

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		1. Dr. Mohammed Ali Shaik 2. Dr. T Sampath Kumar 3. Mr. S Naresh Kumar 4. Dr. V. Rajesh 5. Dr. Brij Kishore 6. Dr Pramoda Patro 7. Dr. Venkataramana 8. Dr. Ravi Chander 9. Dr. Jagjeeth Singh	
CourseCode	24CS002PC215	CourseTitle	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment		Time(s)	
Duration	2 Hours	Applicableto Batches	
AssignmentNumber: 3.3(Present assignment number)/24(Total number of assignments)			

Q.No.	Question	Expected Time to complete
1	<p>Lab 3: Prompt Engineering – Improving Prompts and Context Management</p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> To understand how prompt structure and wording influence AI-generated code. To explore how context (like comments and function names) helps AI generate relevant output. To evaluate the quality and accuracy of code based on prompt clarity. To develop effective prompting strategies for AI-assisted programming. <p>Lab Outcomes (LOs): After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> Generate Python code using Google Gemini in Google Colab. Analyze the effectiveness of code explanations and suggestions by Gemini. Set up and use Cursor AI for AI-powered coding assistance. Evaluate and refactor code using Cursor AI features. Compare AI tool behavior and code quality across different platforms. 	03.08.2025 EOD

Task Description#1

- Try 3 different prompts to generate a factorial function.

Expected Output#1

- Comparison of AI-generated code styles
- Prompt : write a python code a factorial function

```
def factorial(n):
    """
    Calculate the factorial of a non-negative integer n
    Factorial: n! = n * (n-1) * (n-2) * ... * 2 * 1

    Args:
        n (int): Non-negative integer

    Returns:
        int: Factorial of n

    Raises:
        ValueError: If n is negative
    """
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1

    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

def factorial_recursive(n):
    """
    Recursive implementation of factorial function

    Args:
        n (int): Non-negative integer

    Returns:
        int: Factorial of n
    """
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1

    return n * factorial_recursive(n - 1)

def factorial_math(n):
    """
    Using Python's built-in math.factorial() function
    """
```

[Review next file >](#)

```

Args:
|   n (int): Non-negative integer

Returns:
|   int: Factorial of n
"""
import math
if n < 0:
|   raise ValueError("Factorial is not defined for negative numbers")
return math.factorial(n)

# Test the factorial function
if __name__ == "__main__":
    # Test cases
    test_numbers = [0, 1, 5, 10]

    print("FACTORIAL FUNCTION TESTING")
    print("=" * 40)

    for num in test_numbers:
        try:
            result = factorial(num)
            print(f"{num}! = {result}")
        except Exception as e:
            print(f"Error calculating {num}!: {e}")

    print("\n" + "=" * 40)
    print("INTERACTIVE FACTORIAL CALCULATOR")
    print("=" * 40)

    # Interactive input
    try:
        user_input = int(input("Enter a number to calculate factorial: "))
        if user_input < 0:
            print("Please enter a non-negative number.")
        else:
            result = factorial(user_input)
            print(f"{user_input}! = {result}")
    except ValueError:
        print("Please enter a valid integer.")
    except Exception as e:
        print(f"An error occurred: {e}")

```

[Review next file >](#)

```

0! = 1
1! = 1
5! = 120
10! = 3628800

```

```

=====
INTERACTIVE FACTORIAL CALCULATOR
=====

```

```

Enter a number to calculate factorial: 8
8! = 40320

```

PROMPT 2: Create a Python function that finds the factorial of a number using recursion.

```
1  def factorial(n):
2      """
3      Calculate the factorial of a given non-negative integer using a loop.
4
5      Args:
6          n (int): The number to calculate the factorial for.
7
8      Returns:
9          int: The factorial of n.
10
11     Raises:
12         ValueError: If n is negative.
13     """
14     if n < 0:
15         raise ValueError("Factorial is not defined for negative numbers.")
16     result = 1
17     for i in range(2, n + 1):
18         result *= i
19     return result
20
21 if __name__ == "__main__":
22     try:
23         num = int(input("Enter a non-negative integer: "))
24         print(f"Factorial of {num} is {factorial(num)}")
25     except Exception as e:
26         print("Error:", e)
```

```
Enter a non-negative integer: 5
Factorial of 5 is 120
```

PROMPT 3 : Write a python program to compute factorial of a given number.

```
def factorial(n):| TAB to jump here
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

if __name__ == "__main__":
    try:
        num = int(input("Enter a number to compute its factorial: "))
        print(f"Factorial of {num} is {factorial(num)}")
    except ValueError as e:
        print(f"Error: {e}")
```

```
Enter a number to compute its factorial: 6
Factorial of 6 is 720
```

Task Description#2

- Provide a clear example input-output prompt to generate a sorting function.

