVA-

```java
    public void moveZeroes(int[] nums) {
    int i = 0;  // Pointer to track the position of the next non-zero element

    for (int j = 0; j < nums.length; j++) {
        if (nums[j] != 0) {
            // Swap using a temporary variable since Java doesn't have std::swap
            int temp = nums[i];
            nums[i] = nums[j];
            nums[j] = temp;
            i++;
        }
    }
}
```

Answer 13-

```cpp
vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
        vector<int>ans;
        map<int,int>mp;
        for(int i=0;i < nums1.size(); i++){
            mp[nums1[i]]++;
        }
        for(int i=0;i < nums2.size(); i++){
            if(mp[nums2[i]] != 0){
            ans.push_back(nums2[i]);
            mp[nums2[i]] = 0;
            }
        }
        return ans;
    }
```

JAVA-

```java
public List<Integer> intersection(int[] nums1, int[] nums2) {
    List<Integer> ans = new ArrayList<>();
    Map<Integer, Integer> mp = new HashMap<>();

    // Populate the map with elements from the first array
    for (int num : nums1) {
        mp.put(num, mp.getOrDefault(num, 0) + 1);
    }

    // Check for intersection with the second array
    for (int num : nums2) {
        if (mp.getOrDefault(num, 0) != 0) {
            ans.add(num);
            mp.put(num, 0); // Ensure the element is only added once
        }
    }

    return ans;
}
```

Answer 14-

```cpp
vector<int> productExceptSelf(vector<int>& nums) {
        int n=nums.size();
        vector<int>left(nums.size(),1);
        vector<int>right(nums.size(),1);
        vector<int>res;
        for(int i=1;i<nums.size();i++)
        {
            left[i]=left[i-1]*nums[i-1];
        }
        for(int i=n-2;i>=0;i--)
        {
            right[i]=right[i+1]*nums[i+1];
        }
        for(int i=0;i<nums.size();i++)
        {
            res.push_back(left[i]*right[i]);
        }
        return res;
    }
```

JAVA-

```java
    public int[] productExceptSelf(int[] nums) {
    int n = nums.length;
    int[] left = new int[n];
    int[] right = new int[n];
    int[] res = new int[n];

    // Initialize the first element of left and last element of right
    left[0] = 1;
    right[n - 1] = 1;

    // Fill the left array
    for (int i = 1; i < n; i++) {
        left[i] = left[i - 1] * nums[i - 1];
    }

    // Fill the right array
    for (int i = n - 2; i >= 0; i--) {
        right[i] = right[i + 1] * nums[i + 1];
    }

    // Build the result array
    for (int i = 0; i < n; i++) {
        res[i] = left[i] * right[i];
    }

    return res;
}
```

Answer 15-

```cpp
int maxProfit(std::vector<int>& prices) {
        int buy = prices[0];
        int profit = 0;
        for (int i = 1; i < prices.size(); i++) {
            if (prices[i] < buy) {
                buy = prices[i];
            } else if (prices[i] - buy > profit) {
                profit = prices[i] - buy;
            }
        }
        return profit;
    }
```

JAVA-

```java
public int maxProfit(int[] prices) {
    int buy = prices[0];
    int profit = 0;

    for (int i = 1; i < prices.length; i++) {
        if (prices[i] < buy) {
            buy = prices[i];
        } else if (prices[i] - buy > profit) {
            profit = prices[i] - buy;
        }
    }

    return profit;
}
```

Answer 16-

```cpp
int maxProduct(vector<int>& nums) {
    int maxi = INT_MIN;
    int prod=1;

    for(int i=0;i<nums.size();i++)
    {
      prod*=nums[i];
      maxi=max(prod,maxi);
      if(prod==0)
       prod=1;
    }
    prod=1;
    for(int i=nums.size()-1;i>=0;i--)
    {
      prod*=nums[i];

      maxi=max(prod,maxi);
      if(prod==0)
       prod=1;
    }
    return maxi;
}
```

JAVA-

```java
    public int maxProduct(int[] nums) {
    int maxi = Integer.MIN_VALUE;
    int prod = 1;

    // Traverse the array from left to right
    for (int i = 0; i < nums.length; i++) {
        prod *= nums[i];
        maxi = Math.max(prod, maxi);
        if (prod == 0) {
            prod = 1;
        }
    }

    // Reset prod and traverse from right to left
    prod = 1;
    for (int i = nums.length - 1; i >= 0; i--) {
        prod *= nums[i];
        maxi = Math.max(prod, maxi);
        if (prod == 0) {
            prod = 1;
        }
    }

    return maxi;
}
```

Answer 17-

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

        while (low <= high) {
            // Finding the mid using integer division
            int mid = low + (high - low) / 2;

            // Target value is present at the middle of the vector
            if (nums[mid] == target) {
                return mid;
            }
            // Target value is present in the low subvector
            else if (target < nums[mid]) {
                high = mid - 1;
            }
            // Target value is present in the high subvector
            else if (target > nums[mid]) {
                low = mid + 1;
            }
        }
```

JAVA-

