**MONICA B (SUPERSET ID - 5008627)**

**Exercise 1: Inventory Management System**

1. Explain why data structures and algorithms are essential in handling large inventories?

Data structures and algorithms are critical for effectively organizing and retrieving data in vast inventories. They optimize tasks such as searching, sorting, and updating to keep the system fast and scalable, resulting in accurate inventory management and better performance.

1. Discuss the types of data structures suitable for this problem.

1. **Arrays** are useful for simple and static collections of elements that require easy access via index.

2.**Linked lists** are suitable for dynamic inventories with frequent insertions and deletions.

3.**Hash tables** are ideal for rapid retrieval and insertion operations. They are especially handy for indexing inventory objects with unique keys like product IDs.

4.**Trees**: Structures like binary search trees or balanced trees are effective for sorted data, allowing quick search, insert, and delete operations.

5.**Graphs** are useful for describing complex inventory relationships, such as dependencies or networked objects.

6.**Heaps** are useful for prioritizing inventory and gaining quick access to the highest or lowest priority item.

7.**Tries:** Suitable for prefix-based searches; useful in applications that need searching for things using a common prefix or category.

1. Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.

**HashMap**:

**Add**: O(1) - on average

**Update**: O(1) - on average

**Delete**: O(1) - on average

1. Discuss how you can optimize these operations.

To maximize performance in an inventory management system utilizing 'HashMap', consider these strategies: First, use an efficient hash function to distribute entries equally across the table, reducing collisions and preserving an average time complexity of (O(1). Handle collisions effectively by implementing solutions such as chaining, which stores colliding entries in linked lists, or open addressing, which searches for alternative placements, guaranteeing that the 'HashMap' remains efficient as it increases.