

NAME:A.SHASHIDHAR H.NO: 2303A51798 BATCH:26

## ASSIGNMENT-3.3

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Rishabh Mittal	
<b>Instructor(s) Name</b>		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	
<b>Course Code</b>	23CS002PC304	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	III/I	<b>Regulation</b>	R23
<b>Date and Day of Assignment</b>	Week 2 - Wednesday	<b>Time(s)</b>	23CSBTB01 To 23CSBTB52
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	All batches
<b>Assignment Number:</b> 3.3(Present assignment number)/24(Total number of assignments)			

Q.No.	Question	Expected Time to complete
1	<p><b>Lab 3: Application for TGNPDCL – Electricity Bill Generation Using Python &amp; AI Tools</b></p> <p><b>Lab Objectives</b></p> <ul style="list-style-type: none"> <li>• To design a real-world electricity billing application using Python</li> <li>• To use AI-assisted coding tools for logic generation and optimization</li> <li>• To understand conditional logic and arithmetic operations</li> <li>• To generate structured billing output similar to utility bills</li> </ul> <p><b>Lab Outcomes (LOs)</b> After completing this lab, students will be able to:</p>	Week2 - Wednesday

## ASSIGNMENT-3.3

- Read and validate user input in Python
- Apply conditional logic for tariff-based billing
- Use AI tools to assist in program development
- Calculate and display electricity bill components
- Build a complete real-time application

### Task 1: AI-Generated Logic for Reading Consumer Details

#### Scenario

An electricity billing system must collect accurate consumer data.

#### Task Description

Use an AI tool (GitHub Copilot / Gemini) to generate a Python program that:

- Reads:
  - Previous Units (PU)
  - Current Units (CU)
  - Type of Customer
- Calculates units consumed
- Implements logic directly in the main program (no functions)

#### Expected Output

- Correct input reading
- Units consumed calculation
- Screenshot showing AI-generated code
- Sample input and output

```

C:\Users> shash > AAC A (3.3).py > ...
1 # Simple electricity consumer data reader and units calculator
2
3 # Read inputs
4 previous_units = float(input("Enter Previous Units (PU): "))
5 current_units = float(input("Enter Current Units (CU): "))
6 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")
7
8 # Calculate units consumed
9 units_consumed = current_units - previous_units
10
11 # Output the result
12 print(f"Units Consumed: {units_consumed}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

conda : The term 'conda' is not recognized as the name of a cmdlet, function, script file, or oper
+ conda activate Shashidhar
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\shash> & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\vs
231' > 'c:\Users\shash\AMC A (3.3).py'
Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0

```

### Task 2: Energy Charges Calculation Based on Units Consumed

#### Scenario

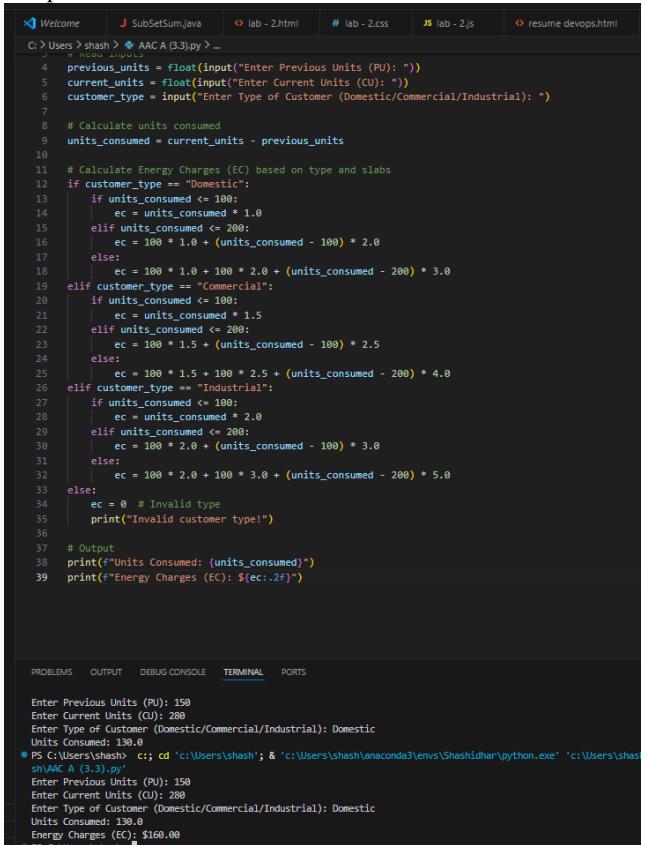
Energy charges depend on the number of units consumed and customer type.