```java
public int search(int[] nums, int target) {
    int low = 0;
    int high = nums.length - 1;

    while (low <= high) {
        // Finding the mid using integer division
        int mid = low + (high - low) / 2;

        // Target value is present at the middle of the array
        if (nums[mid] == target) {
            return mid;
        }
        // Target value is present in the lower half of the array
        else if (target < nums[mid]) {
            high = mid - 1;
        }
        // Target value is present in the upper half of the array
        else {
            low = mid + 1;
        }
    }

    return -1; // Target not found
}
```

Answer 18-

```java
if(nums.length == 1) return 0; // single element

    int n = nums.length;

    // check if 0th/n-1th index is the peak element
    if(nums[0] > nums[1]) return 0;
    if(nums[n-1] > nums[n-2]) return n-1;

    // search in the remaining array
    int start = 1;
    int end = n-2;

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

JAVA-

```java
public int findPeakElement(int[] nums) {
    if (nums.length == 1) return 0; // single element

    int n = nums.length;

    // Check if 0th or (n-1)th index is the peak element
    if (nums[0] > nums[1]) return 0;
    if (nums[n - 1] > nums[n - 2]) return n - 1;

    // Search in the remaining array
    int start = 1;
    int end = n - 2;

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

    return -1; // Dummy return statement; should not be reached
}
```

Answer 19-

```cpp
int countUniqueBitwiseORs(vector<int>& nums) {
    vector<int> bitwiseORs;
    int start = 0;

    for (int num : nums) {
        int end = bitwiseORs.size();
        bitwiseORs.push_back(num); // Add the current number as a new subarray OR
        for (int i = start; i < end; ++i) {
            int newOR = bitwiseORs[i] | num;
            if (bitwiseORs.back() != newOR) {
                bitwiseORs.push_back(newOR);
            }
        }
        start = end; // Update the start index for the next iteration
    }

    // Use a set to get the count of unique OR results
    return unordered_set<int>(bitwiseORs.begin(), bitwiseORs.end()).size();
}
```

JAVA-

```java
public int countUniqueBitwiseORs(int[] nums) {
    Set<Integer> uniqueORs = new HashSet<>();
    List<Integer> bitwiseORs = new ArrayList<>();
    int start = 0;

    for (int num : nums) {
        int end = bitwiseORs.size();
        bitwiseORs.add(num); // Add the current number as a new subarray OR
        for (int i = start; i < end; ++i) {
            int newOR = bitwiseORs.get(i) | num;
            if (bitwiseORs.get(bitwiseORs.size() - 1) != newOR) {
                bitwiseORs.add(newOR);
            }
        }
        start = end; // Update the start index for the next iteration
    }

    // Use a set to get the count of unique OR results
    uniqueORs.addAll(bitwiseORs);
    return uniqueORs.size();
}
```

Answer 20-

```cpp
int countPairs(const vector<string>& shoes) {
    unordered_map<string, int> leftShoeCount;
    unordered_map<string, int> rightShoeCount;

    // Count occurrences of left and right shoes
    for (const string& shoe : shoes) {
        char side = shoe.back();
        string size = shoe.substr(0, shoe.size() - 1);

        if (side == 'L') {
            leftShoeCount[size]++;
        } else if (side == 'R') {
            rightShoeCount[size]++;
        }
    }

    // Calculate number of pairs
    int pairs = 0;
    for (const auto& entry : leftShoeCount) {
        string size = entry.first;
        int leftCount = entry.second;
        int rightCount = rightShoeCount[size];
        pairs += min(leftCount, rightCount);
    }

    return pairs;
}
```

JAVA-

```java
import java.util.*;

public int countPairs(String[] shoes) {
    Map<String, Integer> leftShoeCount = new HashMap<>();
    Map<String, Integer> rightShoeCount = new HashMap<>();

    // Count occurrences of left and right shoes
    for (String shoe : shoes) {
        char side = shoe.charAt(shoe.length() - 1);
        String size = shoe.substring(0, shoe.length() - 1);

        if (side == 'L') {
            leftShoeCount.put(size, leftShoeCount.getOrDefault(size, 0) + 1);
        } else if (side == 'R') {
            rightShoeCount.put(size, rightShoeCount.getOrDefault(size, 0) + 1);
        }
    }

    // Calculate number of pairs
    int pairs = 0;
    for (Map.Entry<String, Integer> entry : leftShoeCount.entrySet()) {
        String size = entry.getKey();
        int leftCount = entry.getValue();
        int rightCount = rightShoeCount.getOrDefault(size, 0);
        pairs += Math.min(leftCount, rightCount);
    }

    return pairs;
}
```

Answer 21-

```cpp
bool isPalindrome(string s) {
        int start=0;
        int end=s.size()-1;
        while(start<=end){
            if(!isalnum(s[start])){start++; continue;}
            if(!isalnum(s[end])){end--;continue;}
            if(tolower(s[start])!=tolower(s[end]))return false;
            else{
                start++;
                end--;
            }
        }
        return true;
    }
```

