

## **1. Implement Bubble Sort**

Sort an array using bubble sort and print the array after each pass.

**Hint:**

Compare adjacent elements and swap if needed.

## **2. Implement Selection Sort**

Find the smallest element and place it at the beginning, repeat for all positions.

## **3. Implement Insertion Sort**

Sort an array by inserting elements into the correct position of the already-sorted part.

## **4. Implement Merge Sort**

Divide array into halves, sort recursively, and merge sorted halves.

**Hint:**

This uses **Divide and Conquer**.

## **5. Implement Quick Sort**

Choose a pivot, partition the array, and recursively sort left and right partitions.

**Hint:**

Best case:  $O(n \log n)$ , Worst case:  $O(n^2)$

## 6. Implement Heap Sort

Use a **Min Heap or Max Heap** to sort elements.

**Steps:**

- Build heap
- Extract root repeatedly
- Re-heapify

## 7. Implement Counting Sort

Use frequency counting to sort non-negative integers.

**Hint:**

Efficient for small integer ranges.