Konda ShivaPrasad 2303A51200

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
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	**Week1 - Friday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
Assignment Number:1.3(Present assignment number)/**24**(Total number of assignments)			

	Question	***Expected Time to compl***

			ete
1	Lab 1: Environment Setup – *GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow* **Lab Objectives:** ❖ To install and configure GitHub Copilot in Visual Studio Code. ❖ To explore AI-assisted code generation using GitHub Copilot. ❖ To analyze the accuracy and effectiveness of Copilot's code suggestions. ❖ To understand prompt-based programming using comments and code context **Lab Outcomes (LOs):** After completing this lab, students will be able to: ❖ Set up GitHub Copilot in VS Code successfully. ❖ Use inline comments and context to generate code with Copilot. ❖ Evaluate AI-generated code for correctness and readability. ❖ Compare code suggestions based on different prompts and programming styles.	Wee k1 - Mon day
	Task 0 ❖ Install and configure GitHub Copilot in VS Code. Take screenshots of each step. Expected Output ❖ Install and configure GitHub Copilot in VS Code. Take screenshots of each step.	
	Task 1: AI-Generated Logic Without Modularization (String Reversal Without Functions) ❖ **Scenario** You are developing a **basic text-processing utility** for a messaging application. ❖ **Task Description** Use GitHub Copilot to generate a Python program that: ➢ Reverses a given string ➢ Accepts user input ➢ Implements the logic directly in the main code ➢ Does not use any user-defined functions	

❖ **Expected Output**

- Correct reversed string
- Screenshots showing Copilot-generated code suggestions
- Sample inputs and outputs
- **Output**
- **Prompt : #generate python code string reversal without function**
- **Code:**

```
#generate python code String Reversal Without Functions

input_string = "Hello, World!"
reversed_string = ""
for i in range(len(input_string) - 1, -1, -1):
    reversed_string += input_string[i]
print(reversed_string)
```

-

```
PS C:\Ai_coding> & C:/Users/mukes/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Ai_coding/day
_1.py
!dlroW ,olleH
PS C:\Ai_coding>
```

Explanation

1.An empty string `reversed_string` is created to store the reversed result.
2. A `for` loop runs from the last character of `input_string` to the first and appends each character.
3. The final reversed string `!dlroW ,olleH` is printed as output.

Task 2: Efficiency & Logic Optimization (Readability Improvement)

❖ **Scenario**

The code will be reviewed by other developers.

❖ **Task Description**

Examine the Copilot-generated code from **Task 1** and improve it by:

- Removing unnecessary variables
- Simplifying loop or indexing logic
- Improving readability
- Use Copilot prompts like:
  - *"Simplify this string reversal code"*
  - *"Improve readability and efficiency"*

Hint:
Prompt Copilot with phrases like

"optimize this code", *"simplify logic"*, or *"make it more readable"*

- **Expected Output**
  - Original and optimized code versions
  - Explanation of how the improvements reduce time complexity

---

- **OUTPUT**
- **PROMPT : #generate python code for string reversal without function optimized code**
- **Code**

```
#generate python code String Reversal Without Functions optimized cod
input_string = "Hello, World!"
reversed_string = input_string[::-1]
print(reversed_string)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Ai_coding> & C:/Users/mukes/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Ai_cod
_1.py
!dlroW ,olleH
PS C:\Ai_coding>
```

- **Explanation**

- The string slicing `[::-1]` creates a reversed copy of `input_string` by stepping backwards.
- This method uses Python's built-in slicing, making the code short and efficient.
- The reversed output `!dlroW ,olleH` is printed directly.

Task 3: Modular Design Using AI Assistance (String Reversal Using Functions)

- **Scenario**
  The string reversal logic is needed in **multiple parts** of an application.

