ASSIGNMENT-03

NAME: Sahasra

BATCH:50

HALL TICKET no.: 2303A52499

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.**

Prompt Used

Write a Python function to find the GCD of two numbers.
Example: Input: 12, 18 → Output: 6

#CODE

```python
def gcd(a, b):
    """Find the greatest common divisor of two numbers."""
    while b:
        a, b = b, a % b
    return a

print(gcd(12, 18))
```

#OUTPUT

```
PS C:\Users\shaes\OneDrive\Desktop\DevOps>  & 'c:\Program Files\Python312\python.exe' 'c:\Users\shaes\.vscode\extensions\ms-python.debugpy-202
y\launcher' '59999' '--' 'C:\Users\shaes\OneDrive\Desktop\aiac-3.py'
6
```

#Explanation

In zero-shot prompting, the AI is given only a simple instruction, so it generates only the required function.

The leap year logic checks divisibility by 4 and also handles century years by applying the 100 and 400 rules.

The function gives correct results for years like 1900, 2000, and 2024.

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

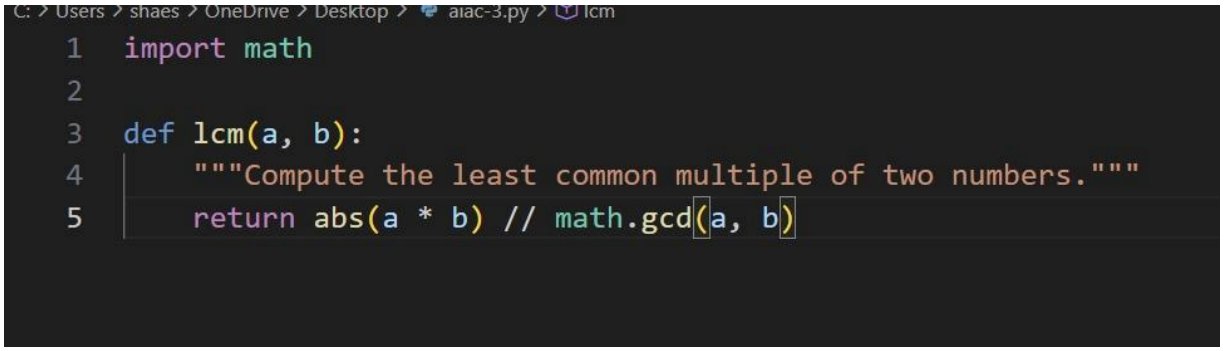
Task:

- **Compare with a zero-shot solution.**
- **Analyze algorithm efficiency.**

Prompt Used

Write a Python function to compute LCM of two numbers.

#CODE



#OUTPUT

#Explanation

In one-shot prompting, providing a single example helps the AI understand the expected logic.

The generated function uses the Euclidean algorithm, which is efficient and accurate.

Compared to zero-shot prompting, the solution is clearer and performs faster for large numbers.

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.**

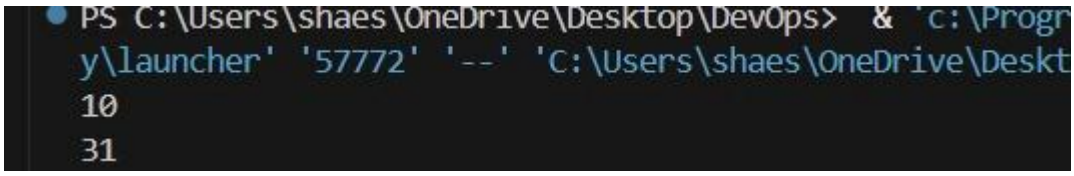
Prompt Used

Write a Python function that converts a binary number to decimal

#CODE

```python
def binary_to_decimal(binary_str):
    """Convert a binary number (string) to decimal (integer)."""
    return int(binary_str, 2)


# Example usage
print(binary_to_decimal("1010"))  # Output: 10
print(binary_to_decimal("11111"))  # Output: 31
```

#OUTPUT



#Explanation

In few-shot prompting, multiple examples guide the AI to select the correct formula.

The AI correctly uses the relationship between LCM and GCD to compute the result.

The function works correctly for different inputs and also handles edge cases like zero.