Expected Output#2

- Functional sorting code from AI

Prompt : generate a python code to clear example input output to generate a sorting function

```
def bubble_sort(arr):
    """
    Sorts a list of numbers in ascending order using the bubble sort algorithm.

    Args:
        arr (list): List of numbers to sort.

    Returns:
        list: New sorted list in ascending order.

    Example:
        >>> bubble_sort([5, 2, 9, 1])
        [1, 2, 5, 9]
        >>> bubble_sort([])
        []
        >>> bubble_sort([3])
        [3]
    """
    n = len(arr)
    result = arr.copy()
    for i in range(n):
        for j in range(0, n - i - 1):
            if result[j] > result[j + 1]:
                result[j], result[j + 1] = result[j + 1], result[j]
    return result

# Example input/output demonstration
if __name__ == "__main__":
    test_cases = [
        [5, 2, 9, 1],
        [],
        [3],
        [4, 4, 2, 2],
        [10, 9, 8, 7]
    ]
    print("Bubble Sort Example Input/Output:")
    for arr in test_cases:
        print(f"Input: {arr} -> Output: {bubble_sort(arr)}")
```

```
Bubble Sort Example Input/Output:
Input: [5, 2, 9, 1] -> Output: [1, 2, 5, 9]
Input: [] -> Output: []
Input: [3] -> Output: [3]
Input: [4, 4, 2, 2] -> Output: [2, 2, 4, 4]
Input: [10, 9, 8, 7] -> Output: [7, 8, 9, 10]
```

Task Description#3

- Start with the vague prompt “Generate python code to calculate power bill” and improve it step-by-step

Expected Output#3

- Enhanced AI output with clearer prompts
-

Prompt : give a python code to calculate power bill” and improve it step-by-step

```
def calculate_power_bill(units):  
    """  
    Calculate the power bill for a given number of units at a fixed rate.  
    Args:  
        units (float): Number of electricity units consumed.  
    Returns:  
        float: Total bill amount.  
    """  
    rate_per_unit = 5.0 # Example fixed rate per unit  
    return units * rate_per_unit  
  
# Step 2: Add input validation  
def calculate_power_bill_v2(units):  
    """  
    Calculate the power bill with input validation.  
    Args:  
        units (float): Number of electricity units consumed.  
    Returns:  
        float: Total bill amount.  
    Raises:  
        ValueError: If units is negative.  
    """  
    if units < 0:  
        raise ValueError("Units consumed cannot be negative.")  
    rate_per_unit = 5.0  
    return units * rate_per_unit  
  
# Step 3: Add slab-wise billing (tiered rates)  
def calculate_power_bill_v3(units):  
    """  
    Calculate the power bill using slab rates.  
    Args:  
        units (float): Number of electricity units consumed.  
    Returns:  
        float: Total bill amount.  
    """  
    if units < 0:  
        raise ValueError("Units consumed cannot be negative.")  
    bill = 0  
    if units <= 100:  
        bill = units * 3.0  
    elif units <= 200:  
        bill = 100 * 3.0 + (units - 100) * 5.0  
    else:
```

[Review next file >](#)

```

        bill = 100 * 3.0 + 100 * 5.0 + (units - 200) * 8.0
    return bill

# Step 4: Add taxes and fixed charges
def calculate_power_bill_v4(units):
    """
    Calculate the power bill with slab rates, fixed charges, and tax.
    Args:
        units (float): Number of electricity units consumed.
    Returns:
        float: Total bill amount including fixed charges and tax.
    """
    if units < 0:
        raise ValueError("Units consumed cannot be negative.")
    fixed_charge = 50.0
    tax_rate = 0.10 # 10% tax
    # Slab calculation
    if units <= 100:
        bill = units * 3.0
    elif units <= 200:
        bill = 100 * 3.0 + (units - 100) * 5.0
    else:
        bill = 100 * 3.0 + 100 * 5.0 + (units - 200) * 8.0
    bill += fixed_charge
    bill += bill * tax_rate
    return bill

# Step 5: Interactive usage
if __name__ == "__main__":
    try:
        units = float(input("Enter the number of electricity units consumed: "))
        print("Simple Bill (fixed rate):", calculate_power_bill(units))
        print("Validated Bill (fixed rate):", calculate_power_bill_v2(units))
        print("Slab-wise Bill:", calculate_power_bill_v3(units))
        print("Final Bill (with fixed charges and tax):", calculate_power_bill_v4(units))
    except Exception as e:
        print("Error:", e)

```

Enter the number of electricity units consumed: 89
Simple Bill (fixed rate): 445.0
Validated Bill (fixed rate): 445.0
Slab-wise Bill: 267.0
Final Bill (with fixed charges and tax): 348.7

Task Description#4

- Write structured comments to help AI generate two linked functions (e.g., login_user() and register_user()).

