Teljeeru Akhil
Roll_no: 2303A53024

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		Dr. Rishabh Mittal	
Instructor(s) Name		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Rapelly Nandini Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju	
CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week1 - Wednesday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
Assignment Number:1.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	**Question**	***Expected Time to complete***
1	Lab 1: Environment Setup – *GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow*	Week1 - Monday

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 0

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

Expected Output

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

Task 1: AI-Generated Logic Without Modularization (Fibonacci Sequence Without Functions)

❖ **Scenario**
You are asked to write a quick numerical sequence generator for a learning platform prototype.

❖ **Task Description**
Use GitHub Copilot to generate a Python program that:

➢ Prints the Fibonacci sequence up to *n* terms
➢ Accepts user input for *n*

- Implements the logic directly in the main code
- Does not use any user-defined functions

❖ **Expected Output**

- Correct Fibonacci sequence for given *n*
- Screenshot(s) showing Copilot-generated suggestions
- Sample inputs and outputs

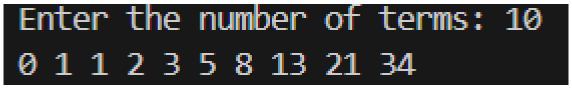
Solution:

Write a python program that takes an integer n as input prints the Fibonacci sequence upto to n terms.

#code

```
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b
n = int(input("Enter the number of terms: "))
fibonacci(n)
```

#Output



```
Enter the number of terms: 10
0 1 1 2 3 5 8 13 21 34
```

Task 2: AI Code Optimization & Cleanup (Improving Efficiency)

❖ **Scenario**

The prototype will be shared with other developers and needs optimization.

❖ **Task Description**

- Examine the Copilot-generated code from Task 1 and improve it by:
- Removing redundant variables
- Simplifying loop logic
- Avoiding unnecessary computations
- Use Copilot prompts such as:
  - *“Optimize this Fibonacci code”*
  - *“Simplify variable usage”*

Hint:

Prompt Copilot with phrases like
“optimize this code”, *“simplify logic”*, or *“make it more readable”*

❖ **Expected Output**

- ➢ Original vs improved code
- ➢ Written explanation of:
    - ▪ What was inefficient
    - ▪ How the optimized version improves performance and readability

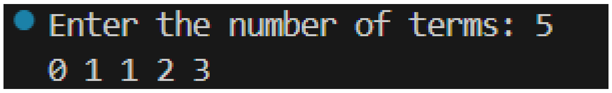
Solution:

Write an optimized Python program that takes an integer n as input and prints the Fibonacci sequence up to n terms by simplifying the logic and removing unnecessary variables, without using any user-defined functions.

#Code

```
n = int(input("Enter the number of terms: "))
a, b = 0, 1
for _ in range(n):
    print(a, end=' ')
    a, b = b, a + b
```

#Output



```
Enter the number of terms: 5
0 1 1 2 3
```

Task 3: Modular Design Using AI Assistance (Fibonacci Using Functions)

❖ **Scenario**

The Fibonacci logic is now required in multiple modules of an application.

❖ **Task Description**

Use GitHub Copilot to generate a function-based Python program that:

- ➢ Uses a user-defined function to generate Fibonacci numbers
- ➢ Returns or prints the sequence up to *n*
- ➢ Includes meaningful comments (AI-assisted)

❖ **Expected Output**

- ➢ Correct function-based Fibonacci implementation
- ➢ Screenshots documenting Copilot’s function generation
- ➢ Sample test cases with outputs

Solution:

Write a Python program that uses a user-defined function to generate and print the Fibonacci sequence up to n terms, accepts user input for n, and includes clear, meaningful comments.

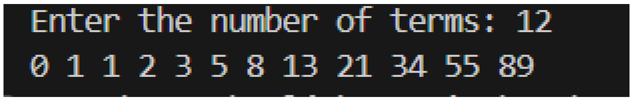
```
#Code

# Function to print Fibonacci series
def fibonacci(n):
   a, b = 0, 1  # First two numbers

   for _ in range(n):   # Loop n times
      print(a, end=' ')
      a, b = b, a + b  # Update values

n = int(input("Enter the number of terms: "))  # User input
fibonacci(n)  # Function call

#Output
```



```
Enter the number of terms: 12
0 1 1 2 3 5 8 13 21 34 55 89
```

Task 4: Comparative Analysis – Procedural vs Modular Fibonacci Code

- **Scenario**
  You are participating in a code review session.

- **Task Description**
  Compare the Copilot-generated Fibonacci programs:
  - Without functions (Task 1)
  - With functions (Task 3)
  - Analyze them in terms of:
    - Code clarity
    - Reusability
    - Debugging ease
    - Suitability for larger systems

- **Expected Output**
  Comparison table or short analytical report

  Solution:

Write a Python program to generate the Fibonacci series for n terms, once without using functions and once using functions.

#Code

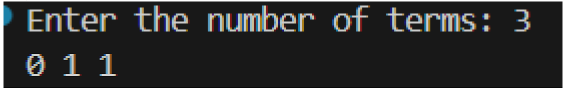
Without using functions

```
n = int(input("Enter the number of terms: "))
a, b = 0, 1
for _ in range(n):
   print(a, end=' ')
   a, b = b, a + b
print()
```

Using functions

```
def fibonacci(n):
   a, b = 0, 1
   for _ in range(n):
      print(a, end=' ')
      a, b = b, a + b
```

#Output



```
Enter the number of terms: 3
0 1 1
```

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches for Fibonacci Series)

❖ **Scenario**

Your mentor wants to assess AI’s understanding of different algorithmic paradigms.

❖ **Task Description**

Prompt GitHub Copilot to generate:

An iterative Fibonacci implementation

A recursive Fibonacci implementation

❖ **Expected Output**

➢ Two correct implementations

➢ Explanation of execution flow for both

- Comparison covering:
    - Time and space complexity
    - Performance for large *n*
    - When recursion should be avoided

Solution:

Write Python programs for the Fibonacci series using iterative and recursive methods, explain how each works, and compare them in terms of time, space, performance for large n, and when recursion should be avoided

#Code

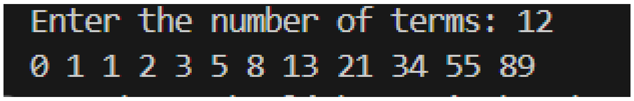
```
# Iterative method

def fibonacci_iterative(n):
  a, b = 0, 1
  for _ in range(n):
    print(a, end=' ')
    a, b = b, a + b
  print()


#Recursive method

def fibonacci_recursive(n, a=0, b=1):
  if n > 0:
    print(a, end=' ')
    fibonacci_recursive(n - 1, b, a + b)
  else:
    print()
```

#Output



```
Enter the number of terms: 12
0 1 1 2 3 5 8 13 21 34 55 89
```