**Report: Inventory Management System**

**Purpose:** The purpose of the code is to create a simple inventory management system with user authentication, product management, and sales tracking. It allows users to add, view, update, delete, and search for products in an inventory. Additionally, it tracks sales, updates stock levels, and supports restocking products that are low in quantity. The system ensures secure user authentication and stores data persistently using text files.

**Importance:** This inventory management system is important for businesses or small enterprises to efficiently manage product inventories. It helps track stock levels, monitor sales, and ensure products are always available. The code provides an organized way to handle inventory data and automate tasks like updating stock, restocking, and recording sales, thus improving business operations and decision-making. The use of file handling ensures that data is saved and can be accessed even after program termination, making the system more reliable and practical.

**Features:-**

1. User Authentication:

* Sign Up: Allows new users to register by providing a username and password.
* Sign In: Validates existing users and ensures secure access to inventory data.

2. Product Management:

* Add Products: Enables adding new products to the inventory.
* Update Products: Allows modification of product details such as name, price, and quantity.
* Delete Products: Removes products from the inventory.

3. Inventory Operations:

* Search Products: Retrieves product details by ID.
* Sort Products: Organizes inventory based on price, quantity, or name.
* Sold Products: Tracks and updates the quantity of sold products.

4. Data Persistence:

* Inventory data is stored in text files specific to each user.
* Ensures data is saved across program sessions.

5. Interactive Console UI:

* Colored text output for better user experience.
* Clear menu-driven navigation.

6. File Handling:

* Ensuring data integrity during simultaneous read and write operations.
* Handling cases where files do not exist or are inaccessible.

7. Error Handling:

* Validating user input to prevent crashes caused by invalid data.
* Managing edge cases, such as attempting to delete a product that does not exist.

8. Sorting Algorithms:

* Implementing sorting based on multiple criteria while maintaining code simplicity.

9. User Authentication:

* Protecting user credentials and ensuring secure login processes.

Function Descriptions

**Function Descriptions**

**1. fileExists()**

* Checks if a specified file exists.
* **Prototype:** int fileExists();
* **Parameters:** filename: File name to check
* **Return Value:** 1 if file exists, 0 otherwise

**2. createFile()**

* Creates a new file with the specified name.
* **Prototype:** void createFile(char \*name);
* **Parameters:** name: File name to create
* **Return Value:** None

**3. sign\_in()**

* Handles user sign-in by verifying credentials and preparing the user's inventory file.
* **Prototype:** int sign\_in(struct Users user[], int \*user\_count);
* **Parameters:** user[]: Array of users, user\_count: Pointer to user count
* **Return Value:** 1 if sign-in successful, 0 otherwise

**4. sign\_up()**

* Allows a new user to register and sign in.
* **Prototype:** void sign\_up(struct Users user[], int \*user\_count);
* **Parameters:** user[]: Array of users, user\_count: Pointer to user count
* **Return Value:** None

**5. write\_product()**

* Adds a new product to the inventory, ensuring unique IDs.
* **Prototype:** void write\_product();
* **Parameters:** None
* **Return Value:** None

**6. read\_products()**

* Reads and displays products from the user's inventory file.
* **Prototype:** void read\_products();
* **Parameters:** None
* **Return Value:** None

**7. del\_product()**

* Deletes a product from the inventory based on its ID.
* **Prototype:** void del\_product();
* **Parameters:** None
* **Return Value:** None

**8. update\_product()**

* Updates product details based on its ID.
* **Prototype:** void update\_product();
* **Parameters:** None
* **Return Value:** None

**9. search\_product()**

* Searches for a product by its ID and displays details.
* **Prototype:** void search\_product();
* **Parameters:** None
* **Return Value:** None

**10. sort\_products()**

* Sorts products based on price, quantity, or name.
* **Prototype:** void sort\_products();
* **Parameters:** None
* **Return Value:** None

**11. sold\_products()**

* Records product sales and updates stock levels.
* **Prototype:** void sold\_products();
* **Parameters:** None
* **Return Value:** None

**12. restock\_products()**

* Identifies and displays products that need restocking.
* **Prototype:** void restock\_products();
* **Parameters:** None
* **Return Value:** None

**13. main()**

* The main function that handles user interaction and drives the application flow.

