CENG 241

LAB EXERCISES-1

The program defines a single **structure** that represents the store's dynamic stock list. It will hold a **1-dimensional dynamic array** for storing product stock quantities, along with two numbers (size and capacity) that describe how much of the array is used and how much memory is available.

- int *data; — a pointer to a **1-dimensional dynamic integer array** that stores the stock quantities of all products. Each element in this array represents the number of units available for a specific product.
- int size; — an integer value indicating the **current number of products** (the number of elements actually in use in the array).
- int capacity; — an integer value showing the **total number of elements the array can currently hold** before more memory must be allocated.

Together, these three fields enable the program to **store, expand, and manage** the list of product stock quantities dynamically during execution. For the inventory management implement the following functions:

1. **create:** Initializes an empty inventory with a given initial capacity; receives an initial capacity and a reference to the inventory; returns success/failure; allocates memory and sets size to zero so the array is ready to store product stock counts.
2. **destroy:** Safely closes the inventory; receives a reference to the inventory; returns nothing; frees all allocated memory and nulls pointers to prevent leaks after the program ends.
3. **reserve:** Pre-allocates shelf space for more products; receives a desired new capacity and the inventory; returns success/failure; uses reallocation to change capacity without altering current product count.
4. **append:** Adds a new product's stock count at the end; receives the inventory and an integer stock quantity; returns success/failure; grows capacity automatically if needed and increments the product count.
5. **insert_at:** Inserts a new product at a specific position; receives the inventory, a target index, and an initial stock quantity; returns success/failure; shifts later items to the right to keep order and increases size by one.
6. **delete_at:** Removes a product from a given position; receives the inventory and an index; returns success/failure; shifts remaining items left to fill the gap and decreases the size.
7. **find:** Locates the first product whose stock equals a target value; receives the inventory and an integer value; returns the first matching index or -1 if none; performs a linear scan.
8. **print:** Shows a snapshot of the inventory; receives the inventory; returns nothing; prints the list of stock quantities along with current size and capacity for quick audits.
9. **sort_asc:** Orders products from lowest to highest stock; receives the inventory; returns nothing; sorts the array to highlight low-stock items for replenishment.
10. **reverse:** Flips the current order of products; receives the inventory; returns nothing; in-place reversal that's handy after sorting to get a high-to-low view.
11. **stats:** Computes quick KPIs over stock levels; receives the inventory and output holders for min, max, and average; returns success/failure (fails if empty); reports the lowest stock, highest stock, and mean stock.
12. **show_size_capacity:** Reports inventory health; receives the inventory; returns nothing; prints the current number of products tracked and the allocated capacity so you can anticipate growth.

13. **print_menu:** Displays the operator menu; receives nothing; returns nothing; lists available actions for the menu-driven interface clearly

Sample Run:

```
=== Welcome to the Dynamic Stock Ledger System ===
Manage your store's product stock easily through the options below.
-----------------------------------------------------
1) Create new stock ledger
2) Add (append) a new product's stock
3) Insert product stock at specific position
4) Remove a product from inventory
5) Find product by stock quantity
6) Show current number of products and total capacity
7) Reverse product list
8) Display inventory statistics (min / max / average stock)
9) Adjust reserved capacity
10) Show all products' stock values
11) Sort inventory (ascending by stock)
0) Exit
-----------------------------------------------------
Enter your choice: 1
Enter initial stock capacity (number of products to prepare space for): 4
✅New inventory ledger created successfully!

Enter your choice: 2
Enter stock quantity for the new product: 25
✅Product added successfully.

Enter your choice: 2
Enter stock quantity for the new product: 12
✅Product added successfully.

Enter your choice: 2
Enter stock quantity for the new product: 40
✅Product added successfully.

Enter your choice: 2
Enter stock quantity for the new product: 18
✅Product added successfully.

Enter your choice: 10
 Current Stock List (size = 4 / capacity = 4):
[25, 12, 40, 18]

Enter your choice: 3
Enter position to insert the new product (0-based index): 1
Enter stock quantity for the product: 99
✅Product inserted successfully.
```

Enter your choice: 10
 Updated Stock List (size = 5 / capacity = 8):
[25, 99, 12, 40, 18]

Enter your choice: 5
Enter stock quantity to search for: 40
 Found product at index: 3

Enter your choice: 8
 Inventory Statistics:
Minimum stock = 12
Maximum stock = 99
Average stock = 38.8

Enter your choice: 11
Sorting inventory from lowest to highest stock...
✅Inventory sorted successfully.

Enter your choice: 10
 Sorted Stock List:
[12, 18, 25, 40, 99]

Enter your choice: 7
Reversing product order...
✅Stock list reversed.

Enter your choice: 10
 Reversed Stock List:
[99, 40, 25, 18, 12]

Enter your choice: 4
Enter index of product to remove: 2
✅Product removed successfully.

Enter your choice: 10
 Updated Stock List:
[99, 40, 18, 12]

Enter your choice: 9
Enter new capacity to reserve: 16
✅Inventory capacity updated to 16.