#### Task Description

Review the AI-generated code from Task 1 and extend it to:

- Calculate **Energy Charges (EC)**
- Use conditional statements based on:
  - Domestic
  - Commercial
  - Industrial consumers
- Improve readability using AI prompts such as:

## ASSIGNMENT-3.3

	<ul style="list-style-type: none"> <li>○ "Simplify energy charge calculation logic"</li> <li>○ "Optimize conditional statements"</li> </ul> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>• Correct EC calculation</li> <li>• Clear conditional logic</li> <li>• Original and improved versions (optional)</li> <li>• Sample execution results</li> </ul>  <pre> 1 Welcome          J SubSetSum.java   O lab - 2.html    # lab - 2.css   JS lab - 2.js   O resume devops.html 2 3 C:&gt; Users &gt; shash &gt; AAC A (3.3).py &gt; ... 4 5 previous_units = float(input("Enter Previous Units (PU): ")) 6 current_units = float(input("Enter Current Units (CU): ")) 7 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ") 8 9 # Calculate units consumed 10 units_consumed = current_units - previous_units 11 12 # Calculate Energy Charges (EC) based on type and slabs 13 if customer_type == "Domestic": 14     if units_consumed &lt;= 100: 15         ec = units_consumed * 1.0 16     elif units_consumed &lt;= 200: 17         ec = 100 * 1.0 + (units_consumed - 100) * 2.0 18     else: 19         ec = 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0 20 elif customer_type == "Commercial": 21     if units_consumed &lt;= 100: 22         ec = units_consumed * 1.5 23     elif units_consumed &lt;= 200: 24         ec = 100 * 1.5 + (units_consumed - 100) * 2.5 25     else: 26         ec = 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0 27 elif customer_type == "Industrial": 28     if units_consumed &lt;= 100: 29         ec = units_consumed * 2.0 30     elif units_consumed &lt;= 200: 31         ec = 100 * 2.0 + (units_consumed - 100) * 3.0 32     else: 33         ec = 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0 34 else: 35     ec = 0 # Invalid type 36     print("Invalid customer type!") 37 38 # Output 39 print("Units Consumed: {units_consumed}") 40 print("Energy Charges (EC): \${ec:.2f}") </pre> <p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS</p> <pre> Enter Previous Units (PU): 150 Enter Current Units (CU): 200 Enter Type of Customer (Domestic/Commercial/Industrial): Domestic Units Consumed: 150.0 PS C:\Users\shash&gt; c:; cd 'c:\Users\shash'; &amp; 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\AAC A (3.3).py' Enter Previous Units (PU): 150 Enter Current Units (CU): 200 Enter Type of Customer (Domestic/Commercial/Industrial): Domestic Units Consumed: 150.0 Energy Charges (EC): \$100.00 PS C:\Users\shash&gt; </pre>	
	<p><b>Task 3: Modular Design Using AI Assistance (Using Functions)</b></p> <p><b>Scenario</b></p> <p>Billing logic must be reusable for multiple consumers.</p> <p><b>Task Description</b></p> <p>Use AI assistance to generate a Python program that:</p> <ul style="list-style-type: none"> <li>• Uses user-defined functions to: <ul style="list-style-type: none"> <li>○ Calculate Energy Charges</li> <li>○ Calculate Fixed Charges</li> </ul> </li> <li>• Returns calculated values</li> <li>• Includes meaningful comments</li> </ul> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>• Function-based Python program</li> <li>• Correct EC and FC values</li> <li>• Screenshots of AI-assisted function generation</li> <li>• Test cases with outputs</li> </ul>	

```
C:\> Users > shash > AAC A (3.3).py > ...
1 # Modular Electricity Billing System
2
3 def calculate_energy_charges(customer_type, units_consumed):
4     """
5         Calculate Energy Charges based on customer type and units consumed.
6         Slabs: Domestic (1/2/3), Commercial (1.5/2.5/4), Industrial (2/3/5) per unit tiers.
7     """
8     if customer_type == "Domestic":
9         if units_consumed <= 100:
10             return units_consumed * 1.0
11         elif units_consumed <= 200:
12             return 100 * 1.0 + (units_consumed - 100) * 2.0
13         else:
14             return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
15     elif customer_type == "Commercial":
16         if units_consumed <= 100:
17             return units_consumed * 1.5
18         elif units_consumed <= 200:
19             return 100 * 1.5 + (units_consumed - 100) * 2.5
20         else:
21             return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
22     elif customer_type == "Industrial":
23         if units_consumed <= 100:
24             return units_consumed * 2.0
25         elif units_consumed <= 200:
26             return 100 * 2.0 + (units_consumed - 100) * 3.0
27         else:
28             return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29     return 0 # Invalid type
30
31 def calculate_fixed_charges(customer_type):
32     """
33         Calculate Fixed Charges based on customer type.
34         Domestic: $100, Commercial: $200, Industrial: $300.
35     """
36     if customer_type == "Domestic":
37         return 100.0
38     elif customer_type == "Commercial":
39         return 200.0
40     elif customer_type == "Industrial":
41         return 300.0
42     return 0 # Invalid type
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
shVAC A (3.3).py*
Units Consumed: 130.0
Energy Charges (EC): $160.00
PS C:\Users\shash> c; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\shVAC A (3.3).py'
Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0
Energy Charges (EC): $160.00
Fixed Charges (FC): $100.00
PS C:\Users\shash> |
```

```
Welcome J SubSetSum.java D lab - 2.html # lab - 2.css JS lab - 2.js O resume dev
C:\> Users > shash > AAC A (3.3).py > ...
3 def calculate_energy_charges(customer_type, units_consumed):
25         elif units_consumed <= 200:
26             return 100 * 2.0 + (units_consumed - 100) * 3.0
27         else:
28             return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29     return 0 # Invalid type
30
31 def calculate_fixed_charges(customer_type):
32     """
33         Calculate Fixed Charges based on customer type.
34         Domestic: $100, Commercial: $200, Industrial: $300.
35     """
36     if customer_type == "Domestic":
37         return 100.0
38     elif customer_type == "Commercial":
39         return 200.0
40     elif customer_type == "Industrial":
41         return 300.0
42     return 0 # Invalid type
43
44 # Main program
45 previous_units = float(input("Enter Previous Units (PU): "))
46 current_units = float(input("Enter Current Units (CU): "))
47 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")
48
49 units_consumed = current_units - previous_units
50 ec = calculate_energy_charges(customer_type, units_consumed)
51 fc = calculate_fixed_charges(customer_type)
52
53 print(f"Units Consumed: {units_consumed}")
54 print(f"Energy Charges (EC): ${ec:.2f}")
55 print(f"Fixed Charges (FC): ${fc:.2f}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\shash> c; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\shAAC A (3.3).py'
Fixed Charges (FC): $100.00
PS C:\Users\shash> c; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\shAAC A (3.3).py'
Enter Previous Units (PU): 0
Enter Current Units (CU): 250
Enter Type of Customer (Domestic/Commercial/Industrial): Commercial
Units Consumed: 250.0
Energy Charges (EC): $500.00
Fixed Charges (FC): $200.00
PS C:\Users\shash>
```