**Data Structures**

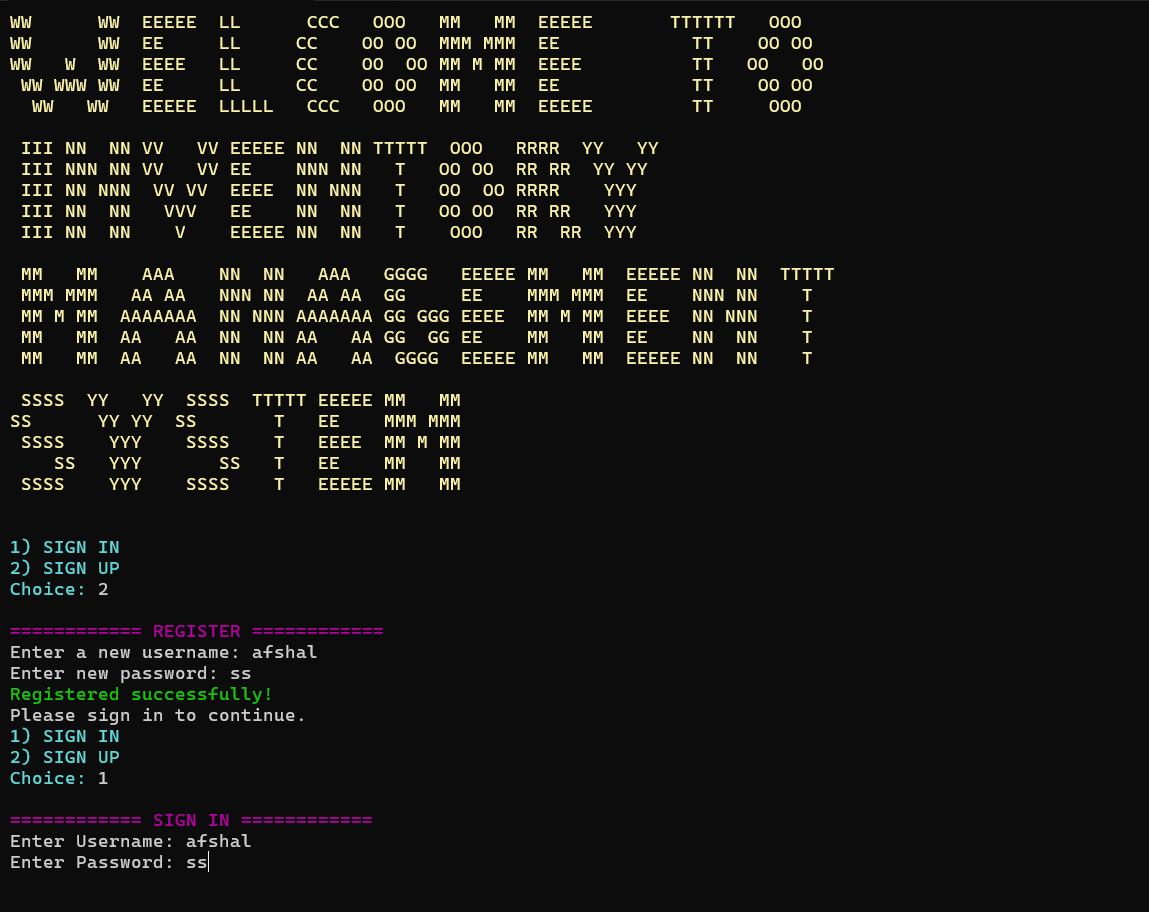
* **Product Structure:**
  + id: Unique product identifier
  + name: Product name
  + price: Product price
  + quantity: Current stock quantity
  + items\_sold: Total items sold
* **User Structure:**
  + username: User's username
  + password: User's password

