WEEK-3

Name: A.KARTHIK

Halltick No: 2303A51132

Batch: 17

Question 1: Zero-Shot Prompting (Palindrome Number Program) Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a palindrome.

Task:

. Record the Al-generated code.

. Test the code with multiple inputs.

. Identify any logical errors or missing edge-case handling.

PROMPT:

check whether the given number is a palindrome or not, take input from user.

PROGRAM:

```
def is_palindrome(number):
    """
    Checks if a given number is a palindrome.
    """
    num_str = str(number)     return
num_str == num_str[::-1] # Take input
from the user user_input =
input("Enter a number: ")
try:
    dynamic_number = int(user_input)
if is_palindrome(dynamic_number):
        print(f"The number {dynamic_number} IS a palindrome.")
```

```
    else:
        print(f"The number {dynamic_number} IS NOT a palindrome.") except
ValueError:
    print("Invalid input. Please enter an integer.")
```

OUTPUT:

```
Enter a number: 323
The number 323 IS a palindrome.
```

```
Enter a number: 1234
The number 1234 IS NOT a palindrome.
```

ANALYSIS:

First the program given was a static code, where the user cannot give input in terminal. Then the Ai assistant used Exceptional handling to take input from user. Final code has good readability which can take input from user.

Question 2: One-Shot Prompting (Factorial Calculation)

Write a one-shot prompt by providing one input-output example and ask

the Al to generate a Python function to compute the factorial of a given

number.

Example:

Input: 5 > Output: 120

Task:

· Compare the generated code with a zero-shot solution.

. Examine improvements in clarity and correctness.

PROMPT:

Give factorial of a given number, take input from user but do not use exceptional handing.
Example: input:5, output:120

PROGRAM:

```
def factorial(n):
    """
    Calculates the factorial of a non-negative integer.
    """     if n
== 0:
return 1
else:
       res = 1        for i in
range(1, n + 1):
           res *= i
return res
# Take input from the user user_input = input("Enter a
number to find its factorial: ")
# Convert input to integer (assuming valid integer input as per request) dynamic_number
= int(user_input)


# Calculate and print the factorial result =
factorial(dynamic_number) print(f"The factorial of
{dynamic_number} is: {result}")
```

OUTPUT:

```
Enter a number to find its factorial: 12
The factorial of 12 is: 479001600
```

ANALYSIS:

The logic is same even when I gave the prompt without example. So as this is a small problem with no big edge cases it gave right code even before mentioning the example of 5!=120.

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

few-shot prompt by providing multiple input-output examples to

guide the Al in generating a Python function to check whether a given

number is an Armstrong number.

Examples:

. Input: 153 > Output: Armstrong Number

. Input: 370 > Output: Armstrong Number .

Input: 123 > Output: Not an Armstrong Number

Task:

· Analyze how multiple examples influence code structure and accuracy.

. Test the function with boundary values and invalid inputs.

PROMPT:

Give a function to check whether the given number is armstrong or not. take input from user without exceptional handling.

Examples: Input: 153 , Output: Armstrong Number

Input: 370 , Output: Armstrong Number

Input: 123 , Output: Not an Armstrong Number

PROGRAM:

```
def is_armstrong(number):
    """
    Checks if a given 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.

"""

Convert the number to a string to easily access its digits and count them

num_str = str(number) n = len(num_str) # Get the number of digits

sum_of_powers = 0 for digit_char in num_str: digit = int(digit_char)

sum_of_powers += digit ** n return sum_of_powers == number # Take

input from the user user_input = input("Enter a number: ")

Convert input to integer (assuming valid integer input as per request) dynamic_number

= int(user_input)

Check and print if it's an Armstrong number

if is_armstrong(dynamic_number):

print("Armstrong Number") else:

print("Not an Armstrong Number")

OUTPUT:

```
Enter a number: 153
Armstrong Number
```

```
Enter a number: 135
Not an Armstrong Number
```

ANALYSIS:

The program did not change its logic given after giving many examples as the program was right even for edge cases with the first example itself.

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.

Task:

. Ensure proper input validation.

. Optimize the logic for efficiency.

. Compare the output with earlier prompting strategies.

PROMPT:

Generate a Python program that classifies a number as prime, composite, or neither. Take the user input without exceptional handling.

PROGRAM:

```
def classify_number(num):
    """
    Classifies a number as prime, composite, or neither.
- Numbers less than or equal to 1 are classified as 'neither'.
- Prime numbers have exactly two distinct positive divisors: 1 and itself.
- Composite numbers have more than two distinct positive divisors.
    """     if
num <= 1:
return
"neither (less
than or equal
to 1)" elif
num == 2:
return
"prime" elif
num % 2 == 0:
```

