<table>
<tr><td colspan="3">SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE</td><td colspan="2">DEPARTMENT OF COMPUTER SCIENCE ENGINEERING</td></tr>
<tr><td colspan="2">NAME:ACHINA MANASA
ENROLL NO.:2403A53043
BATCH NO.:24BTCAICYB02
Program Name: B. Tech</td><td colspan="2">Assignment Type: Lab</td><td>Academic Year:2025-2026</td></tr>
<tr><td colspan="2">Course Coordinator Name</td><td colspan="3">Venkataramana Veeramsetty</td></tr>
<tr><td colspan="2">Instructor(s) Name</td><td colspan="3">Dr. V. Venkataramana (Co-ordinator)
Dr. T. Sampath Kumar
Dr. Pramoda Patro
Dr. Brij Kishor Tiwari
Dr.J.Ravichander
Dr. Mohammand Ali Shaik
Dr. Anirodh Kumar
Mr. S.Naresh Kumar
Dr. RAJESH VELPULA
Mr. Kundhan Kumar
Ms. Ch.Rajitha
Mr. M Prakash
Mr. B.Raju
Intern 1 (Dharma teja)
Intern 2 (Sai Prasad)
Intern 3 (Sowmya)
NS_2 (Mounika)</td></tr>
<tr><td>Course Code</td><td>24CS002PC215</td><td>Course Title</td><td colspan="2">AI Assisted Coding</td></tr>
<tr><td>Year/Sem</td><td>II/I</td><td>Regulation</td><td colspan="2">R24</td></tr>
<tr><td>Date and Day of Assignment</td><td>Week6 - Thursday</td><td>Time(s)</td><td colspan="2"></td></tr>
<tr><td>Duration</td><td>2 Hours</td><td>Applicable to Batches</td><td colspan="2"></td></tr>
<tr><td colspan="5">AssignmentNumber:12.1(Present assignment number)/24(Total number of assignments)</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Q.No.	Question	*Expected Time to complete*
1	**Lab 12: Algorithms with AI Assistance – Sorting, Searching, and Optimizing Algorithms** **Lab Objectives:** • Apply AI-assisted programming to implement and optimize sorting and searching algorithms.	Week6 - Thursday

- Compare different algorithms in terms of efficiency and use cases.
- Understand how AI tools can suggest optimized code and complexity improvements.

Task 1: Implementing Bubble Sort with AI Comments

- **Task**: Write a Python implementation of **Bubble Sort**.
- **Instructions**:
  - Students implement Bubble Sort normally.
  - Ask AI to generate **inline comments explaining key logic** (like swapping, passes, and termination).
  - Request AI to provide **time complexity analysis**.
- **Expected Output**:
  - A Bubble Sort implementation with AI-generated explanatory comments and complexity analysis.

Ass1.py > ...

```python
def bubble_sort(arr):
    # Outer loop for each pass through the list
    for i in range(n - 1):
        # Flag to check if any swapping happened in this pass
        swapped = False

        # Inner loop for comparing adjacent elements
        for j in range(0, n - i - 1):
            # Compare current element with the next one
            if arr[j] > arr[j + 1]:
                # Swap if elements are in the wrong order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True  # Mark that a swap occurred

        # If no swaps occurred, the list is already sorted → stop early
        if not swapped:
            break
    return arr


# --- User Input and Output ---
arr = list(map(int, input("Enter numbers separated by spaces: ").split()))
print("Original List:", arr)
sorted_arr = bubble_sort(arr)
print("Sorted List:", sorted_arr)

# --- Time Complexity Analysis ---
print("\nTime Complexity Analysis:")
print("Best Case: O(n) — when the list is already sorted (no swaps).")
print("Average Case: O(n²) — general case with multiple swaps per pass.")
print("Worst Case: O(n²) — when the list is in reverse order (maximum swaps).")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