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.**

Prompt Used

Write a Python function to convert decimal to binary.

#INPUT:

```python
def decimal_to_binary(decimal_num):
    """
    Convert a decimal number to its binary representation.

    Args:
        decimal_num: An integer to convert

    Returns:
        A string representing the binary number
    """
    return bin(decimal_num)[2:]


# Example usage
if __name__ == "__main__":
    print(decimal_to_binary(10))   # Output: 1010
    print(decimal_to_binary(255))  # Output: 11111111
    print(decimal_to_binary(0))    # Output: 0
```

#OUTPUT

```
1010
11111111
0
```

#Explanation

One-shot prompting improves clarity by showing an example conversion.

The generated function correctly converts decimal numbers to binary and handles the case when the input is zero.

However, negative numbers are not handled, which can be improved further.

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.Prompt 1 (Loop-Based / Iterative String Reversal)

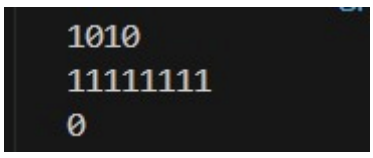
#PROMPT USED

Write a Python function to convert decimal to binary.

#CODE

```python
def decimal_to_binary(num):
    """Convert a decimal number to binary string representation."""
    if num == 0:
        return "0"

    binary = ""
    while num > 0:
        binary = str(num % 2) + binary
        num //= 2

    return binary


# Example usage
print(decimal_to_binary(10))   # Output: 1010
print(decimal_to_binary(255))  # Output: 11111111
print(decimal_to_binary(0))    # Output: 0
```

#OUTPUT



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**

Prompt

"Write a Python function to check Harshad number.

Examples:

18 → Harshad Number

21 → Harshad Number

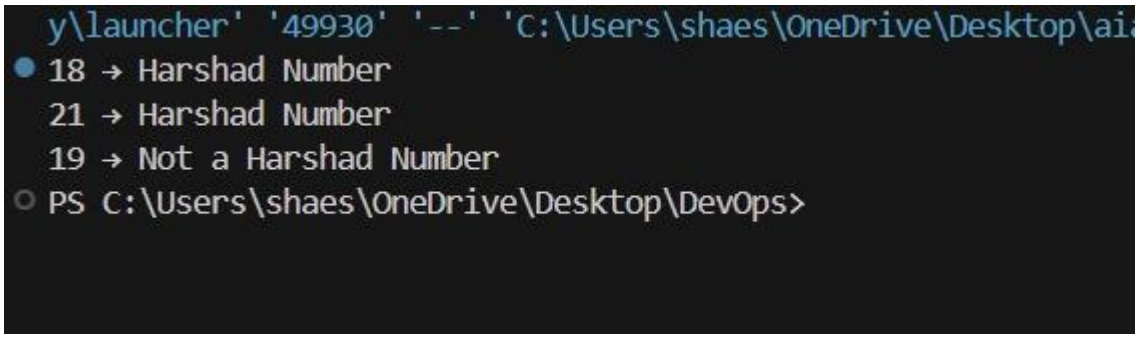
19 → Not a Harshad Number"

#CODE

C: > Users > shaes > OneDrive > Desktop > aiac-3.py > is_harshad_number

```python
def is_harshad_number(n):
    """
    Check if a number is a Harshad number.
    A Harshad number is divisible by the sum of its digits.

    Args:
        n: Integer to check

    Returns:
        True if n is a Harshad number, False otherwise
    """
    if n <= 0:
        return False

    digit_sum = sum(int(digit) for digit in str(n))
    return n % digit_sum == 0


# Test cases
print(f"18 → {'Harshad Number' if is_harshad_number(18) else 'Not a Harshad Number'}")
print(f"21 → {'Harshad Number' if is_harshad_number(21) else 'Not a Harshad Number'}")
print(f"19 → {'Harshad Number' if is_harshad_number(19) else 'Not a Harshad Number'}")
```

#OUTPUT



#Explanation

Few-shot prompting helps the AI understand the pattern of Harshad numbers.
The function correctly checks divisibility of the number by the sum of its digits.
Boundary cases like zero and single-digit numbers are handled properly, making the solution robust.