## ASSIGNMENT-3.3

**Task 4: Calculation of Additional Charges****Scenario**

Electricity bills include multiple additional charges.

**Task Description**

Extend the program to calculate:

- **FC** – Fixed Charges
  - **CC** – Customer Charges
  - **ED** – Electricity Duty (percentage of EC)
- Use AI prompts like:
- “*Add electricity duty calculation*”
  - “*Improve billing accuracy*”

**Expected Output**

- Individual charge values printed
- Correct duty calculation
- Well-structured output
- Verified intermediate results

```
C:\Users\shash> & AAC A (3.3).py ...
1 # Extended Electricity Billing with Additional Charges
2
3 def calculate_energy_charges(customer_type, units_consumed):
4 """
5 Calculate Energy Charges based on customer type and units consumed.
6 Slabs: Domestic (1/2/3), Commercial (1.5/2.5/4), Industrial (2/3/5) per unit tiers.
7 """
8 if customer_type == "Domestic":
9     if units_consumed <= 100:
10         return units_consumed * 1.0
11     elif units_consumed <= 200:
12         return 100 * 1.0 + (units_consumed - 100) * 2.0
13     else:
14         return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
15 elif customer_type == "Commercial":
16     if units_consumed <= 100:
17         return units_consumed * 1.5
18     elif units_consumed <= 200:
19         return 100 * 1.5 + (units_consumed - 100) * 2.5
20     else:
21         return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
22 elif customer_type == "Industrial":
23     if units_consumed <= 100:
24         return units_consumed * 2.0
25     elif units_consumed <= 200:
26         return 100 * 2.0 + (units_consumed - 100) * 3.0
27     else:
28         return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29 return 0 # Invalid type
30
31 def calculate_fixed_charges(customer_type):
32 """
33 Calculate Fixed Charges based on customer type.
34 Domestic: $100, Commercial: $200, Industrial: $300.
35 """
36 if customer_type == "Domestic":
37     return 100.0
38 elif customer_type == "Commercial":
39     return 200.0
40 elif customer_type == "Industrial":
41     return 300.0
42 return 0 # Invalid type
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
conda : The term 'conda' is not recognized as the name of a cmdlet, function, script file, or operable program
+ FullyQualifiedErrorId : CommandNotFound
Exception
● PS C:\Users\shash> & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extension
Enter Previous Units (PU): 150
Enter Current Units (CU): 200
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 150.0
Energy Charges (EC): $150.00
Fixed Charges (FC): $100.00
Customer Charges (CC): $50.00
Electricity Duty (ED): $16.00
○ PS C:\Users\shash>
```

