SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

Program Name: B. Tech	**Assignment Type: Lab**	**Academic Year:**2025-2026

Course Coordinator Name	Dr. Rishabh Mittal
Instructor(s) Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju

CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week2 –**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches

Assignment Number: 4.1(Present assignment number)/**24**(Total number of assignments)

Q.No.	**Question**	***Expected Time to complete***

1

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.

Task:

Record the AI-generated code.

Test with years like 1900, 2000, 2024.

Identify logical flaws or missing conditions.

OUTPUT

PROMPT : `Write a zero-shot prompt to generate a Python function that checks whether a given year is a leap year`

CODE

```python
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
year=int(input("Enter a year: "))
print(is_leap_year(year))
```

Code output

Testcase 1

Enter a year: 2021

False

Testcase 2

Enter a year: 2024

True

- ***Explanation:***

Week2 -

- The function `is_leap_year` checks whether a given year follows leap year rules.
- A year is a leap year if it is divisible by 4 and not divisible by 100.
- Years divisible by 400 are always leap years, even if divisible by 100.
- The program takes a year as input and prints `True` for a leap year, otherwise `False`

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.

OUTPUT

PROMPT : Write a one-shot prompt with one example to generate a Python function that finds the Greatest Common Divisor (GCD) of two numbers

Code

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
a=int(input("Enter first number: "))
b=int(input("Enter second number: "))
print(gcd(a,b))
```

Code output

```
Enter first number: 12
Enter second number: 18
6
```

Comparing :

Zero-shot: The GCD program is written directly without any example.

- **One-shot:** One example (e.g., 12 and 18 → 6) is given before writing the program.
- Zero-shot depends only on prior knowledge of the algorithm.
- One-shot uses the example to understand the expected behavior.
- Both produce the same output, but one-shot is easier to understand for beginners.

Algorithm Efficiency :

- The GCD program uses the **Euclidean algorithm**, which is very efficient.
- **Time complexity:** `O(log(min(a, b)))` because the values reduce quickly each step.
- **Space complexity:** `O(1)` since no extra memory is used.
- It works efficiently even for very large numbers.
- This is the most optimal method for finding GCD.

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.

OUTPUT

PROMPT Write a Python function that calculates the Least Common Multiple (LCM) of two integers.

CODE

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
def lcm(x, y):
    return abs(x * y) // gcd(x, y)
# Example usage:
num1 = 4
num2 = 6
print(f"The LCM of {num1} and {num2} is {lcm(num1,
num2)}")
```

CODE OUTPUT

TESTCASE 1

The LCM of 4 and 6 is 12

TESTCASE 2

The LCM of 5 and 10 is 10

- ***Explanation***
- The `gcd(a, b)` function uses the Euclidean algorithm to repeatedly replace the larger number with the remainder until it becomes zero.
- When b becomes zero, a holds the greatest common divisor and is returned.
- The `lcm(x, y)` function calculates the least common multiple using the formula `|x × y| // gcd(x, y)`.
- Using GCD makes the LCM calculation faster than checking multiples

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.

OUTOUT

PROMPT ***Write a Python function that takes a binary number (as a string) and returns its decimal equivalent.***

CODE

```python
def binary_to_decimal(binary_str):
    return int(binary_str, 2)
binary_num = "1011"
print(f"The decimal equivalent of {binary_num} is 
{binary_to_decimal(binary_num)}")
```

CODE OUTPUT

The decimal equivalent of 1011 is 11

- ***EXPLANATION***

  The function `binary_to_decimal(binary_str)` takes a binary number as a **string** input.
- `int(binary_str, 2)` converts the binary string into its **decimal equivalent**. The 2 tells Python that the input is in base 2.
- In the example, `binary_num = "1011"` represents the binary number `1011`.
- Calling `binary_to_decimal("1011")` returns `11` because `1011` in binary equals `11` in decimal.
- The `print` statement displays the result in a readable format:
  `"The decimal equivalent of 1011 is 11`

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.

OUTPUT

PROMPT

```
Write a one-shot prompt with an example to generate a
Python function that converts a decimal number to binary.
```

CODE

```python
def decimal_to_binary(decimal_num):
    return bin(decimal_num)[2:]
```

```
decimal_num = 10
print(f"The binary equivalent of {decimal_num} is
{decimal_to_binary(decimal_num)}")
```

CODE OUTPUT

The binary equivalent of 10 is 1010

EXPLANATION

- he function `decimal_to_binary(decimal_num)` takes a **decimal number** as input.
- `bin(decimal_num)` converts the decimal number to a binary string prefixed with `"0b"`.
- `[2:]` removes the `"0b"` prefix, leaving only the binary digits.
- In the example, `decimal_num = 10`, which is `1010` in binary.
- The `print` statement displays: `"The binary equivalent of 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

OUTPUT

PROMPT   Write a few-shot prompt to generate a Python function that checks whether a number is a Harshad (Niven) number.

Code

```python
def is_harshad_number(num):
    if num <= 0:
        return False

    digit_sum = sum(int(digit) for digit in str(num))
    return num % digit_sum == 0
# Example usage:
number = 18
if is_harshad_number(number):
    print(f"{number} is a Harshad number.")
else:
    print(f"{number} is not a Harshad number.")
```

Code output

Testcase 1

18 is a Harshad number.

Testcase 2

19 is not a Harshad number

Explanation

- The function `is_harshad_number(num)` takes an integer `num` as input.
- It first checks if `num` is less than or equal to 0; if so, it returns `False` because Harshad numbers are positive integers.
- `digit_sum = sum(int(digit) for digit in str(num))` calculates the sum of all digits in the number.
- `num % digit_sum == 0` checks if the number is divisible by the sum of its digits. If yes, the function returns `True`; otherwise, `False`.
- The `if` statement prints a readable message indicating whether `number = 18` is a Harshad number.