```
/usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
/dev/fd/13:25: command not found: compdef
/Users/brungisrikar/.zshrc:5: no such file or directory: /opt/homebrew/bin/brew
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
Enter numbers separated by spaces: 5 3 8 4 2
Original List: [5, 3, 8, 4, 2]
Sorted List: [2, 3, 4, 5, 8]

Time Complexity Analysis:
Best Case: O(n) — when the list is already sorted (no swaps).
Average Case: O(n²) — general case with multiple swaps per pass.
Worst Case: O(n²) — when the list is in reverse order (maximum swaps).
brungisrikar@Brungis-MacBook-Pro WTMP %
```

Task 2: Optimizing Bubble Sort → Insertion Sort

- **Task**: Provide Bubble Sort code to AI and ask it to suggest a **more efficient algorithm** for partially sorted arrays.
- **Instructions**:
  - Students implement Bubble Sort first.
  - Ask AI to suggest an alternative (Insertion Sort).
  - Compare performance on nearly sorted input.
- **Expected Output**:
  - Two codes (Bubble Sort + Insertion Sort).
  - AI explanation of why Insertion Sort is more efficient for partially sorted data.

Ass1.py > insertion_sort

```python
def bubble_sort(arr):
    """Bubble Sort algorithm — simple but inefficient for large or partially sorted data."""
    n = len(arr)
    for i in range(n - 1):
        swapped = False
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:  # Optimization: stop if no swaps occurred
            break
    return arr


def insertion_sort(arr):
    """Insertion Sort algorithm — efficient for small or nearly sorted datasets."""
    for i in range(1, len(arr)):
        key = arr[i]            # Current element to insert
        j = i - 1
        # Move elements greater than key one position ahead
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key        # Insert key into correct position
    return arr


# --- User Input ---
arr = list(map(int, input("Enter numbers separated by spaces: ").split()))

# --- Sorting and Comparison ---
print("\nOriginal List:", arr)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

```
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py

Original List: [1, 2, 3, 4, 5]
Sorted by Bubble Sort: [1, 2, 3, 4, 5]
Sorted by Insertion Sort: [1, 2, 3, 4, 5]

Performance Comparison and Explanation:

Bubble Sort:
- Works by repeatedly swapping adjacent elements.
- Requires multiple passes even for nearly sorted arrays.
- Time Complexity: O(n²) in all but best cases.
- Best Case: O(n) if already sorted (with early stop optimization).
```

Task 3: Binary Search vs Linear Search

- **Task**: Implement both **Linear Search** and **Binary Search**.

- **Instructions**:
  - Use AI to generate docstrings and performance notes.
  - Test both algorithms on sorted and unsorted data.
  - Ask AI to explain when Binary Search is preferable.
- **Expected Output**:
  - Two implementations with docstrings.
  - A student observation table comparing performance (Linear vs Binary Search).

Ass1.py > ...

```python
def binary_search(arr, target):

    while low <= high:
        mid = (low + high) // 2  # Find middle index
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1  # Search in right half
        else:
            high = mid - 1  # Search in left half
    return -1


# --- User Input Section ---
arr = list(map(int, input("Enter numbers separated by spaces: ").split()))
target = int(input("Enter number to search: "))

print("\n--- Linear Search on Unsorted Data ---")
lin_result = linear_search(arr, target)
print(f"Result: {'Found at index ' + str(lin_result) if lin_result != -1 else 'Not found'}")

print("\n--- Binary Search on Sorted Data ---")
sorted_arr = sorted(arr)
print(f"Sorted List: {sorted_arr}")
bin_result = binary_search(sorted_arr, target)
print(f"Result: {'Found at index ' + str(bin_result) if bin_result != -1 else 'Not found'}")

# --- AI Explanation and Comparison Table ---
print("""
```

