# AIAC Assignment 10_3:

# BATCH 45

# 2303A52316

## Problem Statement 1: AI-Assisted Bug Detection

PROMT:

# def factorial(n):

# result = 1

# for i in range(1, n):

# result = result * i

# return result

# Run the code and test it with factorial(5).Identify the logical bug in the code.Explain why the bug occurs

# Provide a corrected version.Compare the AI's corrected code with your own manual fix.

CODE:

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):  # The loop should include n, so we use n + 1
        result = result * i
    return result
# Example usage
print(factorial(5))  # Output should be 120
# Explanation of the bug:
# The original code had a logical bug because the loop iterated from 1 to n-1, which means it did not include n in the multiplication.
```

## Problem Statement 2: Task 2 — Improving Readability

PROMT:

# ef calc(a, b, c):

# if c == "add":

# return a + b

# elif c == "sub":

# return a - b

# elif c == "mul":

# return a * b

# elif c == "div":

# Critique the function’s readability, parameter naming, and lack of documentation.rewrite Descriptive function and parameter names.

# Exception handling for division by zero. Consideration of input validation.Compare the original and AI-improved versions.

CODE:

```
def calculate(a, b, operation):
    """
    Performs a basic arithmetic operation on two numbers.

    Args:
        a (float): The first number.
        b (float): The second number.
        operation (str): The operation to perform ('add', 'subtract', 'multiply', 'divide').

    Returns:
```

```
        float: The result of the arithmetic operation.

    Raises:
        ValueError: If an invalid operation is provided.
        ZeroDivisionError: If division by zero is attempted.
    """
    if operation == "add":
        return a + b
    elif operation == "subtract":
        return a - b
    elif operation == "multiply":
        return a * b
    elif operation == "divide":
        if b == 0:
            raise ZeroDivisionError("Cannot divide by zero.")
        return a / b
    else:
        raise ValueError("Invalid operation. Please choose 'add', 'subtract',
'multiply', or 'divide'.")
```

Problem Statement 3: Enforcing Coding Standards

PROMT:

```
# def Checkprime(n):
# for i in range(2, n):
# if n % i == 0:
```

# return False

# return True

# Verify the function works correctly for sample inputs.Refactor the code (function name, spacing, indentation,naming).

# Apply the AI-suggested changes and verify functionality is preserved.

CODE:

```
def check_prime(number):
    """
    Checks if a number is prime.


    Args:
        number (int): The number to be checked.
    Returns:
        bool: True if the number is prime, False otherwise.
    """
    if number <= 1:
        return False  # 0 and 1 are not prime numbers
    for i in range(2, int(number**0.5) + 1):  # Check up to the square root of the number
        if number % i == 0:
            return False  # If divisible by any number other than 1 and itself, it's not prime
    return True  # If no divisors found, it's a prime number
```

Problem Statement 4: AI as a Code Reviewer in Real Projects Scenario:

PROMT:

#def processData(d):

# return [x * 2 for x in d if x % 2 == 0]

# Manually review the function for: Readability and naming.Reusability and modularity.Edge cases (non-list input, empty list, non-integerelements).

# Use AI to generate a code review covering:Better naming and function purpose clarity.Better naming and function purpose clarity.

# Refactor the function based on AI feedback.

CODE:

```
def process_data(data):
    """
    Processes a list of integers by doubling the even numbers.

    Args:
        data (list): A list of integers to be processed.

    Returns:
        list: A new list containing the doubled values of the even numbers from the input list.

    Raises:
        TypeError: If the input is not a list or contains non-integer elements.
    """
    if not isinstance(data, list):
        raise TypeError("Input must be a list.")
```

```
    processed_data = []
    for x in data:
        if not isinstance(x, int):
            raise TypeError("All elements in the list must be integers.")
        if x % 2 == 0:
            processed_data.append(x * 2)  # Double the even number and add to the processed list

    return processed_data  # Return the final processed list
```

Problem Statement 5: — AI-Assisted Performance Optimization

PROMT:

# def sum_of_squares(numbers):

# total = 0

# for num in numbers:

# total += num ** 2

# return total

# Test the function with a large list (e.g., range(1000000)).Analyze time complexity.Compare execution time before and after optimization.

# Discuss trade-offs between readability and performance.

CODE:

```
def sum_of_squares(numbers):
```

```
    """
    Calculates the sum of squares of a list of numbers.

    Args:
        numbers (list): A list of numbers to be processed.

    Returns:
        int: The sum of squares of the input numbers.
    """
    return sum(num ** 2 for num in numbers)  # Use a generator expression for better performance
```