JAVA-

```java
public boolean isPalindrome(String s) {
    int start = 0;
    int end = s.length() - 1;

    while (start <= end) {
        // Skip non-alphanumeric characters
        while (start <= end && !Character.isLetterOrDigit(s.charAt(start))) {
            start++;
        }
        while (start <= end && !Character.isLetterOrDigit(s.charAt(end))) {
            end--;
        }

        // Compare characters
        if (start <= end && Character.toLowerCase(s.charAt(start)) != Character.toLowerCase(s.charAt(end))) {
            return false;
        }
        start++;
        end--;
    }

    return true;
}
```

Answer 22-

```cpp
bool isAnagram(string s, string t) {
        unordered_map<char, int> count;

        // Count the frequency of characters in string s
        for (auto x : s) {
            count[x]++;
        }

        // Decrement the frequency of characters in string t
        for (auto x : t) {
            count[x]--;
        }

        // Check if any character has non-zero frequency
        for (auto x : count) {
            if (x.second != 0) {
                return false;
            }
        }

        return true;
    }
```

JAVA-

```java
import java.util.*;

public boolean isAnagram(String s, String t) {
    Map<Character, Integer> count = new HashMap<>();

    // Count the frequency of characters in string s
    for (char c : s.toCharArray()) {
        count.put(c, count.getOrDefault(c, 0) + 1);
    }

    // Decrement the frequency of characters in string t
    for (char c : t.toCharArray()) {
        count.put(c, count.getOrDefault(c, 0) - 1);
    }

    // Check if any character has non-zero frequency
    for (int value : count.values()) {
        if (value != 0) {
            return false;
        }
    }

    return true;
}
```

Answer 23-

```cpp
int main() {
    std::string str = "bbccdefbbaa";
    int n = str.length();

    // Simple bubble sort to sort the string
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (str[j] > str[j + 1]) {
                // Swap the characters
                char temp = str[j];
                str[j] = str[j + 1];
                str[j + 1] = temp;
            }
        }
    }

    // Printing the sorted string
    std::cout << str << std::endl;

    return 0;
}
```

JAVA-

```java
public class Main {
    public static void main(String[] args) {
        String str = "bbccdefbbaa";
        char[] chars = str.toCharArray();
        int n = chars.length;

        // Simple bubble sort to sort the characters
        for (int i = 0; i < n - 1; ++i) {
            for (int j = 0; j < n - i - 1; ++j) {
                if (chars[j] > chars[j + 1]) {
                    // Swap the characters
                    char temp = chars[j];
                    chars[j] = chars[j + 1];
                    chars[j + 1] = temp;
                }
            }
        }

        // Printing the sorted string
        System.out.println(new String(chars));
    }
}
```

Answer 24-

```cpp
int main() {
    std::string input = "hello world";
    std::string output = "";

    // Array to store the mobile keypad sequences for each letter
    std::string keypad[] = {"2", "22", "222",  // a, b, c
                            "3", "33", "333",  // d, e, f
                            "4", "44", "444",  // g, h, i
                            "5", "55", "555",  // j, k, l
                            "6", "66", "666",  // m, n, o
                            "7", "77", "777", "7777",  // p, q, r, s
                            "8", "88", "888",  // t, u, v
                            "9", "99", "999", "9999"}; // w, x, y, z

    for (char c : input) {
        if (c == ' ') {
            output += "0";  // Space corresponds to 0
        } else {
            output += keypad[c - 'a'];  // Convert character to keypad sequence
        }
    }

    std::cout << output << std::endl;

    return 0;
}
```

JAVA –

```java
public class Main {
    public static void main(String[] args) {
        String input = "hello world";
        StringBuilder output = new StringBuilder();

        // Array to store the mobile keypad sequences for each letter
        String[] keypad = {"2", "22", "222",  // a, b, c
                           "3", "33", "333",  // d, e, f
                           "4", "44", "444",  // g, h, i
                           "5", "55", "555",  // j, k, l
                           "6", "66", "666",  // m, n, o
                           "7", "77", "777", "7777",  // p, q, r, s
                           "8", "88", "888",  // t, u, v
                           "9", "99", "999", "9999"}; // w, x, y, z

        for (char c : input.toCharArray()) {
            if (c == ' ') {
                output.append("0");  // Space corresponds to 0
            } else if (Character.isLetter(c)) {
                output.append(keypad[Character.toLowerCase(c) - 'a']);  // Convert character to keypad sequence
            }
        }

        // Printing the output
        System.out.println(output.toString());
    }
}
```

Answer 25-

```cpp
int main() {
    string input = "Hello World! 123";
    int vowels = 0, consonants = 0, digits = 0, specialChars = 0;

    // Iterate through each character in the string
    for (char c : input) {
        if (isalpha(c)) {  // Check if the character is an alphabet
            c = tolower(c);  // Convert to lowercase to simplify checks
            if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
                vowels++;  // Increment vowel counter
            } else {
                consonants++;  // Increment consonant counter
            }
        } else if (isdigit(c)) {
            digits++;  // Increment digit counter
        } else {
            specialChars++;  // Increment special character counter (includes spaces)
        }
    }

    // Output the results
    cout << "Vowels: " << vowels << endl;
    cout << "Consonants: " << consonants << endl;
    cout << "Digits: " << digits << endl;
    cout << "Special Characters: " << specialChars << endl;

    return 0;
}
```

