- NAME : G SAIANIRUDH
- HTNO : 2303A51891
- BATCH : 04

Task Description-1

Progressive Prompting for Calculator Design: Ask the AI to design a simple calculator program by initially providing only the function name. Gradually enhance the prompt by adding comments and usage examples. Expected Output-1 • Comparison showing improvement in AI-generated calculator logic and structure.

```
#write a program to design a simple calculator
def add(x, y):
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
    if y == 0:
        return "Error! Division by zero."
    return x / y
print("Select operation:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")
choice = input("Enter choice (1/2/3/4): ")
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
if choice == '1':
    print(f"{num1} + {num2} = {add(num1, num2)}")
elif choice == '2':
    print(f"{num1} - {num2} = {subtract(num1, num2)}")
elif choice == '3':
    print(f"{num1} * {num2} = {multiply(num1, num2)}")
elif choice == '4':
    print(f"{num1} / {num2} = {divide(num1, num2)}")
else:
    print("Invalid input")
```

```
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice (1/2/3/4): 3
Enter first number: 5
Enter second number: 5
5.0 * 5.0 = 25.0
```

Task Description-2

Refining Prompts for Sorting Logic: Start with a vague prompt for sorting student marks, then refine it to clearly specify sorting order and constraints. Expected Output-2 • AI-generated sorting function evolves from ambiguous logic to an accurate and efficient implementation.

```
#write a program to sort student marks in ascending order
def sort_marks(marks):
    return sorted(marks)
student_marks = [88, 92, 79, 85, 95]
sorted_marks = sort_marks(student_marks)
print("Sorted student marks in ascending order:", sorted_marks)
```

```
Sorted student marks in ascending order: [79, 85, 88, 92, 95]
```

Task Description-3

• Few-Shot Prompting for Prime Number Validation: Provide multiple input-output examples for a function that checks whether a number is prime. Observe how few-shot prompting improves correctness. Expected Output-3 • Improved prime-checking function with better edge-case handling.

```
#Write a function to check whether a number is prime.

# Examples:
# Input: 2 → Output: True
```

```
# Input: 4 → Output: False
# Input: 7 → Output: True
# Input: 1 → Output: False

def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
# Test the function with some examples
test_numbers = [2, 4, 7, 1]
for number in test_numbers:
    result = is_prime(number)
    print(f"Is {number} a prime number? {result}")
```

```
Is 2 a prime number? True
Is 4 a prime number? False
Is 7 a prime number? True
Is 1 a prime number? False
```

Task Description-4

• Prompt-Guided UI Design for Student Grading System: Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input. Expected Output-4 • Well-structured UI code with accurate calculations and clear output display.

```
#Create a user interface for a student grading system.

# The UI should accept marks of 5 subjects.
# Calculate total marks and percentage.
# Display grade based on percentage:
# >=90: A
# >=75: B
# >=60: C
# <60: Fail

def calculate_grade(marks):
    total_marks = sum(marks)
    percentage = (total_marks / 500) * 100
    if percentage >= 90:
        grade = 'A'
    elif percentage >= 75:
        grade = 'B'
    elif percentage >= 60:
        grade = 'C'
    else:
        grade = 'Fail'
    return total_marks, percentage, grade
# Taking input for 5 subjects
marks = []
for i in range(1, 6):
    mark = float(input(f"Enter marks for subject {i}: "))
    marks.append(mark)
total, percent, grade = calculate_grade(marks)
print(f"Total Marks: {total}")
print(f"Percentage: {percent:.2f}%")
print(f"Grade: {grade}")
```

```
Enter marks for subject 1: 95
Enter marks for subject 2: 75
Enter marks for subject 3: 85
Enter marks for subject 4: 63
Enter marks for subject 5: 82
Total Marks: 400.0
Percentage: 80.00%
Grade: B
```

Task Description-5

• Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions. Expected Output-5 • Analysis of code quality and accuracy differences across multiple prompt variations.

```
#Write a function to convert kilometers to miles and miles to kilometers.

def km_to_miles(km):
    return km * 0.621371
def miles_to_km(miles):
    return miles / 0.621371
```

```
# Example usage
kilometers = 10
miles = 6.21371
print(f"{kilometers} kilometers is equal to {km_to_miles(kilometers)} miles")
print(f"{miles} miles is equal to {miles_to_km(miles)} kilometers")
```

```
10 kilometers is equal to 6.21371 miles
6.21371 miles is equal to 10.0 kilometers
```