**Global Variables**

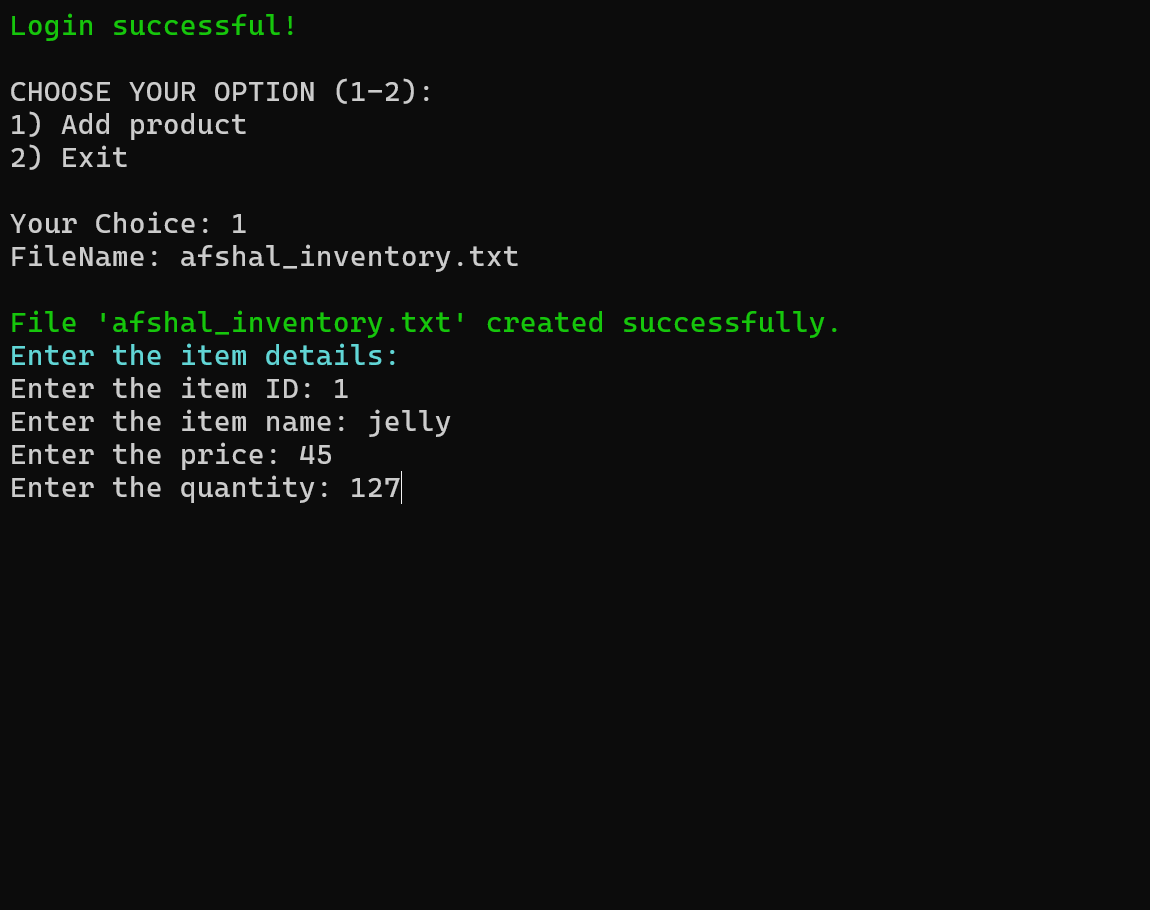
* fileName: Stores the unique inventory file name for the logged-in user
* products: Array to store product information
* product\_count: Tracks the number of products in the inventory

**Sample Test Case:-**

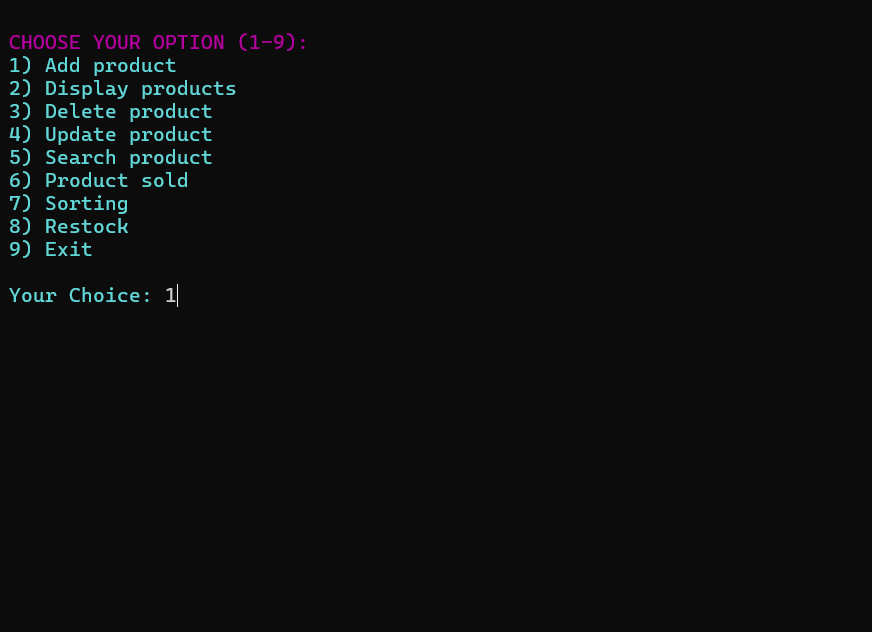
**1:-** signing up and signing in.



