Name: V. Vignesh

H.no:2303A51707

batch:24

Assignment 3.5

Question 1: Zero-Shot Prompting (Leap Year Check)

Write a zero-shot prompt to generate a Python function that checks

whether a given year is a leap year.

Week2 -

Task:

• Record the AI-generated code.

• Test with years like 1900, 2000, 2024.

• Identify logical flaws or missing conditions.

INPUT:

```python
#generate a python program to chech wether a given given year is leap year or not using function.
def is_leap_year(year):
    """Check if a given year is a leap year."""
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return f"{year} is a leap year."
    else:
        return f"{year} is not a leap year."
# user input and a valid input check,the output should be given when only the input is positive.
try:
    year_input = int(input("Enter a year to check if it is a leap year: "))
    if year_input < 0:
        print("Please enter a positive integer for the year.")
    else:
        print(is_leap_year(year_input))
except ValueError:
    print("Invalid input. Please enter a valid integer for the year.")
print("Task 1 completed.")
```

OUTPUT:

```
$ python assn_3.5.py
Enter a year to check if it is a leap year: 2020
2020 is a leap year.
Task 1 completed.
```

Question 2: One-Shot Prompting (GCD of Two Numbers)

Write a one-shot prompt with one example to generate a Python

function that finds the Greatest Common Divisor (GCD) of two numbers.

Example:

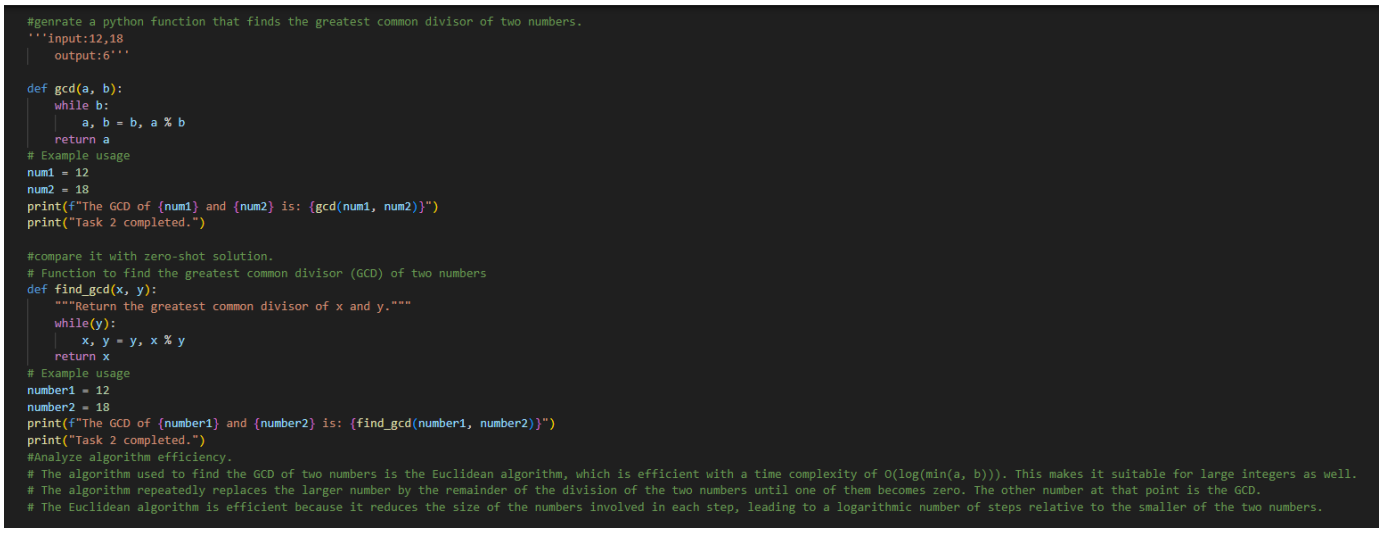
Input: 12, 18 → Output: 6

Task:

• Compare with a zero-shot solution.

• Analyze algorithm efficiency.

INPUT:

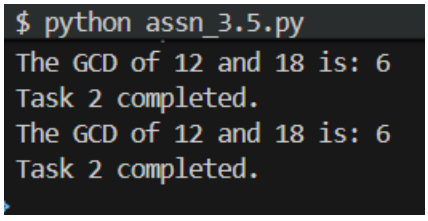


```python
#genrate a python function that finds the greatest common divisor of two numbers.
'''input:12,18
    output:6'''

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
# Example usage
num1 = 12
num2 = 18
print(f"The GCD of {num1} and {num2} is: {gcd(num1, num2)}")
print("Task 2 completed.")

#compare it with zero-shot solution.
# Function to find the greatest common divisor (GCD) of two numbers
def find_gcd(x, y):
    """Return the greatest common divisor of x and y."""
    while(y):
        x, y = y, x % y
    return x
# Example usage
number1 = 12
number2 = 18
print(f"The GCD of {number1} and {number2} is: {find_gcd(number1, number2)}")
print("Task 2 completed.")
#Analyze algorithm efficiency.
# The algorithm used to find the GCD of two numbers is the Euclidean algorithm, which is efficient with a time complexity of O(log(min(a, b))). This makes it suitable for large integers as well.
# The algorithm repeatedly replaces the larger number by the remainder of the division of the two numbers until one of them becomes zero. The other number at that point is the GCD.
# The Euclidean algorithm is efficient because it reduces the size of the numbers involved in each step, leading to a logarithmic number of steps relative to the smaller of the two numbers.
```

OUTPUT:



```
$ python assn_3.5.py
The GCD of 12 and 18 is: 6
Task 2 completed.
The GCD of 12 and 18 is: 6
Task 2 completed.
```

Question 3: Few-Shot Prompting (LCM Calculation)

Write a few-shot prompt with multiple examples to generate a Python

function that computes the Least Common Multiple (LCM).