- **Task Description**
  Use GitHub Copilot to generate a function-based Python program that:
  - Uses a user-defined function to reverse a string
  - Returns the reversed string
  - Includes meaningful comments (AI-assisted)

- **Expected Output**
  - Correct function-based implementation
  - Screenshots documenting Copilot's function generation
  - Sample test cases and outputs
  - **OUTPUT**

- **PROMPT #generate python code string reversal with function**
- **Code**

➢

```python
#generate python code String Reversal (Using Functions)
def reverse_string(s):
    """Reverses the given string."""
    return s[::-1]

input_string = "Hello, World!"
reversed_string = reverse_string(input_string)
print(reversed_string)  # Output: !dlroW ,olleH
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Ai_coding> & C:/Users/mukes/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Ai_codi
_1.py
!dlroW ,olleH
PS C:\Ai_coding>
```

Explanation

The function `reverse_string(s)` reverses a string using Python slicing `[::-1]`.

When `"Hello, World!"` is passed to the function, it prints the reversed output `!dlroW ,olleH`.

Task 4: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

❖ **Scenario**

You are asked to justify design choices during a code review.

❖ **Task Description**

Compare the Copilot-generated programs:

- Without functions (Task 1)
- With functions (Task 3)

Analyze them based on:

- Code clarity
- Reusability
- Debugging ease
- Suitability for large-scale applications

❖ **Expected Output**

Comparison table or short analytical report

Output

Comparison of String Reversal Programs

Aspect	Without Functions (Task 1)	With Functions (Task 3)
Code Structure	Logic is written directly in the main program	Logic is separated into a user-defined function
Code Clarity	Less clear when logic grows	More clear and organized
Readability	Harder to understand for others	Easy to read due to meaningful function name
Reusability	Cannot reuse the logic easily	Function can be reused multiple times
Debugging	Difficult to debug as logic is mixed	Easy to debug by testing the function separately
Scalability	Not suitable for large applications	Suitable for large and complex applications
Maintenance	Changes must be done everywhere	Changes done once inside the function
Best Use Case	Small, simple programs	Medium to large applications

Explanation

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal)

- **Scenario**
  Your mentor wants to evaluate how AI handles **alternative logic paths**.

- **Task Description**
  Prompt GitHub Copilot to generate:
  - A **loop-based** string reversal approach
  - A **built-in / slicing-based** string reversal approach

- **Expected Output**
  - Two correct implementations
  - Comparison discussing:
    - Execution flow
    - Time complexity
    - Performance for large inputs
    - When each approach is appropriate

OUTPUT

PROMPT #generate python code for different algorithm approaches to string reversal

Code

```python
def reverse_string_slicing(s):
    """Reverse a string using slicing."""
    return s[::-1]

def reverse_string_loop(s):
    """Reverse a string using a loop."""
    reversed_str = ''
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str

def reverse_string_recursion(s):
    """Reverse a string using recursion."""
    if len(s) == 0:
        return s
    else:
        return s[-1] + reverse_string_recursion(s[:-1])

def reverse_string_builtin(s):
    """Reverse a string using built-in functions."""
    return ''.join(reversed(s))

# Example usage
if __name__ == "__main__":
    test_string = "Hello, World!"
    print("Original String:", test_string)
    print("Reversed (Slicing):", reverse_string_slicing(test_string))
    print("Reversed (Loop):", reverse_string_loop(test_string))
    print("Reversed (Recursion):", reverse_string_recursion(test_string))
    print("Reversed (Built-in):", reverse_string_builtin(test_string))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Ai_coding> & C:/Users/mukes/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Ai_coding/day
_1.py
Original String: Hello, World!
Reversed (Slicing): !dlroW ,olleH
Reversed (Loop): !dlroW ,olleH
Reversed (Recursion): !dlroW ,olleH
Reversed (Built-in): !dlroW ,olleH
PS C:\Ai_coding>
```

Explanation

This program shows **four ways to reverse a string**: slicing, loop, recursion, and built-in functions.

Each method uses a different logic but produces the **same reversed output**.

The main block tests all methods using the string `"Hello, World!"`.

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.