

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		<table border="1"> <tr><td>Dr. V. Venkataramana (Co-ordinator)</td></tr> <tr><td>Dr. T. Sampath Kumar</td></tr> <tr><td>Dr. Pramoda Patro</td></tr> <tr><td>Dr. Brij Kishor Tiwari</td></tr> <tr><td>Dr.J.Ravichander</td></tr> <tr><td>Dr. Mohammand Ali Shaik</td></tr> <tr><td>Dr. Anirodh Kumar</td></tr> <tr><td>Mr. S.Naresh Kumar</td></tr> <tr><td>Dr. RAJESH VELPULA</td></tr> <tr><td>Mr. Kundhan Kumar</td></tr> <tr><td>Ms. Ch.Rajitha</td></tr> <tr><td>Mr. M Prakash</td></tr> <tr><td>Mr. B.Raju</td></tr> <tr><td>Intern 1 (Dharma teja)</td></tr> <tr><td>Intern 2 (Sai Prasad)</td></tr> <tr><td>Intern 3 (Sowmya)</td></tr> <tr><td>NS_2 (Mounika)</td></tr> </table>		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)
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	24CS002P C215	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)																				
	Question		Exp ect edT ime to co mpl ete																	

1	<p>Lab 1: Environment Setup – GitHub Copilot and VS Code Integration</p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> • 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 <p>Lab Outcomes (LOs): After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • 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. <p>Task Description#1</p> <ul style="list-style-type: none"> • Write a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and allow Copilot to complete it. <p>Expected Output#1</p> <ul style="list-style-type: none"> • A function that correctly returns True for phrases like "A man a plan a canal Panama" <p>PROMPT : Write a python program for function to check if a string is a valid palindrome (ignoring spaces and case)</p> <pre> import re def is_palindrome(s): """ Checks if a string is a valid palindrome, ignoring spaces and case. Args: s: The input string. Returns: True if the string is a palindrome, False otherwise. """ # Remove spaces and convert to lowercase s = re.sub(r'\s+', '', s).lower() # Check if the string is equal to its reverse return s == s[::-1] # Example usage: string1 = "Race car" string2 = "hello" string3 = "A man a plan a canal Panama" print(f'{string1} is a palindrome: {is_palindrome(string1)}') print(f'{string2} is a palindrome: {is_palindrome(string2)}') print(f'{string3} is a palindrome: {is_palindrome(string3)}') </pre> <pre> 'Race car' is a palindrome: True 'hello' is a palindrome: False 'A man a plan a canal Panama' is a palindrome: True </pre> <p>OBSERVATION : the code efficiently checks for palindromes by cleaning the input string (removing spaces and lowercasing) and then comparing it to its reverse.</p> <p>Task Description#2</p> <ul style="list-style-type: none"> • Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring <p>Expected Output#2</p> <ul style="list-style-type: none"> • AI completes the function logic using loop or recursion with accurate output 	Wee k1 - wed nesd ay
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------

PROMPT : Generate a Python function that returns the Fibonacci sequence up to n terms

```
def fibonacci_sequence(n):  
    """  
    Generates the Fibonacci sequence up to n terms.  
  
    Args:  
        n: The number of terms in the sequence.  
  
    Returns:  
        A list containing the Fibonacci sequence up to n terms.  
    """  
    sequence = []  
    a, b = 0, 1  
    for _ in range(n):  
        sequence.append(a)  
        a, b = b, a + b  
    return sequence  
  
# Example usage:  
n_terms = 10  
fib_sequence = fibonacci_sequence(n_terms)  
print(f"Fibonacci sequence up to {n_terms} terms: {fib_sequence}")
```

⇒ Fibonacci sequence up to 10 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

OBSERVATION :

The sequence starts with 0 and 1.

Each subsequent number is the sum of the two preceding ones (e.g., $1+1=2$, $1+2=3$, $2+3=5$, and so on).

The numbers in the sequence grow relatively quickly.

The ratio of consecutive Fibonacci numbers approaches the golden ratio (approximately 1.618) as the sequence progresses (e.g., $34/21 \approx 1.619$).

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 : Function to reverse a string

```
[1] def reverse_string(s):  
    """Reverses a given string."""  
    return s[::-1]  
  
# Example usage:  
my_string = "hello"  
reversed_string = reverse_string(my_string)  
print(f"Original string: {my_string}")  
print(f"Reversed string: {reversed_string}")
```

⇒ Original string: hello
Reversed string: olleh

OBSERVATION : The output shows that the original string "hello" has been successfully reversed to "olleh" using the `reverse_string` function. This demonstrates that the function is working as

intended.

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 : Write a program that simulates a basic calculator (add, subtract, multiply, divide). And simple calculation with 4 operations

```
def calculator(num1, num2, operation):
    """Simulates a basic calculator with add, subtract, multiply, and divide operations.

    Args:
        num1: The first number.
        num2: The second number.
        operation: The operation to perform ('add', 'subtract', 'multiply', 'divide').

    Returns:
        The result of the operation, or an error message for invalid operations or division by zero.
    """
    if operation == 'add':
        return num1 + num2
    elif operation == 'subtract':
        return num1 - num2
    elif operation == 'multiply':
        return num1 * num2
    elif operation == 'divide':
        if num2 != 0:
            return num1 / num2
        else:
            return "Error: Division by zero is not allowed."
    else:
        return "Error: Invalid operation."

# Example usage:
print(calculator(10, 5, 'add'))
print(calculator(10, 5, 'subtract'))
print(calculator(10, 5, 'multiply'))
print(calculator(10, 5, 'divide'))
print(calculator(10, 0, 'divide'))
print(calculator(10, 5, 'power'))
```

15
5
50
2.0
Error: Division by zero is not allowed.
Error: Invalid operation.

OBSERVATION : Based on the output of the calculator function:

Adding 10 and 5 results in 15.

Subtracting 5 from 10 results in 5.

Multiplying 10 and 5 results in 50.

Dividing 10 by 5 results in 2.0.

Dividing 10 by 0 results in an error message: "Error: Division by zero is not allowed."

Using an invalid operation ('power') results in an error message: "Error: Invalid operation."

Task Description#5

- Use a comment to 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 : Write a function that reads a file and returns number of lines

```
def count_lines_in_file(filename):  
    """  
    Reads a file and returns the number of lines.  
  
    Args:  
        filename: The path to the file.  
  
    Returns:  
        The number of lines in the file.  
    """  
    with open(filename, 'r') as f:  
        line_count = sum(1 for line in f)  
    return line_count  
  
# Example usage:  
# Create a dummy file for demonstration  
with open("my_test_file.txt", "w") as f:  
    f.write("This is line 1.\n")  
    f.write("This is line 2.\n")  
    f.write("This is line 3.\n")  
  
# Get the line count  
file_path = "my_test_file.txt"  
num_lines = count_lines_in_file(file_path)  
print(f"The file '{file_path}' has {num_lines} lines.")  
  
The file 'my_test_file.txt' has 3 lines.
```

OBSERVATION : The code defines a Python function called `count_lines_in_file` that takes a filename as input and returns the number of lines in that file. It then creates a dummy file named `my_test_file.txt` with three lines. Finally, it calls the function with this dummy file and prints the result, which correctly shows that the file has 3 lines. The observation is that the function works as expected for the given test case.

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