|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** B. Tech | | | | **Assignment Type: Lab** | | | **Academic Year:**2025-2026 | | |
| **Course Coordinator Name** | | | | Venkataramana Veeramsetty | | | | | |
| **Instructor(s) Name** | | | | |  | | --- | | 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) | | | | | | |
| **Course Code** | | | 24CS002PC215 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week6 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **AssignmentNumber:12.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **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. * Compare different algorithms in terms of efficiency and use cases. * Understand how AI tools can suggest optimized code and complexity improvements.   **Task Description #1 (Sorting – Merge Sort Implementation)**   * Task: Use AI to generate a Python program that implements the Merge Sort algorithm. * Instructions:   + Prompt AI to create a function merge\_sort(arr) that sorts a list in ascending order.   + Ask AI to include time complexity and space complexity in the function docstring.   + Verify the generated code with test cases. * Expected Output:   + A functional Python script implementing Merge Sort with proper documentation.   **Prompt:**  Write a Python program to implement Merge Sort,Create a function merge\_sort(arr) that sorts a list in ascending order.  Include a docstring explaining time and space complexity with user input  **Code:**  IMG_256  IMG_256  **Observations:**   1. The program sorts a list of numbers in **ascending order** using the Merge Sort algorithm. 2. The list is **recursively divided** into halves until single-element lists remain. 3. The merge function **combines two sorted lists** into one sorted list.   **Task Description #2 (Searching – Binary Search with AI Optimization)**   * Task: Use AI to create a binary search function that finds a target element in a sorted list. * Instructions:   + Prompt AI to create a function binary\_search(arr, target) returning the index of the target or -1 if not found.   + Include docstrings explaining best, average, and worst-case complexities.   + Test with various inputs. * Expected Output:   + Python code implementing binary search with AI-generated comments and docstrings.   **Prompt:**  Write a Python function binary\_search(arr, target) that searches for a target element in a sorted list.  Return the index of the target if found, otherwise return -1.  Include docstrings explaining best, average, and worst-case time complexity with user input  **Code:**  IMG_256  **Observations:**   1. The function searches a **sorted list efficiently** by repeatedly dividing the search range in half. 2. Returns the **index** of the target if found, else **-1**. 3. **Best Case:** O(1) (target is in the middle). 4. **Average & Worst Case:** O(log n) (dividing list each step). 5. Works for **numbers, strings, or any comparable elements** in a sorted list. 6. Uses a **while loop** to avoid recursion, making it memory-efficient.   **Task Description #3 (Real-Time Application – Inventory Management System)**   * Scenario: A retail store’s inventory system contains thousands of products, each with attributes like product ID, name, price, and stock quantity. Store staff need to:   1. Quickly search for a product by ID or name.   2. Sort products by price or quantity for stock analysis. * Task:   1. Use AI to suggest the most efficient search and sort algorithms for this use case.   2. Implement the recommended algorithms in Python.   3. Justify the choice based on dataset size, update frequency, and performance requirements. * Expected Output:   1. A table mapping operation → recommended algorithm → justification.   2. Working Python functions for searching and sorting the inventory.   **Prompt:**  A store has thousands of products with ID, name, price, and quantity.  1. Suggest the best way to search by ID or name.  2. Suggest the best way to sort by price or quantity.  3. Write Python functions for searching and sorting with user input.  **Code:**  IMG_256  IMG_256  **Observations:**   1. Searching by **ID** is fastest with **binary search** if the list is sorted, or **hashing/dictionary lookup** for direct access. 2. Searching by **name** works well with **linear search** for small datasets, or **dictionary lookup** if names are unique. 3. Sorting by **price or quantity** is efficient with **Merge Sort** or Python’s built-in **Timsort** (sorted()), which handles large lists quickly. 4. **Binary search** requires a **sorted list**, while dictionary lookups are **O(1)** on average. 5. The chosen algorithms balance **speed and memory use**, suitable for thousands of products with moderate updates.     ✅ Deliverables (For All Tasks)   1. AI-generated prompts for code and test case generation. 2. At least 3 assert test cases for each task. 3. AI-generated initial code and execution screenshots. 4. Analysis of whether code passes all tests. 5. Improved final version with inline comments and explanation. 6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.   Top of Form | | | | | | Week6 - Monday |  |