## ASSIGNMENT-3.3

	<p>The screenshot shows a Python script named <code>lab - 2.py</code> in a code editor. The code defines functions for calculating fixed charges based on customer type, energy charges, and electricity duty. It then prompts the user for previous and current units, and the type of customer (Domestic, Commercial, Industrial). It calculates the total bill by summing up energy charges, fixed charges, and electricity duty. The terminal output shows a sample run where the user inputs 150 for previous units, 200 for current units, and "Domestic" for customer type. The output displays the individual charges (units consumed, energy charges, fixed charges, customer charges, and electricity duty) and the total bill amount.</p>
	<p><b>Task 5: Final Bill Generation and Output Analysis</b></p> <p><b>Scenario</b></p> <p>The final electricity bill must present all values clearly.</p> <p><b>Task Description</b></p> <ul style="list-style-type: none"> <li>Develop the final Python application to:</li> <li>Calculate total bill:</li> <li>Total Bill = EC + FC + CC + ED</li> <li>Display: <ul style="list-style-type: none"> <li>Energy Charges (EC)</li> <li>Fixed Charges (FC)</li> <li>Customer Charges (CC)</li> <li>Electricity Duty (ED)</li> <li>Total Bill Amount</li> </ul> </li> <li>Analyze the program based on: <ul style="list-style-type: none"> <li>Accuracy</li> <li>Readability</li> <li>Real-world applicability</li> </ul> </li> </ul> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>Complete electricity bill output</li> <li>Neatly formatted display</li> <li>Sample input/output</li> <li>Short analysis paragraph</li> </ul>

NAME:A.SHASHIDHAR H.NO: 2303A51798 BATCH:26

### ASSIGNMENT-3.3

```
C:\> Users > shash > AAC A (3.3).py > ...
1  # Final Electricity Bill Generator
2
3  def calculate_energy_charges(customer_type, units_consumed):
4      """
5          Calculate Energy Charges based on customer type and units consumed.
6          Slabs: Domestic (1/2/3), Commercial (1.5/2.5/4), Industrial (2/3/5) per unit tiers.
7      """
8      if customer_type == "Domestic":
9          if units_consumed <= 100:
10              return units_consumed * 1.0
11          elif units_consumed <= 200:
12              return 100 * 1.0 + (units_consumed - 100) * 2.0
13          else:
14              return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
15      elif customer_type == "Commercial":
16          if units_consumed <= 100:
17              return units_consumed * 1.5
18          elif units_consumed <= 200:
19              return 100 * 1.5 + (units_consumed - 100) * 2.5
20          else:
21              return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
22      elif customer_type == "Industrial":
23          if units_consumed <= 100:
24              return units_consumed * 2.0
25          elif units_consumed <= 200:
26              return 100 * 2.0 + (units_consumed - 100) * 3.0
27          else:
28              return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29      return 0
30
31  def calculate_fixed_charges(customer_type):
32      """Fixed Charges: Domestic $100, Commercial $200, Industrial $300."""
33      if customer_type == "Domestic":
34          return 100.0
35      elif customer_type == "Commercial":
36          return 200.0
37      elif customer_type == "Industrial":
38          return 300.0
39      return 0
40
41  def calculate_customer_charges():
42      """Final Customer Charges: ECA """
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

● PS C:\Users\shash> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\AAC A (3.3).py'
Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic

=====
ELECTRICITY BILL SUMMARY
=====
Customer Type: Domestic
Units Consumed: 130.0
Energy Charges (EC): \$160.00
Fixed Charges (FC): \$100.00

NAME:A.SHASHIDHAR H.NO: 2303A51798 BATCH:26  
ASSIGNMENT-3.3

The screenshot shows a Jupyter Notebook interface with a single code cell. The code defines functions for calculating fixed charges, customer charges, and electricity duty, and then calculates a total bill based on user input for previous and current units and customer type. The output terminal shows the execution of the code and the resulting bill summary.

