**SR UNIVERSITY**

**AI ASSISTED CODING**

**ASSIGNMENT 11.4**

**NAME:** MOHAMMED MUTAKABIR HUSSAIN

**H.NO:** 2503A51L35

**BATCH:** 20

**TASK1**

**TASK1 DESCRIPTION:-**

Use AI to help implement a **Stack** class in Python with the following operations: push (), pop (), peek (), and is empty ().

**Instructions**:

* + Ask AI to generate code skeleton with docstrings.
  + Test stack operations using sample data.
  + Request AI to suggest optimizations or alternative implementations (e.g., using collections. Deque).

**PROMPT:**  
A screenshot of a black screen

AI-generated content may be incorrect.  
  
  
  
  
  
  
  
  
  
  
**CODE:**  
A screen shot of a computer program

AI-generated content may be incorrect.  
A screen shot of a computer program

AI-generated content may be incorrect.  
A screen shot of a computer code

AI-generated content may be incorrect.  
**OUTPUT:**  
A screen shot of a computer

AI-generated content may be incorrect.  
  
  
  
**OBSERVATION:**   
cursor ai generated python code with operations like push , pop, peek, and is empty , also generated docstrings so that it would be easy to understand each block of code also implemented stack operations with a sample data

**TASK2**

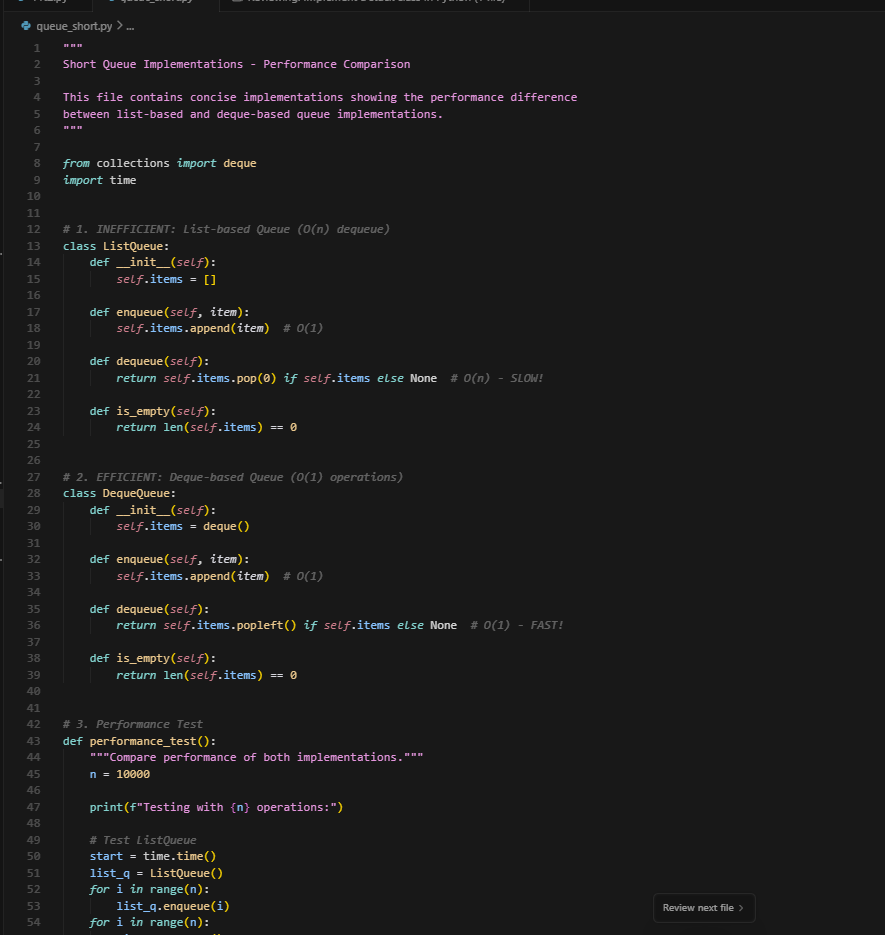
**TASK2 DESCRIPTION:-**

Implement a **Queue** with enqueue (), dequeue (), and is empty () methods.

* **Instructions**:
  + First, implement using Python lists.
  + Then, ask AI to review performance and suggest a more efficient implementation (using collections. Deque).

**PROMPT:**  
A screenshot of a computer program

AI-generated content may be incorrect.

**CODE:**  
  
A screen shot of a computer program

AI-generated content may be incorrect.

**OUTPUT:**  
A screenshot of a computer program

AI-generated content may be incorrect.

**OBSERVATION:**  
cursor ai generated a queue with enqueue(), dequeue(), and is empty() methods using the python lists and also suggested a more efficient implementation using collections , deque

**TASK3**

**TASK3 DESCRIPTION:-**

Implement a **Singly Linked List** with operations: instated (), delete value (), and traverse ().

* **Instructions**:
  + Start with a simple class-based implementation (Node, LinkedList).
  + Use AI to generate inline comments explaining pointer updates (which are non-trivial).
  + Ask AI to suggest test cases to validate all operations.

**PROMPT:**  
A screenshot of a computer

AI-generated content may be incorrect.

**CODE:**  
A screenshot of a computer screen

AI-generated content may be incorrect.  
A screenshot of a computer program

AI-generated content may be incorrect.

**OUTPUT:**  
A black screen with white text

AI-generated content may be incorrect.

**OBSERVATION:**  
cursor ai generated **Singly Linked List** with operations instated (), delete value (), and traverse ().  
started with simple class based implementation (Node, LinkedList)  
also generated inline comments explaining pointer updates (which are non-trivial).

**TASK4**

**TASK4 DESCRIPTION:-**

Implement a **Binary Search Tree** with methods for insert (), search (), and inorder\_traversal ().

* **Instructions**:
  + Provide AI with a partially written Node and BST class.
  + Ask AI to complete missing methods and add docstrings.

Test with a list of integers and compare outputs of search () for present vs absent elements.

**PROMPT:**  
A screenshot of a black box

AI-generated content may be incorrect.

**CODE:**  
A screen shot of a computer

AI-generated content may be incorrect.  
A screenshot of a computer program

AI-generated content may be incorrect.

**OUTPUT:**  
A screen shot of a computer

AI-generated content may be incorrect.

**OBSERVATION:**

Ai generated a binary search tree with methods for insert (), search (), and inorder\_traversal ().  
using a partially written node and BST class and also completed missing methods and added docstrings

**TASK5**

**TASK5 DESCRIPTION:-**

Implement a **Graph** using an adjacency list, with traversal methods BFS () and DFS ().

* **Instructions**:
  + Start with an adjacency list dictionary.
  + Ask AI to generate BFS and DFS implementations with inline comments.
  + **Compare recursive vs iterative DFS if suggested by AI.**

**PROMPT:**  
A screenshot of a black and white screen

AI-generated content may be incorrect.

**CODE:**  
A screenshot of a computer program

AI-generated content may be incorrect.  
A computer screen with text and images

AI-generated content may be incorrect.

**OUTPUT:**  
A screen shot of a computer

AI-generated content may be incorrect.

**OBSERVATION:**  
ai has generated a graph using a adjacency list, with traversal methods BFS () and DFS ().  
and alos gave us implementations with inline comments