JAVA-

```java
public class Main {
    public static void main(String[] args) {
        String input = "Hello World! 123";
        int vowels = 0, consonants = 0, digits = 0, specialChars = 0;

        // Iterate through each character in the string
        for (char c : input.toCharArray()) {
            if (Character.isAlphabetic(c)) {  // Check if the character is an alphabet
                c = Character.toLowerCase(c);  // Convert to lowercase to simplify checks
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
                    vowels++;  // Increment vowel counter
                } else {
                    consonants++;  // Increment consonant counter
                }
            } else if (Character.isDigit(c)) {
                digits++;  // Increment digit counter
            } else {
                specialChars++;  // Increment special character counter (includes spaces)
            }
        }

        // Output the results
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
        System.out.println("Digits: " + digits);
        System.out.println("Special Characters: " + specialChars);
    }
}

```

Answer 26-

```cpp
bool detectCapitalUse(string word)
    {
        int capital=0;
        int small=0;
        int first=0;
        for(int i=0;i<word.size();i++)
        {
            if(i==0)
            {
                if(word[i]>=65 && word[i]<=90)
                    first++;

            }
            if(word[i]>=97 && word[i]<=122)
                small++;
            else
                 capital++;
        }
        if(first==1 && capital==1 && small==word.size()-1)
            return true;
         if(word.size()==capital && small==0)
            return true;
         if(word.size()==small && capital==0)
            return true;

            return false;


    }
```

JAVA-

```java
    public static boolean detectCapitalUse(String word) {
        int capital = 0;
        int small = 0;
        boolean firstCapital = Character.isUpperCase(word.charAt(0));

        for (int i = 0; i < word.length(); i++) {
            if (Character.isUpperCase(word.charAt(i))) {
                capital++;
            } else {
                small++;
            }
        }

        // All letters are capital or all are small, or only the first letter is capital and the rest are small
        return (capital == word.length()) || (small == word.length()) || (firstCapital && small == word.length() - 1);
    }
}
```

Answer 27-

```cpp
void helperfun(const vector<int>& arr) {
    for (int num : arr) {
        if (num % 3 == 0 && num % 5 == 0) {
            cout << "ThreeFive ";
        } else if (num % 3 == 0) {
            cout << "Three ";
        } else if (num % 5 == 0) {
            cout << "Five ";
        } else {
            cout << num << " ";
        }
    }
    cout << endl;
}
```

JAVA-

```java
public static void helperFun(List<Integer> arr) {
        for (int num : arr) {
            if (num % 3 == 0 && num % 5 == 0) {
                System.out.print("ThreeFive ");
            } else if (num % 3 == 0) {
                System.out.print("Three ");
            } else if (num % 5 == 0) {
                System.out.print("Five ");
            } else {
                System.out.print(num + " ");
            }
        }
        System.out.println();
    }
}
```

Answer 28-

```cpp
string removeDuplicates(string s) {
        string temp = "";
        int i = 0;

        while (i < s.length()) {
            if (temp.empty() || s[i] != temp.back()) {
                temp.push_back(s[i]);
            } else {
                temp.pop_back();
            }
            i++;
        }

        return temp;
    }
```

JAVA-

```java
public static String removeDuplicates(String s) {
        StringBuilder temp = new StringBuilder();

        for (char c : s.toCharArray()) {
            if (temp.length() == 0 || c != temp.charAt(temp.length() - 1)) {
                temp.append(c);
            } else {
                temp.deleteCharAt(temp.length() - 1);
            }
        }

        return temp.toString();
    }
}
```

Answer 29-

```cpp
vector<string> uncommonFromSentences(string A, string B) {

        vector<string> res;
        unordered_map<string, int> mp;

        string word = "";
        for(char ch : A)
        {
            if(ch == ' ')
            {
                mp[word]++;
                word = "";
            }
            else
                word += ch;
        }

        mp[word]++;
        word = "";

        for(char ch : B)
        {
            if(ch == ' ')
            {
                mp[word]++;
                word = "";
            }
            else
                word += ch;
        }

        mp[word]++;

        for(auto i : mp)
            if(i.second == 1) res.push_back(i.first);

        return res;
    }
```

Answer 30-

```cpp
int longestCommonSubsequence(string text1, string text2) {
        int n=text1.size();
        int m= text2.size();

        int dp[n+1][m+1];
        memset(dp,0,sizeof(dp));

        for(int i=n-1 ; i>=0 ; --i){
            for(int j=m-1 ; j>=0 ; --j){
                int res=0;
                if(text1[i]==text2[j]){
                    res = 1 + dp[i+1][j+1];
                }
                else{
                    res = max( dp[i+1][j] ,dp[i][j+1]);
                }
                dp[i][j]=res;
            }
        }

        return dp[0][0];
    }
```