```
C:\Users> shash > AAC A (3.3).py >_
51     def calculate_fixed_charges(customer_type):
52         return 0
53
54     def calculate_customer_charges():
55         """Fixed Customer Charges: $50."""
56         return 50.0
57
58     def calculate_electricity_duty(ec):
59         """Electricity Duty: 10% of EC."""
60         return 0.10 * ec
61
62 # Main program
63 previous_units = float(input("Enter Previous Units (PU): "))
64 current_units = float(input("Enter Current Units (CU): "))
65 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip()
66
67 if customer_type not in ["Domestic", "Commercial", "Industrial"]:
68     print("Invalid type! Defaulting to Domestic.")
69     customer_type = "Domestic"
70
71 units_consumed = current_units - previous_units
72 ec = calculate_energy_charges(customer_type, units_consumed)
73 fc = calculate_fixed_charges(customer_type)
74 cc = calculate_customer_charges()
75 ed = calculate_electricity_duty(ec)
76 total_bill = ec + fc + cc + ed
77
78 # Neatly formatted bill display
79 print("\n" + "="*40)
80 print("ELECTRICITY BILL SUMMARY")
81 print("="*40)
82 print(f"Customer Type: {customer_type}")
83 print(f"Units Consumed: {units_consumed}")
84 print(f"Energy Charges (EC): ${ec:.2f}")
85 print(f"Fixed Charges (FC): ${fc:.2f}")
86 print(f"Customer Charges (CC): ${cc:.2f}")
87 print(f"Electricity Duty (ED): ${ed:.2f}")
88 print("-"*40)
89 print(f"TOTAL BILL AMOUNT: ${total_bill:.2f}")
90 print("="*40)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash> c:; cd "c:\Users\shash" & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\AAC A (3.3).py'
=====
Customer Type: Domestic
Units Consumed: 130.0
Energy Charges (EC): $160.00
Fixed Charges (FC): $100.00
Customer Charges (CC): $50.00
Electricity Duty (ED): $16.00
-----
TOTAL BILL AMOUNT: $326.00
=====
```

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

NAME:A.SHASHIDHAR

H.NO:2303A51798

BATCH:26

## ASSIGNMENT – 3.4

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Rishabh Mittal	
<b>Instructor(s) Name</b>		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	
<b>CourseCode</b>	23CS002PC304	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	III/II	<b>Regulation</b>	R23
<b>Date and Day of Assignment</b>	Week2	<b>Time(s)</b>	23CSBTB01 To 23CSBTB52
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	All batches
<b>Assignment Number: 3.4 (Present assignment number)/24(Total number of assignments)</b>			
<b>Q.No.</b>	<b>Question</b>		<b>Expected Time to complete</b>
1	Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and		Week2

	<p>Few-shot Techniques</p> <p><b>Task 1: Zero-shot Prompt – Fibonacci Series Generator</b></p> <p><b>Task Description #1</b></p> <ul style="list-style-type: none"><li>Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers.</li></ul> <p><b>Expected Output #1</b></p> <ul style="list-style-type: none"><li>A complete Python function generated by Copilot without any example provided.</li><li>Correct output for sample input <math>N = 7 \rightarrow 0 1 1 2 3 5 8</math></li><li>Observation on how Copilot understood the instruction with zero context.</li></ul>	
--	--	--

```
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)
```

The screenshot shows a terminal window with two distinct sections. The top section is a code editor with the following Python code:

```
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)
```

The bottom section is a terminal window with a PowerShell prompt (PS) and the following output:

```
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\py3\python' -m pip install --upgrade pip
TOTAL BILL AMOUNT: $326.00
=====
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\py3\python' -m pip install --upgrade pip
● PS C:\Users\shash> 7
7
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\py3\python' -m pip install --upgrade pip
0 1 2 3 5 8
● PS C:\Users\shash> c;; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\py3\python' -m pip install --upgrade pip
[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' [3, 2, 1]
● 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
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' True
False
False
● 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:

```
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**