Lab Assignment- 3.1

Experiment – Prompt Engineering Techniques

P.Karthik

2303A51908

Batch-12

Question 1: Zero-Shot Prompting (Palindrome Number Program)

Prompts: Generate a Python function that checks whether a given number is a palindrome.

Code:

```
# to generate a Python function that checks whether a given number is a palindrome.
def is_palindrome(number):
    # Convert the number to string
    str_num = str(number)
    # Check if the string is the same forwards and backwards
    return str_num == str_num[::-1]
```

Output:

```
Enter a number to check if it's a palindrome: 121
True
```

Explaination: This code determines whether a number entered by the user is a palindrome, which means it reads the same forwards and backwards. When the user inputs a number, the code first converts it to a string so that it can easily compare the sequence of digits. It then checks if this string is identical to its reverse. If they match, the function returns True, indicating the number is a palindrome; otherwise, it returns False. The result is printed out, letting the user know whether their number is a palindrome or not.

Question 2: One-Shot Prompting (Factorial Calculation)

Prompts: Generate a Python function to compute the factorial of a given number.

Code:

```
#Question 2: One-Shot Prompting (Factorial Calculation)
# Write a one-shot prompt by providing one input-output example and ask the AI to
# generate a Python function to compute the factorial of a given number.
def factorial(n):
    # Base case
    if n == 0 or n == 1:
        return 1
    # Recursive case
    return n * factorial(n - 1)
# Example usage:
print(factorial(5))  # 120
print(factorial(0))  # 1
print(factorial(3))  # 6
```

Output:

```
120
1
6
PS C:\Users\chari> &
```

Explaination: This code defines a function to calculate the factorial of a given number using recursion. The factorial of a number (written as n!) is the product of all positive integers up to that number. For example, 5! = 5 × 4 × 3 × 2 × 1 = 120. The function checks if the input is 0 or 1, in which case it returns 1, since the factorial of both 0 and 1 is 1. The example usage shows how the function works for different inputs, printing the results for 5, 0, and 3.

Question 3: Few-Shot Prompting (Armstrong Number Check)

Prompts: Generating a Python function to check whether a given number is an Armstrong number.

Code:

```python
#Question 3: Few-Shot Prompting (Armstrong Number Check)
# Write a few-shot prompt by providing multiple input-output examples to guide the AI in
# generating a Python function to check whether a given number is an Armstrong number.
def is_armstrong(number):
    # Convert the number to string to easily iterate over digits
    str_num = str(number)
    num_digits = len(str_num)
    sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
    return sum_of_powers == number
# Example usage:
print(is_armstrong(153))  # True
print(is_armstrong(9474)) # True
print(is_armstrong(123))  # False
```

Output:

```
True
True
False
PS C:\Users\chari>
```

Explaination: This code defines a function to check if a number is an Armstrong number. An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits. The function works by first converting the number to a string so it can easily access each digit. It then calculates the number of digits and computes the sum of each digit raised to that power. If this sum matches the original number, the function returns True; otherwise, it returns False

Question 4: Context-Managed Prompting (Optimized Number Classification)

Prompts: Generate an optimized Python program that classifies a number as prime, composite, or neither.

Code:

```
# Question 4: Context-Managed Prompting (Optimized Number
# Classification)
# Design a context-managed prompt with clear instructions and constraints to
# generate an optimized Python program that classifies a number as prime, composite, or neither.
def classify_number(n):
    if n <= 1:
        return "neither"
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return "composite"
    return "prime"
# Example usage:
print(classify_number(1))   # neither
print(classify_number(2))   # prime
print(classify_number(4))   # composite
```

Output:

```
neither
prime
composite
```

Explaination: This code defines a function that classifies a given number as "prime," "composite," or "neither." If the input number is less than or equal to 1, it returns "neither" because 0 and 1 are not considered prime or composite. For numbers greater than 1, it checks if the number is divisible by any integer from 2 up to the square root of the number. If it finds a divisor, the number is "composite".

Question 5: Zero-Shot Prompting (Perfect Number Check)

Prompts: Generate a Python function that checks whether a given number is a perfect number.

Code:

```python
#Question 5: Zero-Shot Prompting (Perfect Number Check)
# Write a zero-shot prompt (without providing any examples) to
# generate a Python function that checks whether a given number is a
# perfect number.
def is_perfect_number(number):
    if number < 1:
        return False
    divisors_sum = sum(i for i in range(1, number) if number % i == 0)
    return divisors_sum == number
# Example usage:
print(is_perfect_number(6))    # True
print(is_perfect_number(28))   # True
print(is_perfect_number(12))   # False
```

Output:

```
True
True
False
```

Explaination: This code defines a function to check if a number is a perfect number. A perfect number is a positive integer that is equal to the sum of all its proper divisors (excluding itself). The function first checks if the number is less than 1, returning False since perfect numbers must be positive. It then calculates the sum of all divisors of the number from 1 up to (but not including) the number itself. If this sum equals the original number, the function returns True.

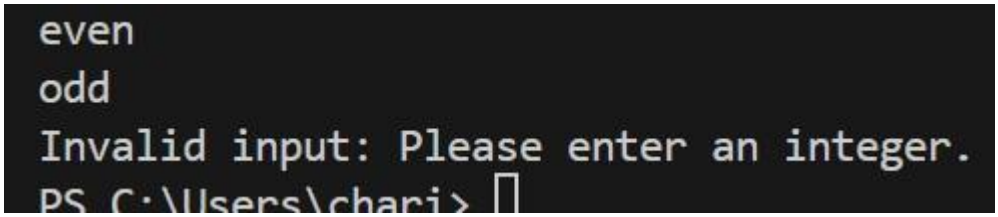
Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

Prompts: Determines whether a given number is even or odd, including proper input validation.

Code:

```python
#Question 6: Few-Shot Prompting (Even or Odd Classification with
# Validation)
# Write a few-shot prompt by providing multiple input-output
# examples to guide the AI in generating a Python program that
# determines whether a given number is even or odd, including proper
# input validation.
def even_or_odd(number):
    if not isinstance(number, int):
        return "Invalid input: Please enter an integer."
    return "even" if number % 2 == 0 else "odd"
# Example usage:
print(even_or_odd(4))      # even
print(even_or_odd(7))      # odd
print(even_or_odd(3.5))    # Invalid input: Please enter an
```

Output:



Explaination: This code defines a function that determines whether a given number is even or odd, with input validation. The function first checks if the input is an integer; if not, it returns a message asking for a valid integer. If the input is an integer, it checks if the number is divisible by 2. If it is, the function returns "even"; otherwise, it returns "odd."