Name:Chaithra Alluri id:2403A51105 batch:06

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	Lab 3: Prompt Engineering – Improving Prompts and Context Management **Lab Objectives:** • 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. **Lab Outcomes (LOs):** After completing this lab, students will be able to: • 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 program to calculate the factorial of a number using functions

Code:

```
def factorial(n):
    if n < 0:
        return None  # Factorial is not defined for negative numbers
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

# Input from user
num = int(input("Enter a number to find its factorial: "))

fact = factorial(num)
if fact is None:
    print("Factorial is not defined for negative numbers.")
else:
    print(f"The factorial of {num} is: {fact}")
```

Output:

```
Enter a number to find its factorial: 5
The factorial of 5 is: 120
```

Prompt:

Write a recursive function in Python to find the factorial of a number.

Code:

```python
def factorial(n):
    if n < 0:
        return None  # Factorial is not defined for negative numbers
    elif n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Input from user
num = int(input("Enter a number to find its factorial: "))

fact = factorial(num)
if fact is None:
    print("Factorial is not defined for negative numbers.")
else:
    print(f"The factorial of {num} is: {fact}")
```

Output:

```
Enter a number to find its factorial: 6
The factorial of 6 is: 720
```

Prompt:

Write a Python function to calculate factorial, and handle edge cases like 0, negative numbers, and non-integers.

Code:

```python
def factorial(n):
    if not isinstance(n, int):
        raise TypeError("Input must be an integer.")
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

# Input from user
try:
    num = float(input("Enter a number to find its factorial: "))
    if not num.is_integer():
        raise ValueError("Input must be an integer.")
    num = int(num)
    fact = factorial(num)
    print(f"The factorial of {num} is: {fact}")
except (ValueError, TypeError) as e:
    print(f"Error: {e}")
```

Output:

```
Enter a number to find its factorial: 0
The factorial of 0 is: 1
Enter a number to find its factorial: -3
Error: Factorial is not defined for negative numbers.
```

Task Description#2

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

Expected Output#2

- Functional sorting code from AI

Prompt:

```
create a python code input-output to generate a sorting function
```

Code:

```python
def sort_numbers(numbers):
    """
    Sorts a list of numbers in ascending order.
    """
    return sorted(numbers)

# Input from user
try:
    user_input = input("Enter numbers separated by spaces: ")
    num_list = [float(num) for num in user_input.strip().split()]
    sorted_list = sort_numbers(num_list)
    print("Sorted numbers:", sorted_list)
except ValueError:
    print("Error: Please enter valid numbers separated by spaces.")
```

Output:

```
Enter numbers separated by spaces: 1 2 4 5 6 3
Sorted numbers: [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
```

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:

```
Generate python code to calculate power bill
```

Code:

```python
def calculate_power_bill(units):
    """
    Calculates the power bill based on units consumed.
    Example slab:
    - First 100 units: Rs. 1.5/unit
    - Next 100 units (101-200): Rs. 2.5/unit
    - Above 200 units: Rs. 4/unit
    """
    if units < 0:
        raise ValueError("Units cannot be negative.")
    bill = 0
    if units <= 100:
        bill = units * 1.5
    elif units <= 200:
        bill = 100 * 1.5 + (units - 100) * 2.5
    else:
        bill = 100 * 1.5 + 100 * 2.5 + (units - 200) * 4
    return bill

# Input from user
try:
    units = float(input("Enter the number of units consumed: "))
    total_bill = calculate_power_bill(units)
    print(f"Total power bill for {units} units is: Rs. {total_bill:.2f}")
except ValueError as e:
    print(f"Error: {e}")
```

Output:

```
Enter the number of units consumed: -50
Error: Units cannot be negative.
```

```
Enter the number of units consumed: 60
Total power bill for 60.0 units is: Rs. 90.00
```

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:

```
write a python code that will linked two functions (e.g.,
login_user() and register_user()) and get output
```

Code:

```python
users_db = {}

def register_user():
    username = input("Enter a username to register: ")
    if username in users_db:
        print("Username already exists. Try logging in.")
        return False
    password = input("Enter a password: ")
    users_db[username] = password
    print("Registration successful!")
    return True

def login_user():
    username = input("Enter your username: ")
    password = input("Enter your password: ")
    if users_db.get(username) == password:
        print("Login successful!")
        return True
    else:
        print("Invalid username or password.")
        return False

def main():
    while True:
        print("\n1. Register\n2. Login\n3. Exit")
        choice = input("Choose an option: ")
        if choice == '1':
            register_user()
        elif choice == '2':
            login_user()
        elif choice == '3':
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    main()
```

Output:

```
1. Register
2. Login
3. Exit
Choose an option: 1
Enter a username to register: sai
Enter a password: 1234
Registration successful!

1. Register
2. Login
3. Exit
Choose an option: 2
Enter your username: sai
Enter your password: 1234
Login successful!
```

```
1. Register
2. Login
3. Exit
Choose an option: 3
Exiting program.
```

Task Description#5

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

Expected Output#5

- Code quality difference analysis for various prompts

Prompt:

```
write a python code that will do Conversion of
Temperature using Function
```

Code:

```python
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

def main():
    while True:
        print("\nTemperature Conversion Menu:")
        print("1. Celsius to Fahrenheit")
        print("2. Fahrenheit to Celsius")
        print("3. Exit")
        choice = input("Choose an option: ")
        if choice == '1':
            c = float(input("Enter temperature in Celsius: "))
            f = celsius_to_fahrenheit(c)
            print(f"{c}°C = {f:.2f}°F")
        elif choice == '2':
            f = float(input("Enter temperature in Fahrenheit: "))
            c = fahrenheit_to_celsius(f)
            print(f"{f}°F = {c:.2f}°C")
        elif choice == '3':
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    main()
```

Output:

```
Temperature Conversion Menu:
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
3. Exit
Choose an option: 1
Enter temperature in Celsius: 50
50.0°C = 122.00°F

Temperature Conversion Menu:
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
3. Exit
Choose an option: 2
Enter temperature in Fahrenheit: 130
130.0°F = 54.44°C

Temperature Conversion Menu:
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
3. Exit
Choose an option: 3
Exiting program.
```

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**