JAVA-

```java
    public static int longestCommonSubsequence(String text1, String text2) {
        int n = text1.length();
        int m = text2.length();

        int[][] dp = new int[n + 1][m + 1];

        // Fill the dp table
        for (int i = n - 1; i >= 0; i--) {
            for (int j = m - 1; j >= 0; j--) {
                if (text1.charAt(i) == text2.charAt(j)) {
                    dp[i][j] = 1 + dp[i + 1][j + 1];
                } else {
                    dp[i][j] = Math.max(dp[i + 1][j], dp[i][j + 1]);
                }
            }
        }

        return dp[0][0];
    }
}
```

Answer 31-

```cpp
int main() {
    int num = 255;
    string hex = "";
    char hexDigits[] = "0123456789abcdef";

    // Handle the special case where the number is 0
    if (num == 0) {
        hex = "0";
    } else {
        // Convert the number to hexadecimal
        while (num > 0) {
            int remainder = num % 16;
            hex = hexDigits[remainder] + hex;  // Prepend the corresponding hex digit
            num /= 16;
        }
    }

    // Output the hexadecimal representation
    cout << "Hexadecimal: " << hex << endl;

    return 0;
}
```

JAVA-

```java
    public static String toHex(int num) {
        if (num == 0) {
            return "0";
        }

        char[] hexDigits = "0123456789abcdef".toCharArray();
        StringBuilder hex = new StringBuilder();

        while (num > 0) {
            int remainder = num % 16;
            hex.insert(0, hexDigits[remainder]);  // Prepend the corresponding hex digit
            num /= 16;
        }

        return hex.toString();
    }
}
```

Answer 32-

```
bool isPerfectSquare(int num) {
        int s=1; int e= num;
        int mid= s+(e-s)/2;
        while(s<=e){ //high-low>=0
            //cout<<mid<<endl;
            if(num%mid==0 && mid==num/mid)return true;
            if (mid > num/mid){  //9 1 9 5 25>num
                e=mid-1;
            }
            else s= mid+1;
            mid= s+(e-s)/2;
        }
        return false;
    }
```

JAVA-

```
    public static boolean isPerfectSquare(int num) {
        int s = 1;
        int e = num;
        while (s <= e) {
            int mid = s + (e - s) / 2;
            if (mid * mid == num) {
                return true;
            } else if (mid > num / mid) {
                e = mid - 1;
            } else {
                s = mid + 1;
            }
        }
        return false;
    }
```

Answer 33-

```
bool isArmStrong(int n, int k) {
    int sum = 0;
    int num = n;
    while (num > 0) {
        int digit = num % 10;
        sum += pow(digit, k);
        num /= 10;
    }
    return sum == n;
}
```

JAVA-

```java
boolean isArmStrong(int n, int k) {
        int sum = 0;
        int num = n;

        while (num > 0) {
            int digit = num % 10;
            sum += Math.pow(digit, k); // Use Math.pow for exponentiation
            num /= 10;
        }

        return sum == n;
    }
```

Answer 34-

```cpp
int main() {
    int num = 102030;

    // Convert the integer to a string
    string numStr = to_string(num);

    // Replace all '0' characters with '5'
    for (char &c : numStr) {
        if (c == '0') {
            c = '5';
        }
    }

    // Convert the string back to an integer
    int newNum = stoi(numStr);

    // Output the modified integer
    cout << "Modified number: " << newNum << endl;

    return 0;
}
```

JAVA-

```java
static void main() {
        int num = 102030;

        // Convert the integer to a string
        String numStr = Integer.toString(num);

        // Replace all '0' characters with '5'
        StringBuilder modifiedStr = new StringBuilder();
        for (char c : numStr.toCharArray()) {
            if (c == '0') {
                modifiedStr.append('5');
            } else {
                modifiedStr.append(c);
            }
        }

        // Convert the string back to an integer
        int newNum = Integer.parseInt(modifiedStr.toString());

        // Output the modified integer
        System.out.println("Modified number: " + newNum);
    }
```

Answer 35-

```
int maximumProduct(int[] nums) {
        int n = nums.length;
        Arrays.sort(nums);
        int m1 = nums[n-1]*nums[n-2]*nums[n-3];
        int m2 = nums[0]*nums[1]*nums[n-1];
        if(m1>m2){
            return m1;
        }
        else{
            return m2;
        }
```

JAVA-

```
public static int maximumProduct(int[] nums) {
    Arrays.sort(nums);
    int m1 = nums[nums.length - 1] * nums[nums.length - 2] * nums[nums.length - 3];
    int m2 = nums[0] * nums[1] * nums[nums.length - 1];
    return Math.max(m1, m2);
}
```

Answer 36-

```
int Circular(int n)
{

    int Result = 1;

    while (n > 0) {
        Result = Result * n;
        n--;
    }

    return Result;
}
```

