**Name: Akshaya Neerati**

**Enrollment No: 2403A510B8**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **Assignment Type: 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 | | | |
| **Date and Day**  **of Assignment** | | | Week1 - Tuesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | | 24CSBTB01 To 24CSBTB39 | | | |
| **AssignmentNumber:1.2**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 1: Environment Setup – GitHub Copilot and VS Code Integration  **Lab Objectives:**   * To install and configure GitHub Copilot in Visual Studio Code. * To explore AI-assisted code generation using GitHub Copilot. * To analyze the accuracy and effectiveness of Copilot's code suggestions. * To understand prompt-based programming using comments and code context   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Set up GitHub Copilot in VS Code successfully. * Use inline comments and context to generate code with Copilot. * Evaluate AI-generated code for correctness and readability. * Compare code suggestions based on different prompts and programming styles.   **Task Description#1**   * Write a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and allow Copilot to complete it.   **Expected Output#1**   * A function that correctly returns True for phrases like "A man a plan a canal Panama"   **Prompt:**  Write a code to check the string function is a palindrome or not in python and give an example by ignoring spaces and cases.  Screenshot 2025-08-05 102730  **Output:**  **Screenshot 2025-08-05 102741**  **Observation:**  The implemented function correctly ignores spaces and letter case when checking for palindromes. It successfully identifies "A man a plan a canal Panama" as a palindrome, matching the expected output. The logic is clear: it removes spaces, converts all characters to lowercase, and compares the cleaned string with its reverse. This ensures accuracy for phrases containing spaces and mixed case letters.  **Task Description#2**   * Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring   **Expected Output#2**   * AI completes the function logic using loop or recursion with accurate output   **Prompt:**  Python function for fibonacci sequence up to n terms using recursion or loop with accurate output.  Screenshot 2025-08-05 101452  **Output:**  Screenshot 2025-08-05 101509  **Observation:**  The Fibonacci sequence function correctly generates the first n terms using an iterative loop. It initializes the first two terms (0 and 1), appends them sequentially to a list, and updates the values in each iteration. For n = 10, the output [0, 1, 1, 2, 3, 5, 8, 13, 21, 34] matches the expected Fibonacci sequence, confirming accurate logic and execution.  **Task Description#3**   * Write a comment like # Function to reverse a string and use Copilot to generate the function.   **Expected Output#3**   * Auto-completed reverse function   **Prompt:**  A python code for funtion to reverse a string.  Screenshot 2025-08-05 101239  **Output:**  **Screenshot 2025-08-05 101307**  **Observation:**  The reverse string function works as intended by using Python’s slicing syntax [::-1] to reverse the input string. The provided example "hello" correctly returns "olleh", demonstrating the function’s correctness and efficiency. The implementation is concise, leveraging Python's built-in capabilities without requiring additional loops or libraries.  **Task Description#4**   * Generate a program that simulates a basic calculator (add, subtract, multiply, divide). Write the comment: # Simple calculator with 4 operations and let AI complete it.   **Expected Output#4**   * Fully working calculator with input/output and operator selection logic.   **Prompt:**  A python code that simulates a simple calculator with 4 operations- add, subtract, multiply, divide with accurate output as operator logic.  Screenshot 2025-08-05 103422  **Output:**  Screenshot 2025-08-05 103511  **Observation:**  The calculator program is functioning correctly, allowing users to select from four operations- addition, subtraction, multiplication, and division- via menu-based input. It handles invalid menu choices gracefully and includes a check for division by zero. In the test run, selecting subtraction (choice = 2) with inputs 4 and 2 correctly returned 2.0, confirming both accurate computation and user interaction flow.  **Task Description#5**   * Use a comment to instruct AI to write a function that reads a file and returns the number of lines..   **Expected Output#5**   * Functional implementation using open() or with open() and readlines()   **Prompt:**  Function to read a file and return the number of lines using open() and readlines().  Screenshot 2025-08-12 102354    **Output:**  **Screenshot 2025-08-12 102230**  **Observation:**  The function count\_lines\_in\_file(filename) correctly implements the logic to read a file and return the number of lines. It uses the with open() context manager for safe file handling, reads all lines using readlines(), and returns their count with len(). The example call count\_lines\_in\_file('python.txt') follows the expected structure, fulfilling the task requirements accurately.  **Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.**  **Evaluation Criteria:**   | **Criteria** | **Max Marks** | | --- | --- | | Task #1 | 0.5 | | Task #2 | 0.5 | | Task #3 | 0.5 | | Task #4 | 0.5 | | Task #5 | 0.5 | | **Total** | **2.5 Marks** | | | | | | | Week1 - wednesday |  |