**School of Computer Science and Artificial Intelligence**

**Lab Assignment # 12.1**

**Program : B. Tech (CSE)**

**Specialization : CSE**

**Course Title : AI ASSISTED CODING**

**Course Code : 24CS101PC214**

**Semester : III**

**Academic Session : 2025-2026**

**Name of Student : E.KARTHIK PATEL**

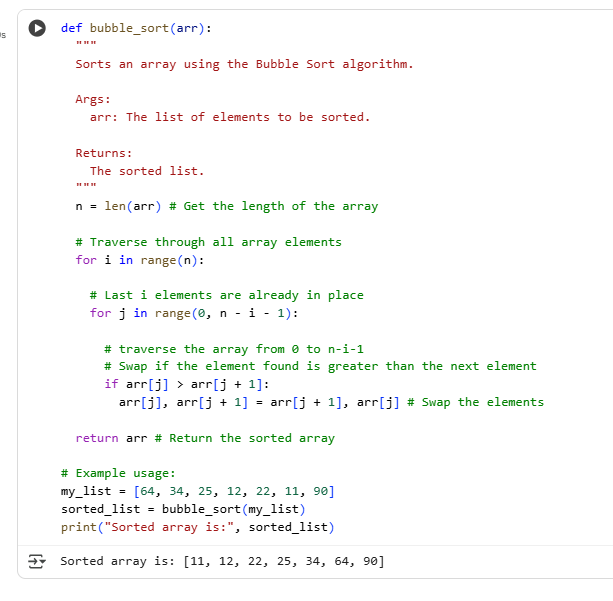
**Enrollment No. : 2403A51416**

**Batch No. : 16**

**Date :08/10/2025**

**TASK1:**

Write a python program Implementing Bubble Sort with AI CommentsCODE **AND OUTPUT:**

****

**EXPLANATION:**

1. The bubble\_sort function takes a list arr as input.
2. It iterates through the list multiple times.
3. In each pass, it compares adjacent elements.
4. If an element is greater than the next, they are swapped.
5. This process "bubbles up" the largest unsorted element to its correct position.
6. The outer loop reduces the range of comparison as elements get sorted.

**Task 2:** write a python program for bubble sort with insertion sort

**CODE AND OUTPUT:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**EXPLANATION:**

The insertion\_sort function sorts a list arr by iterating through it starting from the second element. For each element, it considers it as the key and compares it with the elements before it. If a preceding element is larger than the key, it is shifted one position to the right. This shifting continues until the correct position for the key is found, where all preceding elements are smaller than or equal to it. The key is then inserted into this position. This process is repeated for all elements in the list, gradually building a sorted sublist from the beginning of the array until the entire list is sorted and returned.

**TASK3:**

Write a python program to find difference between Binary Search vs Linear Search.

**CODE AND OUTPUT:**

**Linear Search:**

**A white background with black text

AI-generated content may be incorrect.**

**Binary Search:**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**EXPLANATION:**

Linear Search and Binary Search are two fundamental algorithms used to find elements in a list or array. Linear Search examines each item sequentially, making it easy to implement and effective for unsorted data. However, it can be inefficient for large datasets, as it may scan every element before finding the target or giving up. Binary Search, on the other hand, works only with sorted lists and repeatedly divides the search range in half, leading to much faster results in large arrays. It checks the middle value, then focuses on the left or right sub-list depending on the target's relationship to that value. While Binary Search’s speed is an advantage (with time complexity ), it requires the extra step of sorting data, which can be a drawback in some cases. In summary, Linear Search offers simplicity for small or unsorted lists, while Binary Search provides efficiency for large, sorted arrays.

**TASK4:**

Write a python program to Implement Quick Sort and Merge Sort using recursion

**CODE AND OUTPUT:**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**EXPLANATION:**

Quick Sort is a divide-and-conquer algorithm that works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively. Quick Sort is an in-place sorting algorithm (it doesn't require extra space for sorting) and is generally faster in practice than Merge Sort, especially for smaller datasets or when data is stored in memory. However, its worst-case time complexity is O(n^2), which occurs with a poor choice of pivot.

**TASK5:**

write a python program that AI-Suggested Algorithm Optimization.

**A screenshot of a computer code

AI-generated content may be incorrect.**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**EXPLANATON:**

The naive way to find duplicates checks every pair of elements by comparing each item with all others, which takes a lot of time when the list is large. This is called a brute-force approach and has a time complexity of . A faster way uses a set, which keeps track of items already seen and helps detect duplicates quickly. This optimized method scans the list only once and uses fast lookups inside the set, making its time complexity . The set-based approach is much more efficient for large lists. Both methods return the duplicated elements, but the efficient one is preferred in real applications. Using these algorithms is a great way to understand how choosing the right approach impacts performance.