Expected Output#4

- Consistent functions with shared logic
- Prompt : give a python code two linked functions (e.g., login_user() and register_user()).

```

def register_user(users_db, username, password):
    """
    Registers a new user with a username and password.

    Args:
        users_db (dict): Dictionary storing user credentials.
        username (str): Desired username.
        password (str): Desired password.

    Returns:
        bool: True if registration is successful, False if username already exists.
    """
    if username in users_db:
        print("Username already exists. Please choose a different username.")
        return False
    users_db[username] = password
    print(f"User '{username}' registered successfully.")
    return True


def login_user(users_db, username, password):
    """
    Logs in a user by checking username and password.

    Args:
        users_db (dict): Dictionary storing user credentials.
        username (str): Username to login.
        password (str): Password to check.

    Returns:
        bool: True if login is successful, False otherwise.
    """
    if username not in users_db:
        print("Username does not exist.")
        return False
    if users_db[username] != password:
        print("Incorrect password.")
        return False
    print(f"User '{username}' logged in successfully.")
    return True


# Example usage
if __name__ == "__main__":
    users = {}
    # Register users
    register_user(users, "alice", "password123")
    register_user(users, "bob", "securepass")
    # Attempt duplicate registration
    register_user(users, "alice", "newpass")
    # Login attempts
    login_user(users, "alice", "password123")
    login_user(users, "bob", "wrongpass")
    login_user(users, "charlie", "nopass")

```

```

Registering user 'alice': True
Registering user 'bob': True
Registering user 'alice' again: False
Login 'alice' with correct password: True
Login 'alice' with wrong password: False
Login 'charlie' (not registered): False

```


Task Description#5

- Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

Expected Output#5

- Code quality difference analysis for various prompts
- Prompt : give a python code Improving Temperature Conversion Function with Clear Instructions

```
def convert_temperature(value, from_unit, to_unit):  
    """  
    Convert temperature between Celsius, Fahrenheit, and Kelvin.  
  
    Args:  
        value (float): The temperature value to convert.  
        from_unit (str): The unit of the input temperature ('C', 'F', or 'K')  
        to_unit (str): The unit to convert to ('C', 'F', or 'K').  
  
    Returns:  
        float: The converted temperature value.  
  
    Raises:  
        ValueError: If an invalid unit is provided.  
    """  
    from_unit = from_unit.upper()  
    to_unit = to_unit.upper()  
    if from_unit not in ['C', 'F', 'K'] or to_unit not in ['C', 'F', 'K']:  
        raise ValueError("Units must be 'C', 'F', or 'K'.")  
  
    # Convert input to Celsius first  
    if from_unit == 'C':  
        celsius = value  
    elif from_unit == 'F':  
        celsius = (value - 32) * 5/9  
    elif from_unit == 'K':  
        celsius = value - 273.15  
  
    # Convert from Celsius to target unit  
    if to_unit == 'C':  
        return celsius  
    elif to_unit == 'F':  
        return celsius * 9/5 + 32  
    elif to_unit == 'K':  
        return celsius + 273.15  
  
if __name__ == "__main__":  
    print("Temperature Converter")  
    print("Supported units: C (Celsius), F (Fahrenheit), K (Kelvin)")  
    try:  
        value = float(input("Enter the temperature value: "))  
        from_unit = input("Enter the unit to convert from (C/F/K): ").strip()  
        to_unit = input("Enter the unit to convert to (C/F/K): ").strip()  
        result = convert_temperature(value, from_unit, to_unit)  
        print(f"{value} {from_unit.upper()} is {result:.2f} {to_unit.upper()}")  
    except Exception as e:  
        print("Error:", e)
```

Temperature Converter

Supported units: C (Celsius), F (Fahrenheit), K (Kelvin)

Enter the temperature value: 30

Enter the unit to convert from (C/F/K): c

Enter the unit to convert to (C/F/K): f

30.0 C is 86.00 F

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
Factorial Function (Task#1)	0.5
Sorting Function (Task#2)	0.5
Vogue Vs. Specific Prompting (Task #3)	0.5
Linked Functions (Task #4)	0.5
Temperature Conversion Function (Task #5)	0.5
Total	2.5 Marks