HashMart: Efficient Stock & Order Management

Project Overview

- **Perform CRUD operations on products:** Create new items, read (view) current stock, update quantities or prices, and delete items when they are no longer sold.

- **Handle customer orders by priority:** Accept orders tagged with a “priority” (lower number means more urgent), process them in the correct order, and automatically adjust inventory levels.

Users interact through a menu, choosing one of these options each time:

- **Create or Update an Item** (enter ID, name, quantity, price)
- **Delete an Item** by ID
- **Read (Display) All Inventory Items**
- **Place a New Order** (enter item ID, quantity, priority)
- **Process the Next (Most Urgent) Order**
- **Show All Pending Orders**
- **Search an Item by ID**
- **Exit**

When processing an order, the system:

- Checks if the requested item exists.
- If stock ≥ requested quantity, it reduces the quantity and reports “Order fulfilled.”
- If stock < requested quantity, it sells whatever remains, reports the partial fulfillment, and leaves any shortfall unfulfilled.
- If an item’s quantity drops to zero, it is automatically deleted from inventory.

Data Structures Used

1. **Doubly Linked List**
   - Manages inventory in insertion order.
   - Nodes store product info and are connected via `prev` and `next`.

2. **Hash Table (unordered_map)**
   - Provides O(1) average-time lookup by product ID.
   - Used for quick updates, deletions, and lookups of inventory items.

3. **Priority Queue (Min-Heap)**
   - Manages customer orders by urgency using custom comparator.
   - Ensures urgent orders are processed first.

4. **Heap (via priority_queue)**
   - Internal structure of priority queue is a binary heap for efficiency.

5. **Custom Comparator**
   - `CompareOrder` struct defines sorting rules for the priority queue.

6. **Pointers**
   - Used to link nodes in the doubly linked list and manage dynamic memory.

7. **Dynamic Memory Allocation**
   - `new` and `delete` used to create and manage inventory items at runtime.

8. **Traversal Algorithms**
   - Linked list traversal from `head` to `nullptr` to display inventory.
   - Priority queue copy and pop used to show pending orders.

9. **Map Operations**
   - `insert`, `find`, `erase`, `count` operations used efficiently on the `unordered_map`.