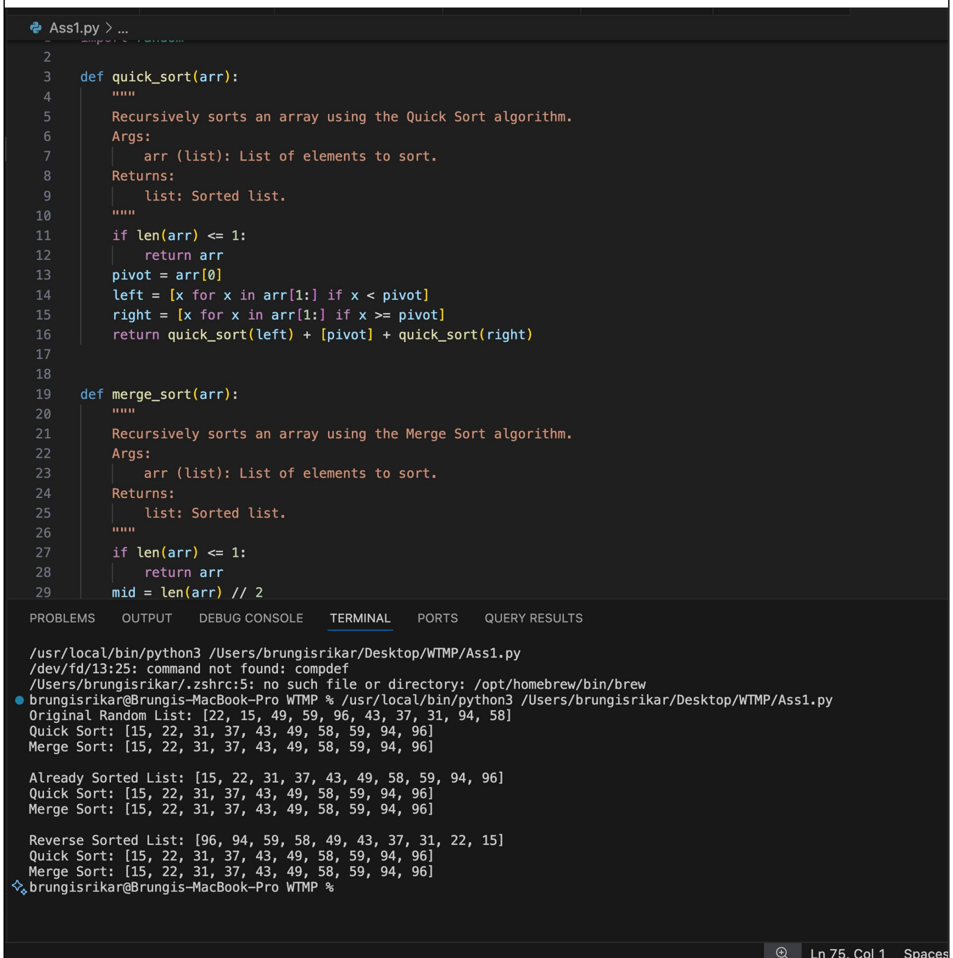
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   QUERY RESULTS

```
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
Enter number to search: 6

--- Linear Search on Unsorted Data ---
Result: Found at index 3

--- Binary Search on Sorted Data ---
Sorted List: [1, 3, 5, 6, 8, 9]
Result: Found at index 3


Performance Comparison Table
+-----------------+-----------------------+---------------------+
|   Search Type   |    Time Complexity    |       Works On      |
+-----------------+-----------------------+---------------------+
| Linear Search   | O(n)                  | Sorted/Unsorted Data |
| Binary Search   | O(log n)              | Sorted Data Only    |
+-----------------+-----------------------+---------------------+
```

Task 4: Quick Sort and Merge Sort Comparison

- **Task:** Implement Quick Sort and Merge Sort using recursion**.**
- **Instructions:**
  - Provide AI with partially completed functions for recursion.
  - Ask AI to complete the missing logic and add docstrings.

- ◦ Compare both algorithms on random, sorted, and reverse-sorted lists.
- **Expected Output:**
    - ◦ Working Quick Sort and Merge Sort implementations.
    - ◦ AI-generated explanation of average, best, and worst-case complexities.



