Batch-06 **id-2403A51101** **Name :Guggilla Anuja**

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName: B. Tech		**Assignment Type: Lab**	**AcademicYear:** 2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week2 - Wednesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber: **2.3**(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*ExpectedTime to complete*
1	Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques **Lab Objectives:** • To explore and apply different levels of prompt examples in AI-assisted code generation.	Week2 - Wednesday

- To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality.
- To evaluate the impact of context richness and example quantity on AI performance.
- To build awareness of prompt strategy effectiveness for different problem types.

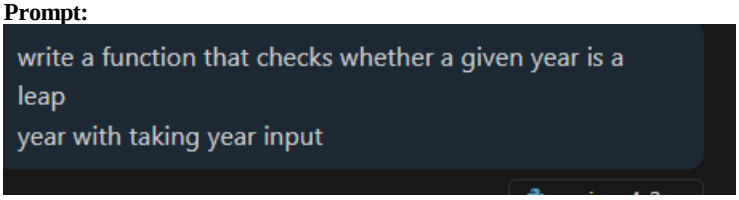
Lab Outcomes (LOs):
After completing this lab, students will be able to:

- Use zero-shot prompting to instruct AI with minimal context.
- Use one-shot prompting with a single example to guide AI code generation.
- Apply few-shot prompting using multiple examples to improve AI responses.
- Compare AI outputs across the three prompting strategies.

Task Description#1

- Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

Prompt:

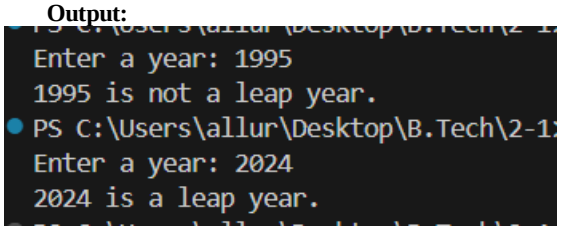


Code:

```
def is_leap_year(year):
    """
    Returns True if the given year is a leap year, False otherwise.
    """
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

def check_leap_year():
    year = int(input("Enter a year: "))
    if is_leap_year(year):
        print(f"{year} is a leap year.")
    else:
        print(f"{year} is not a leap year.")
check_leap_year()
```

Output:



Expected Output#1

- AI-generated function with no examples provided

Task Description#2

- One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

Prompt:

writing a function that converts centimeters to inches.
cand call the function and take input and call the function

Code:

```
def cm_to_inches(cm):
    """
    Converts centimeters to inches.
    """
    return cm / 2.54

def convert_cm_to_inches():
    cm = float(input("Enter length in centimeters: "))
    inches = cm_to_inches(cm)
    print(f"{cm} cm is equal to {inches:.2f} inches.")

convert_cm_to_inches()
```

Output:

```
Enter length in centimeters: 59
59.0 cm is equal to 23.23 inches.
PS C:\Users\allur\Desktop\B.Tech\2-1> p
Enter length in centimeters: 546
546.0 cm is equal to 214.96 inches.
```

Expected Output#2

- Function with correct conversion logic

Task Description#3

- Few-shot: Provide 2–3 examples to generate a function that formats full names as "Last, First".

Expected Output#3

- Well-structured function respecting the examples

Prompt:

generate a python code that formats full names as "Last, First" using functions

Code:

```python
def format_name(first, last):
    """
    Formats the full name as 'Last, First'.
    """
    return f"{last}, {first}"

def get_and_format_name():
    first = input("Enter first name: ")
    last = input("Enter last name: ")
    formatted = format_name(first, last)
    print(f"Formatted name: {formatted}")

# Example usage
get_and_format_name()
```

Output:

```
Enter first name: allu
Enter last name: kyath
Formatted name: kyath, allu
PS C:\Users\allur\Desktop\B.Tech\
Enter first name: miss
Enter last name: sri
Formatted name: sri, miss
```

Task Description#4

- Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.

Expected Output#4

- Functional output and comparative reflection

Prompt:

```
generate a python code that counts the number of
vowels in a string using a function
```

Code:

```
def count_vowels(s):
    """
    Counts the number of vowels in the given string.
    """
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count

def get_string_and_count_vowels():
    s = input("Enter a string: ")
    num_vowels = count_vowels(s)
    print(f"Number of vowels in the string: {num_vowels}")

get_string_and_count_vowels()
```

Output:

```
PS C:\Users\allur\Desktop\B.Tech\2-1> python -u "c:\U
Enter a string: ashgfyi7sdbsnamvdas
Number of vowels in the string: 4
PS C:\Users\allur\Desktop\B.Tech\2-1> python -u "c:\U
Enter a string: jshgdjyadtyastdjwevdmewd
Number of vowels in the string: 4
PS C:\Users\allur\Desktop\B.Tech\2-1> python -u "c:\U
Enter a string: aaaaaaaaaaaaaaa
Number of vowels in the string: 16
PS C:\Users\allur\Desktop\B.Tech\2-1> python -u "c:\U
Enter a string: aeiouwggcvdka
Number of vowels in the string: 6
PS C:\Users\allur\Desktop\B.Tech\2-1>
```

Task Description#5

- Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.

Expected Output#5

- Working file-processing function with AI-guided logic

Prompt:

```
generate a python code that creates,inserts textreads a
.txt file and returns the number of lines using functions
```

Code:

```python
def create_and_write_file(filename, text):
    """
    Creates a .txt file and writes the given text to it.
    """
    with open(filename, 'w') as f:
        f.write(text)

def read_file_and_count_lines(filename):
    """
    Reads the file and returns the number of lines.
    """
    with open(filename, 'r') as f:
        lines = f.readlines()
    return len(lines)

def file_operations():
    filename = "sample.txt"
    text = input("Enter text to write to the file (use \\n for new lines):\n")
    # Replace literal \n with actual newlines
    text = text.replace("\\n", "\n")
    create_and_write_file(filename, text)
    num_lines = read_file_and_count_lines(filename)
    print(f"Number of lines in '{filename}': {num_lines}")
```

Output:

```
Number of lines in 'sample.txt': 1
```

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Zero Shot (Task #1)	0.5
One Shot (Task#2)	0.5
Few Shot (Task#3 & Task #5)	1.0
Comparison (Task#4)	0.5
Total	**2.5 Marks**