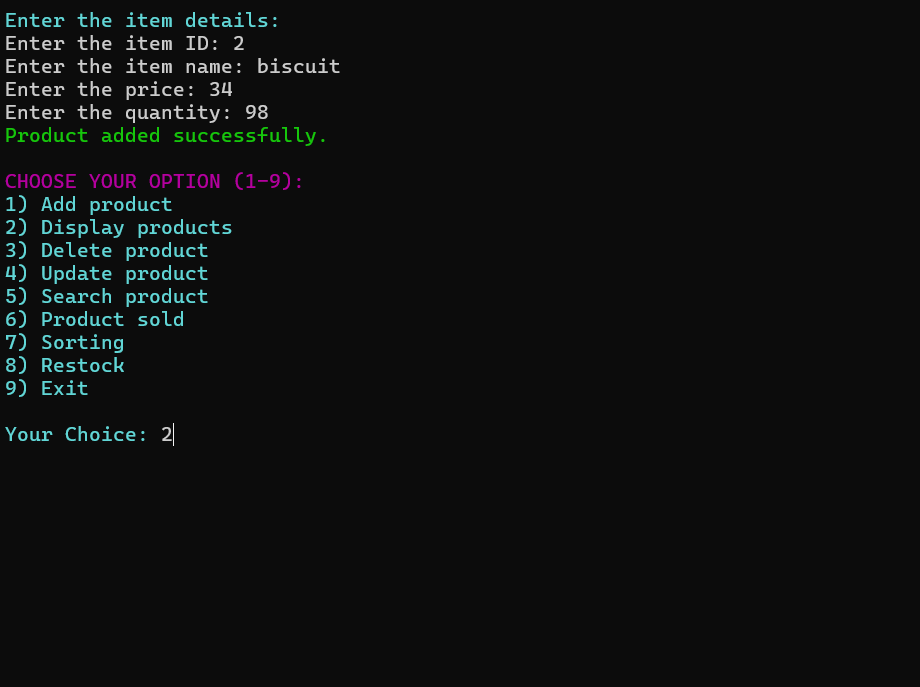
**2:-** Creating file upon adding a product:-



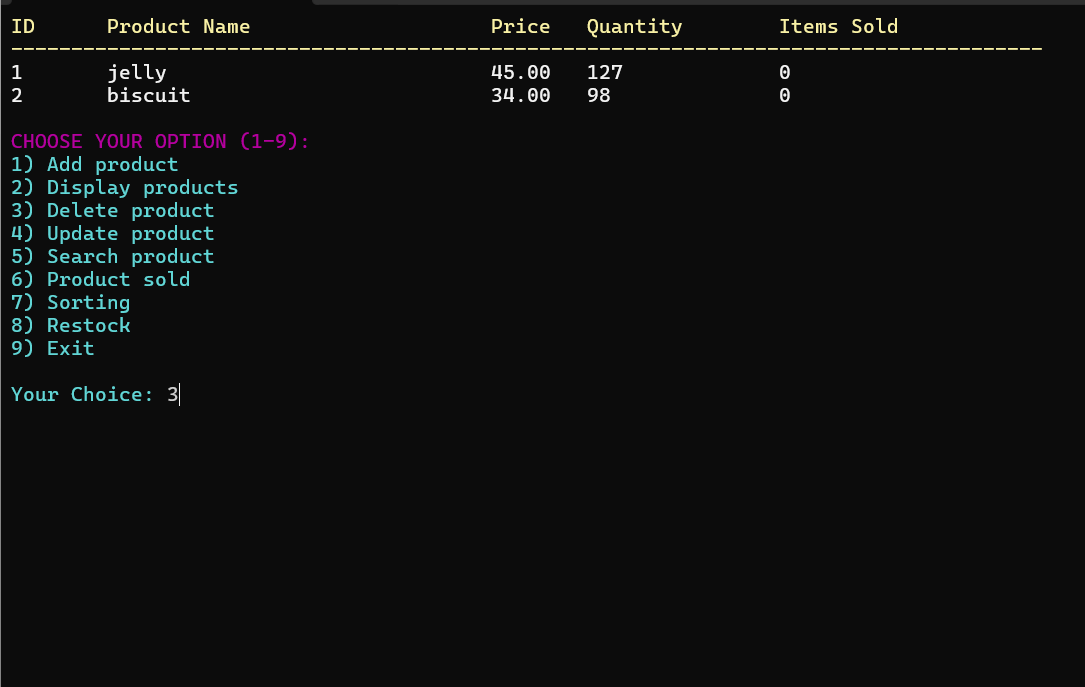
**3:-** Choosing the option of add product:-



