AI Assisted Coding

Assignment - 1

B. Sarayu || 2303A51842|| Batch:- 8

Task 1: AI-Generated Logic Without Modularizaton (Fibonacci Sequence Without Functons) **Code:**

```
]: n = int(input("Enter number of terms: "))

a = 0
b = 1

print("Fibonacci sequence:")

for i in range(n):
    print(a, end=" ")
    c = a + b
    a = b
    b = c
```

```
Enter number of terms: 2
Fibonacci sequence:
0 1
```

Task 2: AI Code Optmizaton & Cleanup (Improving Efciency) **Code**:

```
n = int(input("Enter number of terms: "))

prev, curr = 0, 1

print("Fibonacci sequence:")

for _ in range(n):
    print(prev, end=" ")
    prev, curr = curr, prev + curr
```

```
Enter number of terms: 2
Fibonacci sequence:
0 1
```

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

```
def fibonacci(n):
    """
    Generates Fibonacci numbers up to n terms.
    """

    prev, curr = 0, 1

    for _ in range(n):
        print(prev, end=" ")
        prev, curr = curr, prev + curr


n = int(input("Enter number of terms: "))
fibonacci(n)
```

```
Enter number of terms: 2
0 1
```

Task 4: Comparative Analysis – Procedural vs Modular Fibonacci Code **Code:**

```
]:  def fib_iter(n):
        prev, curr = 0, 1

        for _ in range(n):
            print(prev, end=" ")
            prev, curr = curr, prev + curr
```

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

```
:  def fib_rec(n):
       if n <= 1:
           return n
       return fib_rec(n - 1) + fib_rec(n - 2)


   n = int(input("Enter number of terms: "))

   for i in range(n):
       print(fib_rec(i), end=" ")
```

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