**NAME:** B.ARUN KUMAR

**ROLL\_NO:** 2403A510A0

**BATCH\_NO:** 04

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 1:**  🡪The task is that if we give the input to check the condition whether the string is a Palindrome or not if the string is palindrome then print true else false so here if I give “A Man a plan a canal a panama” if the string is palindrome such that the output is true else false.  **Output 1:**  A screenshot of a computer program  AI-generated content may be incorrect.  **Observation 1:**  🡪 Uses a generator expression to filter out non-alphanumeric characters (ignoring punctuation too, not just spaces).   Converts all characters to lowercase for case-insensitive comparison.   Compares the cleaned string to its reverse using slicing.  **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 2:**  🡪write a python code to print the fibonnaci series such that the output will be in list.  **Output 2:**    **Observation 2:**   * Copilot typically begins the Fibonacci sequence with [0, 1] and uses a loop to append subsequent terms until the desired count is reached. It handles edge cases like n = 0 or n = 1 gracefully, often returning an empty list or [0] respectively. The logic is straightforward and avoids unnecessary complexity. This makes the function reliable for a range of input values.   **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 3:**  🡪Write a python code such that the input string is to be reversed and it will display the reversed string.  **Output 3:**  A screenshot of a computer program  AI-generated content may be incorrect.  **Observation 3:**   * Copilot often uses Python slicing ([::-1]) as the most concise and efficient method to reverse a string. If slicing isn't suggested, it may generate a loop-based approach that builds the reversed string character by character. The function generally returns the reversed result directly without unnecessary variables or steps. This makes the output clean, readable, and beginner-friendly.   **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 4:**  **🡪**Write a python code that works as a basic calculator such that the output will ask multiple choices like add , subtract ,multiply, divide and closing calculator.  **Output 4:**    **Observation 4:**  When prompted with the comment # Simple calculator with 4 operations, Copilot typically generates a user-interactive program using input() to collect numbers and the desired operation. It includes conditional logic (if-elif) to handle addition, subtraction, multiplication, and division. The function often checks for division by zero to avoid runtime errors. Overall, the output is a clean, functional calculator script suitable for beginners.  **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 5:**  **🡪** write a python code that reads a file and returns the number of line using dynamic input.  **Output 5:**    **Observation 5:**  When given a comment like # Function to count number of lines in a file, Copilot typically generates a function using Python’s built-in open() or with open() for safe file handling. It reads the file using .readlines() or iterates directly over the file object. The function then returns the count using len() or a loop-based counter. It may also include basic error handling for missing files or incorrect paths.  **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 |  |