Q1. The King's Feast
The King has n plates of food, each with a certain quantity. He wants to know the **maximum food plate**.

- **Input**: n=5, arr=[2,7,1,9,5]
- **Output**: 9
- **Constraints**: $1 \leq n \leq 10^5$, $-10^9 \leq arr[i] \leq 10^9$

Q2. The Lost Soldier
In the battlefield, soldiers are numbered 0…n. One soldier is missing. Find him.

- **Input**: n=5, arr=[0,1,2,4,5]
- **Output**: 3
- **Constraints**: O(n) or O(log n) solution required.

Q3. Potion Mixing (Two Sum)
A wizard wants to mix two potions whose strengths add up to target.

- **Input**: n=4, arr=[3,2,4,7], target=6
- **Output**: Indices (1,2)
- **Constraints**: $1 \leq n \leq 10^5$, $-10^9 \leq arr[i] \leq 10^9$

Q4. The Secret Message
A spy wrote a secret message as numbers. To decode it, reverse the array.

- **Input**: arr=[1,2,3,4]
- **Output**: [4,3,2,1]

Q5. The King's Parade
Soldiers stand in line. Check if their heights are **sorted in non-decreasing order**.

- **Input**: arr=[1,3,5,7] → Output: true
- **Input**: arr=[3,2,1] → Output: false

Q6. The Treasure Island
Each island grid has gold. Find the island row with **maximum gold**.

Input:

3 3
1 2 3
4 5 6
7 8 9

- **Output**: Row 2 (sum=24)

Q7. The Spiral Library
The King built a library where books are kept in spiral shelves. Print them in **spiral order**.

Input:

3 3
1 2 3
4 5 6
7 8 9

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

Q8. The Royal Diagonal
In a royal hall represented as a square, find **sum of both diagonals**.

Input:

3 3
1 2 3
4 5 6
7 8 9

- **Output**: 1+5+9 =15, 3+5+7 = 15

Q9. The Messenger's Path
A messenger wants to go from (0,0) to (n-1,m-1). Cells with 1 are blocked. Can he reach?

Input:

3 3
0 0 0
0 1 0
0 0 0

- **Output**: true

Q10. The Rainwater Pond
Count the number of water ponds in a village (1 = water, 0 = land).

Input:

3 3
1 0 1
0 1 0
1 0 1

- **Output**: 5

Q11. Tower of Temples (Hanoi)
Temples have n golden disks. Move them from source → destination using helper temple. Return moves.

- **Input**: n=3
- **Output**: 7

Q12. The Magical Staircase
A child climbs 1 or 2 steps. Find number of ways to reach step n.

- **Input**: n=4
- **Output**: 5

Q13. The Sorcerer's Spell
Reverse a string using recursion.

- **Input**: abc
- **Output**: cba

Q14. The Dragon's Roar
Print numbers 1 to n using recursion.

- **Input**: n=5
- **Output**: 1 2 3 4 5

Q15. The Hidden Chamber
Find sum of array elements using recursion.

- **Input**: arr=[1,2,3,4]
- **Output**: 10

Q16. The Ancient Scroll
Search for a scroll ID in the archive.

- **Input**: arr=[2,5,7,8], key=7
- **Output**: 2

Q17. The Farmer's Basket
Find if a fruit (number) exists in the basket.

- **Input**: arr=[10,20,30], key=25
- **Output**: -1

Q18. The Secret Door

Doors are numbered in increasing order. Find target door using binary search.

- **Input**: arr=[1,3,5,7,9], key=7
- **Output**: 3

Q19. The Archer's Range

Find the **first occurrence** of an arrow's distance.

- **Input**: arr=[1,2,2,2,3], key=2
- **Output**: 1

Q20. The Treasure Chest

Find the **last occurrence** of a key using binary search.

- **Input**: arr=[1,2,2,2,3], key=2
- **Output**: 3

Q21. The **first index** where the element is **greater than or equal to** the target.

- If element is found → return its first occurrence.
- If not found → return position where it can be inserted.
- If not possible → return n (array size).

Example

Array = [1, 2, 4, 6, 6, 8], target = 6

- Lower bound = index **3** (first 6).

Array = [1, 2, 4, 6, 6, 8], target = 5

- Lower bound = index **3** (as 6 is the first ≥ 5).

Q22. The **first index** where the element is **strictly greater** than the target.

- If all elements ≤ target → return n.

Example

Array = [1, 2, 4, 6, 6, 8], target = 6

- Upper bound = index **5** (first element greater than 6 is 8).

Array = [1, 2, 4, 6, 6, 8], target = 7

- Upper bound = index **5** (8 is first > 7).

Q23. The **smallest element ≥ target** (actual value, not index).

- If no such element exists → return -1.

Example

Array = [1, 2, 4, 6, 6, 8], target = 5

- Ceil = 6.

Array = [1, 2, 4, 6, 6, 8], target = 9

- Ceil = -1 (no element ≥ 9).

Q24. The **largest element ≤ target**.

- If no such element exists → return -1.

Example

Array = [1, 2, 4, 6, 6, 8], target = 5

- Floor = 4.

Array = [1, 2, 4, 6, 6, 8], target = 0

- Floor = -1 (no element ≤ 0).

Q25. The Treasure Map (Linear Search)

A treasure map is represented as a grid n x m. Each cell contains a number.
The King wants to know if the treasure (target) exists on the map.

- **Input**:

n=3, m=3
matrix = [[1,2,3], [4,5,6], [7,8,9]]
target = 5

- **Output**: Yes
- **Constraints**: 1 ≤ n,m ≤ 500, -10^6 ≤ matrix[i][j] ≤ 10^6

Q26. The Magical Scrolls (Linear Search Return Index)

In the royal library, scrolls are arranged in a 2D cabinet of size n x m.
Find the row and column of the scroll with ID = target. If not found, return (-1,-1).

- **Input**:

matrix = [[10,20,30], [40,50,60], [70,80,90]]

target = 60

- **Output**: (1,2)
- **Constraints**: $1 \leq n,m \leq 1000$

Q27. The Battle Formation (Binary Search - Flattened)

Soldiers stand in a grid formation. Their strengths are sorted row-wise **and** the first element of each row is greater than the last of the previous row.
The commander wants to know if a soldier with strength x exists.

- **Input**:

matrix = [[1,3,5], [7,10,11], [16,20,30]]
target = 10

- **Output**: True
- **Constraints**: $1 \leq n,m \leq 300$

Q28. The Queen's Jewels (Binary Search First Occurrence)

The Queen's jewels are stored in a 2D sorted grid. She wants to find the **first position** of a jewel type x.

- **Input**:

matrix = [[1,2,2], [3,4,4], [5,6,7]]
target = 4

- **Output**: (1,1)
- **Constraints**: $1 \leq n,m \leq 1000$

Q29. The Hidden Scrolls (Staircase Search)

The King hides scrolls in a 2D matrix where rows and columns are **sorted**.
Find if a scroll with ID x exists. Use **O(n+m)** method (start from top-right corner).

- **Input**:

matrix = [[1,4,7,11], [2,5,8,12], [3,6,9,16], [10,13,14,17]]
target = 6

- **Output**: True
- **Constraints**: $1 \leq n,m \leq 1000$

Q30. The Magic Portal (Binary Search 2D)

A wizard created portals in a 2D grid sorted in ascending order row-wise and column-wise. To activate a portal, he must find a specific number x.
Return "Activated" if found else "Failed".

- **Input**:

matrix = [[1, 2, 8], [3, 6, 10], [7, 9, 12]]
target = 9

- **Output**: Activated
- **Constraints**: $1 \leq n,m \leq 500$