|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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:12.5**(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 1: Sorting Student Records for Placement Drive**  **Scenario:** SR University is preparing for a campus placement drive. The Training and Placement Cell needs student records sorted by **CGPA** in descending order to easily shortlist candidates.   * Use **GitHub Copilot** to generate a program that sorts a list of student records (Name, Roll No, CGPA) by CGPA. * Implement both **Quick Sort** and **Merge Sort** using AI assistance. * Compare the runtime performance of both algorithms on large datasets. * Write a function that outputs the **top 10 students** with the highest CGPA.   Input:          Output:      **Task 2: Optimized Search in Online Library System**  **Scenario:** SR University’s digital library has thousands of research papers. Students frequently search for a paper by **title or author name**. The current linear search is too slow.   * Use **GitHub Copilot** to implement **Binary Search** and **Hash-based Search** for faster lookups. * Load a dataset of book titles and authors (CSV or JSON file). * Allow the user to input a keyword and return all matching entries. * Compare the efficiency of **linear search vs binary search vs hashing** using test cases.   Input:                Output:    **Task 3: Route Optimization for AUV Swarm**  **Scenario:** A research team at SR University is simulating **Autonomous Underwater Vehicle (AUV) swarms**. Each AUV must visit multiple underwater sensors, and the goal is to minimize travel distance (like the **Traveling Salesman Problem**).   * With **GitHub Copilot**, implement an algorithm to optimize the route:   1. Start with a **Greedy approach**.   2. Improve with **Genetic Algorithm (GA)** or **Simulated Annealing (SA)**. * Use a dataset of sensor coordinates (x, y). * Visualize the optimized route using a plotting library (e.g., Matplotlib). * Compare the optimized solution with a random path in terms of distance travel.   Input:        Output:    **Task 4: Real-Time Stock Data Sorting & Searching**  **Scenario:** An AI-powered **FinTech Lab** at SR University is building a tool for analyzing **stock price movements**. The requirement is to quickly **sort stocks by daily gain/loss** and search for specific stock symbols efficiently.   * Use **GitHub Copilot** to fetch or simulate stock price data (Stock Symbol, Opening Price, Closing Price). * Implement sorting algorithms to rank stocks by **percentage change**. * Implement a **search function** that retrieves stock data instantly when a stock symbol is entered. * Optimize sorting with **Heap Sort** and searching with **Hash Maps**. * Compare performance with standard library functions (sorted(), dict lookups) and analyze trade-offs.   Input:            Output:        Top of Form | | | | | | Week6 - Friday |  |