|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOLOFCOMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENTOFCOMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **AssignmentType: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | 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) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **DateandDay**  **of Assignment** | | | Week6 - WednesDay | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:11.3**(Presentassignmentnumber)/**24**(Totalnumberofassignments) | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | **Lab 11 – Data Structures with AI: Implementing Fundamental Structures**  **Lab Objectives**   * To implement fundamental data structures with the assistance of AI tools. * To understand how AI suggests different implementations and optimizations. * To analyze the readability, correctness, and performance of AI-generated code. * To reinforce problem-solving skills using AI-powered coding assistance.   **Learning Outcomes**  After completing this lab, students will be able to:   1. Implement stack, queue, and linked list using Python with AI support. 2. Use AI tools to optimize and refactor basic data structure operations. 3. Compare multiple AI-suggested implementations for the same structure. 4. Apply AI assistance to generate test cases for verifying data structure behavior. 5. Demonstrate understanding of trade-offs in AI-generated solutions.   **Task Description #1 – Stack class implementation**  Task: Ask AI to implement a stack class with push(), pop(), peek() and is\_empty() methods  Output :    **Explanation :**  The init method initializes an empty list to store the stack elements. The push method adds an element to the end of the list (top of the stack). The pop method removes and returns the last element from the list if the stack is not empty. The peek method returns the last element without removing it, if the stack is not empty. The is\_empty method checks if the list (stack) is empty.  **Task Description #2 – Queue Implementation**  Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is\_empty().  Output:    **Explanation:**  init creates an empty list to hold queue items. enqueue adds an item to the end of the list (rear of the queue). dequeue removes and returns the first item (front of the queue) if the queue isn't empty. is\_empty checks if the list is empty.  **Task Description #3 – Linked List Implementation**  Task: Ask AI to create a singly linked list with insert\_at\_end(), insert\_at\_beginning(), and display().  Output:    Explanation:  The Node class represents an element with data and a next pointer. The SinglyLinkedList class has a head to the first node. insert\_at\_beginning adds a new node at the start. insert\_at\_end adds a new node at the end by traversing the list. display iterates through the list and prints the data of each node.  **Task Description #4 – Binary Search Tree (BST)**  Task: Ask AI to generate a simple BST with insert() and inorder\_traversal().  Output:    Explanation:  This code defines a Binary Search Tree (BST).  The Node class represents a node with a key and pointers to its left and right children. The BST class has a root node. The insert method adds a new node while maintaining the BST property (smaller keys go to the left, larger to the right) using a recursive helper function. The inorder\_traversal method performs an inorder traversal (left, root, right) and prints the keys, resulting in a sorted output. | | | | | | Week5 - Monday |  |