```
        return "composite"
    else:
        # Check for divisibility by odd numbers from 3 up to sqrt(num)
        i = 3
        while i * i <= num:
if num % i == 0:
                return "composite"
            i += 2
        return "prime"
# Take input from the user user_input = input("Enter an integer to classify
(prime, composite, or neither): ") try:
    number_to_classify = int(user_input)     classification =
classify_number(number_to_classify)     print(f"The number
{number_to_classify} is classified as: {classification}.") except ValueError:
    print("Invalid input. Please enter an integer.")
```

OUTPUT:

```
Enter an integer to classify (prime, composite, or neither): 3
The number 3 is classified as: prime.
```

ANALYSIS:

After proper input validation, it does not give ValueError.

```
Enter an integer to classify (prime, composite, or neither): A
Invalid input. Please enter an integer.
```

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

perfect number.

Task:

Record the Al-generated code.

. Test the program with multiple inputs.

Identify any missing conditions or inefficiencies in the logic.

PROMPTS:
Check whether the given number is perfect or not. Take input from user.

PROGRAM:

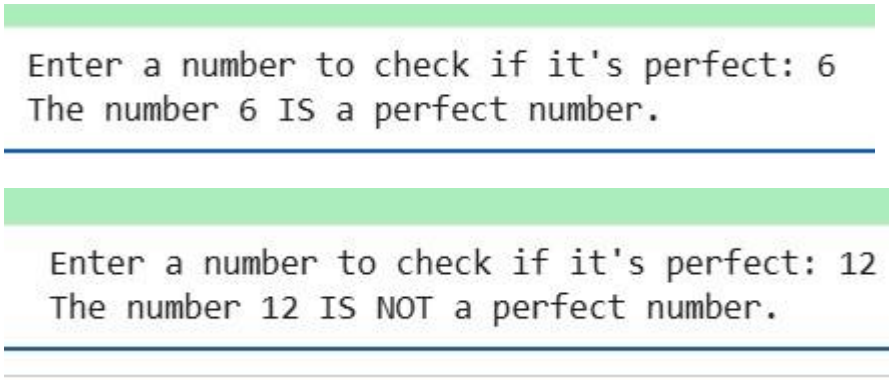
```
def is_perfect(number):
    """
    Checks if a given number is a perfect number.

    A perfect number is a positive integer that is equal to the sum of its
proper positive divisors (divisors excluding the number itself).
    """     if number <= 1:         return False     sum_of_divisors = 1 #
Start with 1 because 1 is always a divisor for i in range(2,
int(number**0.5) + 1):     if number % i == 0:
sum_of_divisors += i         if i * i != number: # Add the other
divisor if it's not the square root             sum_of_divisors +=
(number // i)

    return sum_of_divisors == number # Take input from the
user user_input = input("Enter a number to check if it's
perfect: ") try:
```

```
    dynamic_number = int(user_input)
if is_perfect(dynamic_number):
        print(f"The number {dynamic_number} IS a perfect number.")
    else:
        print(f"The number {dynamic_number} IS NOT a perfect number.") except
ValueError:
    print("Invalid input. Please enter an integer.")
```

OUTPUT:



ANALYSIS:

The program works fine with multiple inputs and also has input validation.

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 Al in generating a Python program that determines whether a given number is even or odd, including proper

input validation.

Examples:

. Input: 8 > Output: Even

. Input: 15 > Output: Odd

. Input: 0 > Output: Even

Task:

. Analyze how examples improve input handling and output

clarity.

PROMPTS:
Classify Even or Odd with Validation. take input from user.

Clarify with these examples, Examples: . Input: 8 > Output: Even . Input: 15 > Output: Odd . Input: 0 > Output: Even.

PROGRAM:

```
def classify_even_odd(number):
    """
    Classifies a number as 'Even' or 'Odd'.
    """
    if number % 2 == 0:
        return "Even"
else:
        return "Odd"
# Take input from the user user_input = input("Enter an integer
to classify as Even or Odd: ") try:
    # Convert input to integer
num_to_classify = int(user_input)

    # Classify the number     classification =
classify_even_odd(num_to_classify)     print(f"The number
{num_to_classify} is {classification}.") except ValueError:
    print("Invalid input. Please enter a valid integer.")
```

OUTPUT:

```
Enter an integer to classify as Even or Odd: 3
The number 3 is Odd.
```

```
Enter an integer to classify as Even or Odd: 2
The number 2 is Even.
```

ANALYSIS:

There is no change in the code, before or after mentioning the examples in prompt.