JAVA-

```
public static int circular(int n) {
    int result = 1;
    while (n > 0) {
        result *= n;
        n--;
    }
    return result;
}
```

Answer 37-

```
void findRoots(int a, int b, int c)
{
    // If a is 0, then equation is not quadratic, but
    // linear
    if (a == 0) {
        cout << "Invalid";
        return;
    }

    int d = b * b - 4 * a * c;
    double sqrt_val = sqrt(abs(d));

    if (d > 0) {
        cout << "Roots are real and different \n";
        cout << (double)(-b + sqrt_val) / (2 * a) << "\n"
             << (double)(-b - sqrt_val) / (2 * a);
    }
    else if (d == 0) {
        cout << "Roots are real and same \n";
        cout << -(double)b / (2 * a);
    }
    else // d < 0
    {
        cout << "Roots are complex \n";
        cout << -(double)b / (2 * a) << " + i"
             << sqrt_val / (2 * a) << "\n"
             << -(double)b / (2 * a) << " - i"
             << sqrt_val / (2 * a);
    }
}
```

JAVA-

```
public static void findRoots(int a, int b, int c) {
    // If a is 0, then equation is not quadratic, but linear
    if (a == 0) {
        System.out.println("Invalid");
        return;
    }

    int d = b * b - 4 * a * c;
    double sqrtVal = Math.sqrt(Math.abs(d));

    if (d > 0) {
        System.out.println("Roots are real and different");
        System.out.println((-b + sqrtVal) / (2 * a));
        System.out.println((-b - sqrtVal) / (2 * a));
    } else if (d == 0) {
        System.out.println("Roots are real and same");
        System.out.println(-(double)b / (2 * a));
    } else { // d < 0
        System.out.println("Roots are complex");
        System.out.println(-(double)b / (2 * a) + " + i" + sqrtVal / (2 * a));
        System.out.println(-(double)b / (2 * a) + " - i" + sqrtVal / (2 * a));
    }
}
```

Answer 38-

```c
// Function to return gcd of a and b
int gcd(int a, int b)
{
    if (a == 0)
        return b;
    return gcd(b%a, a);
}

// Function to convert the obtained fraction
// into it's simplest form
void lowest(int &den3, int &num3)
{
    // Finding gcd of both terms
    int common_factor = gcd(num3,den3);

    // Converting both terms into simpler
    // terms by dividing them by common factor
    den3 = den3/common_factor;
    num3 = num3/common_factor;
}

//Function to add two fractions
void addFraction(int num1, int den1, int num2,
                 int den2, int &num3, int &den3)
{
    // Finding gcd of den1 and den2
    den3 = gcd(den1,den2);

    // Denominator of final fraction obtained
    // finding LCM of den1 and den2
    // LCM * GCD = a * b
    den3 = (den1*den2) / den3;

    // Changing the fractions to have same denominator
    // Numerator of the final fraction obtained
    num3 = (num1)*(den3/den1) + (num2)*(den3/den2);

    // Calling function to convert final fraction
    // into it's simplest form
    lowest(den3,num3);
}

// Driver program
int main()
{
    int num1=1, den1=500, num2=2, den2=1500, den3, num3;
    addFraction(num1, den1, num2, den2, num3, den3);
    printf("%d/%d + %d/%d is equal to %d/%d\n", num1, den1,
                                    num2, den2, num3, den3);
    return 0;
}
```

JAVA-

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

    // Function to convert the obtained fraction into its simplest form
    public static void lowest(int[] num3Den3) {
        int num3 = num3Den3[0];
        int den3 = num3Den3[1];

        int commonFactor = gcd(num3, den3);

        // Converting both terms into simpler terms
        num3Den3[0] = num3 / commonFactor;
        num3Den3[1] = den3 / commonFactor;
    }

    // Function to add two fractions
    public static void addFraction(int num1, int den1, int num2, int den2, int[] num3Den3) {
        int den3 = gcd(den1, den2);

        // Denominator of final fraction obtained
        // finding LCM of den1 and den2
        // LCM * GCD = a * b
        den3 = (den1 * den2) / den3;

        // Changing the fractions to have same denominator
        // Numerator of the final fraction obtained
        int num3 = (num1 * (den3 / den1)) + (num2 * (den3 / den2));

        // Set result in num3Den3
        num3Den3[0] = num3;
        num3Den3[1] = den3;

        // Calling function to convert final fraction into its simplest form
        lowest(num3Den3);
    }
```

Answer 39-

```cpp
#include <iostream>
#include <math.h>
using namespace std;

// Function to print the divisors
void printDivisors(int n)
{
    int i;
    for (i = 1; i * i < n; i++) {
        if (n % i == 0)
            cout<<i<<" ";
    }
    if (i - (n / i) == 1) {
        i--;
    }
    for (; i >= 1; i--) {
        if (n % i == 0)
            cout<<n / i<<" ";
    }
}

