Assignment-3.5

Name: NIMMALA NANDA MUKESH

HT.No: 2303A51410

Batch- 24

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.

```
Assignment-3.5.py > ...
1    #write a python code to print that checks whether the given yearr is leap year
2    year = int(input("Enter a year: "))
3    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
4        print(f"{year} is a leap year.")
5    else:
6        print(f"{year} is not a leap year.")
```

Output:

```
Enter a year: 1900
1900 is not a leap year.
PS C:\Users\SRAVANI\Documents\AI Assist> &
ssist/Assignment-3.5.py"
Enter a year: 2000
2000 is a leap year.
PS C:\Users\SRAVANI\Documents\AI Assist> &
ssist/Assignment-3.5.py"
Enter a year: 2024
2024 is a leap year.
PS C:\Users\SRAVANI\Documents\AI Assist>
```

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.

```
a,b=12,18
display gcd of a,b
'''

def compute_gcd(a, b):
    while b:
        a, b = b, a % b
    return a
a = 12
b = 18
gcd = compute_gcd(a, b)
print(f"The GCD of {a} and {b} is {gcd}.")
```

Output:

```
ssist/Assignment-3.5.py"
The GCD of 12 and 18 is 6.
```

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.

```
'''
a,b=4,6
display least common multiple of a.b
a,b=5,10
display least common multiple of a.b
a,b=7,3
display least common multiple of a.b'''
def compute_gcd(a, b):
    while b:
        a, b = b, a % b
    return a
def compute_lcm(a, b):
    return abs(a * b) // compute_gcd(a, b)
a = 4
b = 6
lcm = compute_lcm(a, b)
print(f"The LCM of {a} and {b} is {lcm}.")
a = 5
b = 10
lcm = compute_lcm(a, b)
print(f"The LCM of {a} and {b} is {lcm}.")
a = 7
b = 3
lcm = compute_lcm(a, b)
print(f"The LCM of {a} and {b} is {lcm}.")
```

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.
PS C:\Users\SRAVANI\Documents\AI Assist> 
```

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.

```
#write a python code for conversion of binary to decimal in short and optimized way
binary_input = input("Enter a binary number: ")
decimal_output = int(binary_input, 2)
print(f"The decimal equivalent of binary {binary_input} is {decimal_output}")
```

Output:

```
Enter a binary number: 101
The decimal equivalent of binary 101 is 5
```

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 is 10
output is 1010
display that convert decimal to binary'''
def decimal_to_binary(n):
    if n == 0:
        return "0"
    binary_str = ""
    while n > 0:
        binary_str = str(n % 2) + binary_str
        n = n // 2
    return binary_str
a = input("Enter a decimal number: ")
binary_output = decimal_to_binary(int(a))
print(f"The binary equivalent of decimal {a} is {binary_output}.")
```

Output:

```
Enter a decimal number: 10
The binary equivalent of decimal 10 is 1010.
```

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

```python
'''
a=18
b=21
c=19
display that checks the given number is a harshad(niven) number'''
def is_harshad_number(n):
    digit_sum = sum(int(digit) for digit in str(n))
    return n % digit_sum == 0
a = int(input("Enter a number: "))
b = int(input("Enter a number: "))
c = int(input("Enter a number: "))
for number in (a, b, c):
    if is_harshad_number(number):
        print(f"{number} is a Harshad number.")
    else:
        print(f"{number} is not a Harshad  number.")
```

Output:

```
Enter a number: 18
Enter a number: 21
Enter a number: 19
18 is a Harshad number.
21 is a Harshad number.
19 is not a Harshad  number.
```