Examples:

• Input: 4, 6 → Output: 12

• Input: 5, 10 → Output: 10

• Input: 7, 3 → Output: 21

Task:

• Examine how examples guide formula selection.

• Test edge cases.

INPUT:

```
assn_3.3.py > ...

#generate a python function program that computes the least common multiple(lcm)of two numbers.
'''
input:4,6
output:12
input:5,10
output:10
input:7,3
output:21
'''
def lcm(a, b):
    """Compute the least common multiple of two numbers."""
    def gcd(x, y):
        while y:
            x, y = y, x % y
        return x
    return abs(a * b) // gcd(a, b)
# Example usage
num1 = 4
num2 = 6
print(f"The LCM of {num1} and {num2} is: {lcm(num1, num2)}")
num3 = 5
num4 = 10
print(f"The LCM of {num3} and {num4} is: {lcm(num3, num4)}")
num5 = 7
num6 = 3
print(f"The LCM of {num5} and {num6} is: {lcm(num5, num6)}")
print("Task 3 completed.")
```

OUTPUT:

```
The LCM of 4 and 6 is: 12
The LCM of 5 and 10 is: 10
The LCM of 7 and 3 is: 21
Task 3 completed.
```

Question 4: Zero-Shot Prompting (Binary to Decimal Conversion)

Write a zero-shot prompt to generate a Python function that converts a

binary number to decimal.

Task:

• Test with valid and invalid binary inputs.

• Identify missing validation logic

INPUT:

```python
#Generate a python function program that converts a binary number to decimal number.
def binary_to_decimal(binary_str):
    """Convert a binary number (as a string) to a decimal number."""
    decimal_number = 0
    binary_str = binary_str[::-1]  # Reverse the string for easier calculation
    for index, digit in enumerate(binary_str):
        if digit == '1':
            decimal_number += 2 ** index
    return decimal_number
# user input with valid binary inputs.
binary_input = input("Enter a binary number: ")
if all(char in '01' for char in binary_input):
    decimal_output = binary_to_decimal(binary_input)
    print(f"The decimal equivalent of binary {binary_input} is: {decimal_output}")
else:
    print("Invalid binary number. Please enter a valid binary string.")
print("Task 4 completed.")
```

OUTPUT:

```
Enter a binary number: 1101
The decimal equivalent of binary 1101 is: 13
Task 4 completed.
```

Question 5: One-Shot Prompting (Decimal to Binary Conversion)

Write a one-shot prompt with an example to generate a Python function

that converts a decimal number to binary.

Example:

Input: 10 → Output: 1010

Task:

• Compare clarity with zero-shot output.

• Analyze handling of zero and negative numbers.

INPUT:

```python
#Generate a python function code that converts a decimal number to binary number that should analyze handling of zero and negative numbers.
'''input:10
   output:1010'''
def decimal_to_binary(n):
    """Convert a decimal number to a binary number (as a string)."""
    if n == 0:
        return "0"
    elif n < 0:
        return "Negative numbers cannot be converted to binary."

    binary_str = ""
    while n > 0:
        binary_str = str(n % 2) + binary_str
        n = n // 2
    return binary_str
# Example usage
decimal_input = int(input("Enter a decimal number: "))
binary_output = decimal_to_binary(decimal_input)
print(f"The binary equivalent of decimal {decimal_input} is: {binary_output}")
print("Task 5 completed.")
```

OUTPUT:

```
Enter a decimal number: 34
The binary equivalent of decimal 34 is: 100010
Task 5 completed.
```

Question 6: Few-Shot Prompting (Harshad Number Check)

Write a few-shot prompt to generate a Python function that checks

whether a number is a Harshad (Niven) number.

Examples:

• Input: 18 → Output: Harshad Number

• Input: 21 → Output: Harshad Number

• Input: 19 → Output: Not a Harshad Number

Task:

• Test boundary conditions.

• Evaluate robustness

INPUT:

```python
#Generate a python function program that checks whether a given number is Harshad number or not.
'''input:18
   output:Harshad number
   input:21
   output:Not a Harshad number
   intput:19
   output:Not a Harshad number'''
def is_harshad_number(n):
    """Check if a number is a Harshad number."""
    digit_sum = sum(int(digit) for digit in str(n))
    if n % digit_sum == 0:
        return f"{n} is a Harshad number."
    else:
        return f"{n} is not a Harshad number."
# Example usage
number_input = int(input("Enter a number to check if it is a Harshad number: "))
print(is_harshad_number(number_input))
print("Task 6 completed.")
#Test boundary conditions.
# Testing boundary conditions for Harshad number function
test_values = [1, 9, 10, 11, 18, 19, 20, 21, 100, 101]
for test_value in test_values:
    print(is_harshad_number(test_value))
# The boundary conditions tested include:
# - Single-digit numbers (1 to 9)
# - Transition from single-digit to double-digit numbers (9 to 10)
# - Known Harshad numbers (18, 100)
# - Known non-Harshad numbers (19, 21, 101)
# This ensures that the function behaves correctly across a range of inputs, including edge cases.
```

OUTPUT:

```
Enter a number to check if it is a Harshad number: 18
18 is a Harshad number.
Task 6 completed.
1 is a Harshad number.
9 is a Harshad number.
10 is a Harshad number.
11 is not a Harshad number.
18 is a Harshad number.
19 is not a Harshad number.
20 is a Harshad number.
21 is a Harshad number.
100 is a Harshad number.
101 is not a Harshad number.
```