// Driver code
int main()
{
    cout << "The divisors of 100 are: \n";

    printDivisors(100);

    return 0;
}
```

JAVA-

```java
void printDivisors(int n) {
    int i;
    for (i = 1; i * i < n; i++) {
        if (n % i == 0)
            System.out.print(i + " ");
    }
    if (i * i == n) {
        i--;
    }
    for (; i >= 1; i--) {
        if (n % i == 0)
            System.out.print(n / i + " ");
    }
}
```

Answer 40-

```
int addDigits(int num) {
        int ans=0;
        int temp=num;
        while(temp){
            ans+=temp%10;
            temp/=10;
        }
        temp=ans;
        ans=0;
        while(temp/10){
            while(temp){
                ans+=temp%10;
                temp/=10;
            }
            temp=ans;
            ans=0;
        }
        ans=temp;
        return ans;
    }
```

JAVA-

```
    int addDigits(int num) {
    while (num >= 10) {
        int sum = 0;
        while (num > 0) {
            sum += num % 10;
            num /= 10;
        }
        num = sum;
    }
    return num;
}
```

Answer 41-

```
int factorial(int n) {
    // Base case: factorial of 0 or 1 is 1
    if (n == 0 || n == 1) {
        return 1;
    } else {
        // Recursive case: n * factorial of (n-1)
        return n * factorial(n - 1);
    }
}
```

JAVA-

```
int factorial(int n) {
    // Base case: factorial of 0 or 1 is 1
    if (n == 0 || n == 1) {
        return 1;
    } else {
        // Recursive case: n * factorial of (n-1)
        return n * factorial(n - 1);
    }
}

```

Answer 42-

```
int climbStairs(int n) {
        if (n == 0 || n == 1) {
            return 1;
        }
        return climbStairs(n-1) + climbStairs(n-2);
    }
```

JAVA-

```
 int climbStairs(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return climbStairs(n - 1) + climbStairs(n - 2);
}

```

Answer 43-

```
int recursion(int m,int n,int i,int j){

    if(i==m || j==n){
        return 0;
    }

    if(i==m-1 && j==n-1){
        return 1;
    }

   return  recursion(m,n,i+1,j)+recursion(m,n,i,j+1);
}

int uniquePaths(int m, int n) {
   return recursion(m,n,0,0);

}
```

JAVA-

```
int recursion(int m, int n, int i, int j) {
    if (i == m || j == n) {
        return 0;
    }
    if (i == m - 1 && j == n - 1) {
        return 1;
    }
    return recursion(m, n, i + 1, j) + recursion(m, n, i, j + 1);
}

int uniquePaths(int m, int n) {
    return recursion(m, n, 0, 0);
}

```

Answer 44-

```cpp
int solve(vector<int>& coins, int amount){
        if(amount==0) return 0;
        if(amount<0) return INT_MAX;
        int m = INT_MAX;
        for(auto i: coins){
            int res = solve(coins,amount-i);
            if(res!=INT_MAX){
                m = min(m,res+1);
            }
        }
        return m;
    }

    int coinChange(vector<int>& coins, int amount) {
        return solve(coins,amount)==INT_MAX ? -1 : solve(coins,amount);
    }
```

JAVA-

```java
    int solve(int[] coins, int amount) {
    if (amount == 0) return 0;
    if (amount < 0) return Integer.MAX_VALUE;
    int m = Integer.MAX_VALUE;
    for (int i : coins) {
        int res = solve(coins, amount - i);
        if (res != Integer.MAX_VALUE) {
            m = Math.min(m, res + 1);
        }
    }
    return m;
}

int coinChange(int[] coins, int amount) {
    return solve(coins, amount) == Integer.MAX_VALUE ? -1 : solve(coins, amount);
}
```

Answer 45-

```cpp
int rob(vector<int>& arr) {
        int n = arr.size();

        return recursion(arr, n-1);
    }


    int recursion(vector<int> &arr, int i){
        if(i>=arr.size()) return 0;

        int take = arr[i]+recursion(arr, i-2);
        int dontTake = recursion(arr, i-1);

        return max(take, dontTake);
    }
```

JAVA-

```java
    int rob(int[] arr) {
    return recursion(arr, arr.length - 1);
}

int recursion(int[] arr, int i) {
    if (i >= arr.length) return 0;

    int take = arr[i] + recursion(arr, i - 2);
    int dontTake = recursion(arr, i - 1);

    return Math.max(take, dontTake);
}
```

Answer 46-

```java
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap arr[j] and arr[j+1]
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

Answer 47-

```java
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        int minIdx = i;
        for (int j = i+1; j < n; j++) {
            if (arr[j] < arr[minIdx]) {
                minIdx = j;
            }
        }
        // Swap arr[i] and arr[minIdx]
        int temp = arr[minIdx];
        arr[minIdx] = arr[i];
        arr[i] = temp;
    }
}
```

Answer 48 -

```java
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        // Move elements that are greater than key to one position ahead
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
```

Answer 49-

```java
void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    int i = 0, j = 0, k = l;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
```

Answer 50-

```
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            // Swap arr[i] and arr[j]
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    // Swap arr[i+1] and arr[high] (or pivot)
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

BONUS QUESTION –

Answer 1 -

