**LAB ASSIGNMENT-11.2**

**NAME: Bollam Sathvika**

**HTNO:2403A51344**

**BATCH:14**

**TASK1:**

**Prompt:**

Implement a HashTable class in Python with insert, search, and delete methods.

Use chaining (lists) to handle collisions.

Include comments explaining each method.

**Code:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

**Observation:**

In this task, I used AI assistance (GitHub Copilot) to implement a Hash Table in Python with basic operations such as insert, search, and delete. The AI automatically generated a class structure and suggested well-commented methods for each operation.  
It also implemented collision handling using chaining, where each index in the hash table stores a list of key-value pairs. During testing, I observed that the insert, search, and delete operations worked correctly, and the AI-generated code efficiently handled collisions by maintaining separate lists at each bucket.  
Using AI made it easier to understand the logic of hashing and collision resolution while saving time in writing repetitive code.

**TASK2:**

**Prompt:**

Implement a Graph class using an adjacency list.

Include methods to add vertices, add edges, and display connections.

**Code:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Observation:**

In this task, I used GitHub Copilot to implement a Hash Table with basic operations: insert, search, and delete. The AI suggested a proper class structure and well-commented methods. It also handled collisions using chaining, storing multiple key-value pairs in each bucket. During testing, all operations worked correctly, and collisions were managed efficiently. Using AI helped me understand hashing and collision handling faster while reducing the effort of writing repetitive code.

**TASK3:**

**Prompt:**

Implement a PriorityQueue class using Python's heapq module.

Include methods: enqueue (with priority), dequeue (highest priority), and display.

**Code:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**TASK4:**

**Prompt:**

Implement a DequeDS class using collections.deque.

Include methods to insert and remove from both ends.

Add docstrings to explain each method.

**Code:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Observation:**

In this task, I used AI assistance (GitHub Copilot) to implement a double-ended queue (Deque) using Python’s collections.deque. The AI suggested a class structure with methods to insert and remove elements from both ends and added proper docstrings for each method. During testing, all operations worked correctly from both the front and rear of the deque. Using AI helped me quickly understand how a deque works and made implementing the methods faster and more organized.