

NAME:G.BHAGATH

H.NO:2303A51807

BATCH:26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	
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: 3.4 (Present assignment number)/ 24 (Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques		Week2

Task 1: Zero-shot Prompt – Fibonacci Series Generator**Task Description #1**

- Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers.

Expected Output #1

- A complete Python function generated by Copilot without any example provided.
- Correct output for sample input $N = 7 \rightarrow 0 1 1 2 3 5 8$
- Observation on how Copilot understood the instruction with zero context.

```
C: > Users > shash > AAC A(3.4).py > ...
1  def print_fibonacci(n):
2      if n <= 0:
3          return
4      a, b = 0, 1
5      print(a, end=" ")
6      if n > 1:
7          print(b, end=" ")
8      for i in range(2, n):
9          a, b = b, a + b
10         print(b, end=" ")
11     print()
12
13 # Test with input N = 7
14 print_fibonacci(7)
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash'>
Customer Charges (CC): $50.00
Electricity Duty (ED): $16.00
-----
TOTAL BILL AMOUNT: $326.00
=====
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash'
● PS C:\Users\shash> 7
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash'
0 1 1 2 3 5 8
○ PS C:\Users\shash> []
```

Task 2: One-shot Prompt – List Reversal Function

Task Description #2

- Write a comment prompt to reverse a list and provide one example below the comment to guide Copilot.

Expected Output #2

- Copilot-generated function to reverse a list using slicing or loop.
- Output: [3, 2, 1] for input [1, 2, 3]
- Observation on how adding a single example improved Copilot's

accuracy.

```
C:\> Users > shash > AAC A(3.4).py > ...
1  def reverse_list(lst):
2  |     return lst[::-1]
3
4  # Test with input [1, 2, 3]
5  result = reverse_list([1, 2, 3])
6  print(result)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\shash\python.exe' AAC A(3.4).py
-----
TOTAL BILL AMOUNT: $326.00
=====
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\shash\python.exe' AAC A(3.4).py
● PS C:\Users\shash> 7
7
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\shash\python.exe' AAC A(3.4).py
0 1 1 2 3 5 8
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\shash\python.exe' AAC A(3.4).py
[3, 2, 1]
○ PS C:\Users\shash> []
```

Task 3: Few-shot Prompt – String Pattern Matching

Task Description #3

- Write a comment with 2–3 examples to help Copilot understand how to check if a string starts with a capital letter and ends with a period.

Expected Output #3

- A function `is_valid()` that checks the pattern.
- Output: True or False based on input.

- Students reflect on how multiple examples guide Copilot to generate more accurate code.

```
C:\> Users > shash > AAC A(3.4).py > ...
1 def is_valid(s):
2     if not s: # Empty string
3         return False
4     return s[0].isupper() and s[-1] == '.'
5
6 # Test inputs
7 print(is_valid("Hello."))
8 print(is_valid("hello."))
9 print(is_valid("Hello"))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\shash> c:> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash' 7
● PS C:\Users\shash> 7
● PS C:\Users\shash> c:> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash' 0 1 1 2 3 5 8
● PS C:\Users\shash> c:> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash' [3, 2, 1]
● PS C:\Users\shash> c:> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash' True
False
False
PS C:\Users\shash>
```

Task 4: Zero-shot vs Few-shot – Email Validator

Task Description #4

- First, prompt Copilot to write an email validation function using zero-shot (just the task in comment).
- Then, rewrite the prompt using few-shot examples.

Expected Output #4

- Compare both outputs:

Zero-shot may result in basic or generic validation.

Few-shot gives detailed and specific logic (e.g., @ and domain checking).

- Submit both code versions and note how few-shot improves

reliability.

The screenshot shows a terminal window with the following content:

```
C:\> Users > shash > AAC A(3.4).py > ...
1 import re
2
3 def validate_email(email):
4     pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
5     return bool(re.match(pattern, email))
6
7 # Test inputs
8 print(validate_email("user@example.com")) # True
9 print(validate_email("user@")) # False
10 print(validate_email("user.example.com")) # False
```

Below the code, the terminal shows the execution results:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' AAC A(3.4).py
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' [3, 2, 1]
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' True
True
False
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' False
False
False
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' None
None
None
● PS C:\Users\shash> [ ]
```

Task 5: Prompt Tuning – Summing Digits of a Number

Task Description #5

- Experiment with 2 different prompt styles to generate a function that returns the sum of digits of a number.

Style 1: Generic task prompt

Style 2: Task + Input/Output example

Expected Output #5

- Two versions of the `sum_of_digits()` function.
- Example Output: `sum_of_digits(123) → 6`
- Short analysis: which prompt produced cleaner or more

optimized code and why?

The screenshot shows a terminal window with the following content:

```
C:\Users> shash > AAC A(3.4).py > ...
1 def sum_of_digits(n):
2     total = 0
3     while n > 0:
4         total += n % 10
5         n = n // 10
6     return total
7
8 # Test with input 123
9 print(sum_of_digits(123))
```

Below the code, the terminal shows the output of running the script:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
True
False
False
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
True
False
False
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'
6
○ PS C:\Users\shash> [ ]
```

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