Ass1.py > ...

```python

def quick_sort(arr):
    """
    Recursively sorts an array using the Quick Sort algorithm.
    Args:
        arr (list): List of elements to sort.
    Returns:
        list: Sorted list.
    """
    if len(arr) <= 1:
        return arr
    pivot = arr[0]
    left = [x for x in arr[1:] if x < pivot]
    right = [x for x in arr[1:] if x >= pivot]
    return quick_sort(left) + [pivot] + quick_sort(right)


def merge_sort(arr):
    """
    Recursively sorts an array using the Merge Sort algorithm.
    Args:
        arr (list): List of elements to sort.
    Returns:
        list: Sorted list.
    """
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

```
/usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
/dev/fd/13:25: command not found: compdef
/Users/brungisrikar/.zshrc:5: no such file or directory: /opt/homebrew/bin/brew
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
Original Random List: [22, 15, 49, 59, 96, 43, 37, 31, 94, 58]
Quick Sort: [15, 22, 31, 37, 43, 49, 58, 59, 94, 96]
Merge Sort: [15, 22, 31, 37, 43, 49, 58, 59, 94, 96]

Already Sorted List: [15, 22, 31, 37, 43, 49, 58, 59, 94, 96]
Quick Sort: [15, 22, 31, 37, 43, 49, 58, 59, 94, 96]
Merge Sort: [15, 22, 31, 37, 43, 49, 58, 59, 94, 96]

Reverse Sorted List: [96, 94, 59, 58, 49, 43, 37, 31, 22, 15]
Quick Sort: [15, 22, 31, 37, 43, 49, 58, 59, 94, 96]
Merge Sort: [15, 22, 31, 37, 43, 49, 58, 59, 94, 96]
brungisrikar@Brungis-MacBook-Pro WTMP %
```

Ln 75, Col 1 Spaces

Task 5: AI-Suggested Algorithm Optimization

- **Task:** Give AI a naive algorithm (e.g., O(n²) duplicate search).
- **Instructions:**
    - ◦ Students write a brute force duplicate-finder.
    - ◦ Ask AI to optimize it (e.g., by using sets/dictionaries with O(n) time).
    - ◦ Compare execution times with large input sizes.
- **Expected Output:**
    - ◦ Two versions of the same algorithm (brute force +

optimized).

- AI explanation of how complexity was improved.

Ass1.py > ...

```python
def find_duplicates_optimized(arr):
```

```python
            if item in seen:
                duplicates.add(item)
            else:
                seen.add(item)
        return list(duplicates)


#    Example Input
example_input = [5, 3, 8, 5, 2, 3, 9, 1, 8]

#    Expected Output: [5, 3, 8] (order may vary)
print("Input List:", example_input)
print("Brute Force Duplicates:", find_duplicates_brute(example_input))
print("Optimized Duplicates:", find_duplicates_optimized(example_input))

#    Performance Comparison on Large Input
large_input = [random.randint(0, 10000) for _ in range(10000)]

def time_function(func, arr):
    start = time.time()
    result = func(arr)
    end = time.time()
    return result, end - start

brute_result, brute_time = time_function(find_duplicates_brute, large_input)
opt_result, opt_time = time_function(find_duplicates_optimized, large_input)

print("\n⏱ Performance on Large Input (10,000 elements):")
print(f"Brute Force Time: {brute_time:.4f}s")
print(f"Optimized Time: {opt_time:.4f}s")
print("Same Results:", set(brute_result) == set(opt_result))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

```
/usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
/dev/fd/13:25: command not found: compdef
/Users/brungisrikar/.zshrc:5: no such file or directory: /opt/homebrew/bin/brew
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
Input List: [5, 3, 8, 5, 2, 3, 9, 1, 8]
Brute Force Duplicates: [5, 3, 8]
Optimized Duplicates: [8, 3, 5]

⏱ Performance on Large Input (10,000 elements):
Brute Force Time: 2.3887s
Optimized Time: 0.0018s
Same Results: True
brungisrikar@Brungis-MacBook-Pro WTMP %
```