```cpp
ListNode* reverseList(ListNode* head) {
        // Initialize pointers
        ListNode* prev = nullptr; // Previous node starts as NULL
        ListNode* next = nullptr; // Next node
        ListNode* curr = head;    // Current node starts at the head

        // Traverse the list
        while (curr != nullptr) {
            // Save the next node
            next = curr->next;

            // Reverse the link
            curr->next = prev;

            // Move pointers forward
            prev = curr; // Move prev to the current node
            curr = next; // Move curr to the next node
        }

        // prev is now the new head of the reversed list
        return prev;
    }
```

JAVA-

```cpp
ListNode* reverseList(ListNode* head) {
        // Initialize pointers
        ListNode* prev = nullptr; // Previous node starts as NULL
        ListNode* next = nullptr; // Next node
        ListNode* curr = head;    // Current node starts at the head

        // Traverse the list
        while (curr != nullptr) {
            // Save the next node
            next = curr->next;

            // Reverse the link
            curr->next = prev;

            // Move pointers forward
            prev = curr; // Move prev to the current node
            curr = next; // Move curr to the next node
        }

        // prev is now the new head of the reversed list
        return prev;
    }
```

Answer 2 –

```
bool hasCycle(ListNode *head)
    {
        ListNode *slow = head;
        ListNode *fast = head;

        while (fast != NULL and fast->next != NULL)
        {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast)
            {
                return true;
            }
        }
        return false;
    }
```

JAVA-

```
bool hasCycle(ListNode *head)
    {
        ListNode *slow = head;
        ListNode *fast = head;

        while (fast != NULL and fast->next != NULL)
        {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast)
            {
                return true;
            }
        }
        return false;
    }
```

Answer 3-

```
ListNode* middleNode(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

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

        if (fast->next != nullptr) return slow->next;
        else return slow;
    }
```

JAVA-

```
ListNode* middleNode(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

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

        if (fast->next != nullptr) return slow->next;
        else return slow;
    }
```

Answer 4-

```
bool isValid(string s) {
        stack <char> stack;
        for(int i=0;i<s.length();i++)
        {
            if(s[i]=='(' || s[i]=='{' || s[i]=='[')
            {
                stack.push(s[i]);
            }
            else if(s[i]==')' || s[i]=='}' || s[i]==']') //closing bracket
            {
              if(stack.empty())
              {
                  return false;
              }
               else if( ((s[i]==')' && stack.top()=='(') ||
                     (s[i]==']' && stack.top()=='[') ||
                       (s[i]=='}' && stack.top()=='{') )==false)
              {
                   return false;
              }
              else
              {
                  stack.pop();
              }
            }
        }

        return (stack.empty()==true);


    }
```

JAVA-

```
bool isValid(string s) {
        stack <char> stack;
        for(int i=0;i<s.length();i++)
        {
            if(s[i]=='(' || s[i]=='{' || s[i]=='[')
            {
                stack.push(s[i]);
            }
            else if(s[i]==')' || s[i]=='}' || s[i]==']') //closing bracket
            {
              if(stack.empty())
              {
                 return false;
              }
               else if( ((s[i]==')' && stack.top()=='(') ||
                    (s[i]==']' && stack.top()=='[') ||
                       (s[i]=='}' && stack.top()=='{') )==false)
              {
                  return false;
              }
              else
              {
                 stack.pop();
              }
            }
        }

        return (stack.empty()==true);


    }
```

Answer 5 -

```
vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {
        unordered_map<int,int>mp;
        vector<int>v;
        stack<int>s;
        for(int i=nums2.size()-1; i>=0; i--){
            if(s.size()==0){

                mp[nums2[i]]=-1;
            }
            else if(s.size()>0 && s.top()>nums2[i]){

                mp[nums2[i]]=s.top();
            }
            else if(s.size()>0 && s.top()<=nums2[i]){
                while(s.size()>0 && s.top()<=nums2[i]){
                s.pop();
           }
           if(s.size()==0){

                mp[nums2[i]]=-1;
            }
            else{
                mp[nums2[i]]=s.top();
            }
        }
        s.push(nums2[i]);
        }
        for(auto i:nums1){
         v.push_back(mp[i]);
        }
        return v;
    }
```

JAVA-

```java
vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {
        unordered_map<int,int>mp;
        vector<int>v;
        stack<int>s;
        for(int i=nums2.size()-1; i>=0; i--){
            if(s.size()==0){

                mp[nums2[i]]=-1;
            }
            else if(s.size()>0 && s.top()>nums2[i]){

                mp[nums2[i]]=s.top();
            }
            else if(s.size()>0 && s.top()<=nums2[i]){
                while(s.size()>0 && s.top()<=nums2[i]){
                s.pop();
           }
           if(s.size()==0){

                mp[nums2[i]]=-1;
            }
            else{
                  mp[nums2[i]]=s.top();
            }
        }
        s.push(nums2[i]);
        }
        for(auto i:nums1){
         v.push_back(mp[i]);
        }
        return v;
    }
```