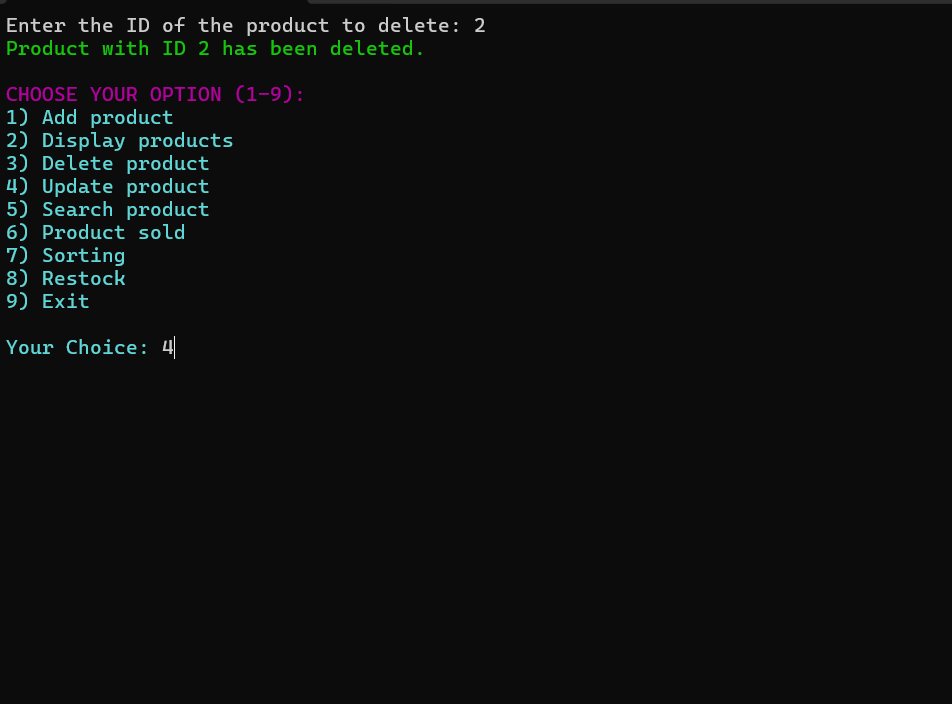
**4:-** Adding another item and choosing display option.



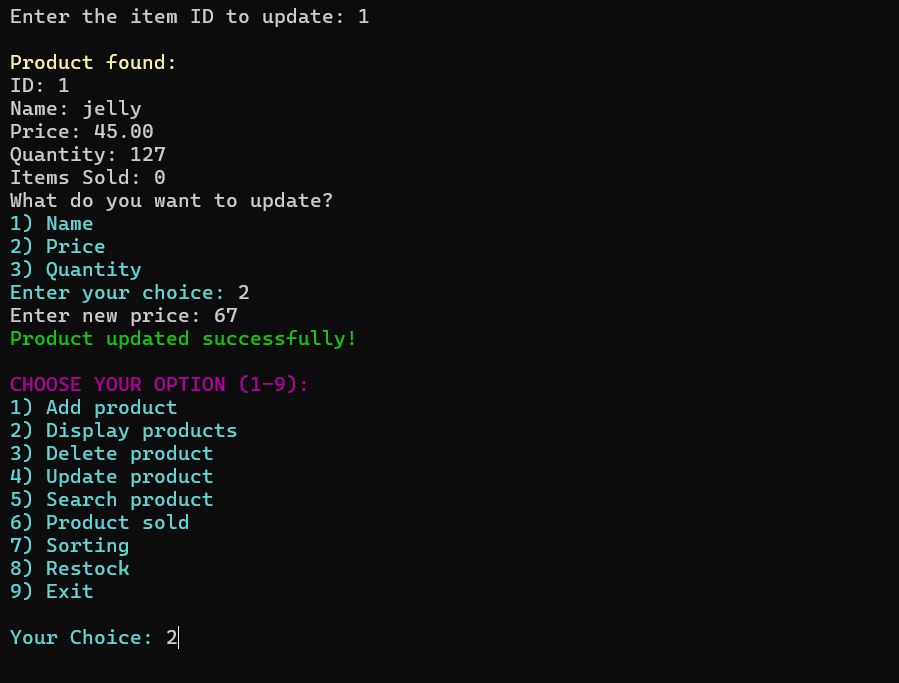
**5:-** Product is displayed and choosing delete option.



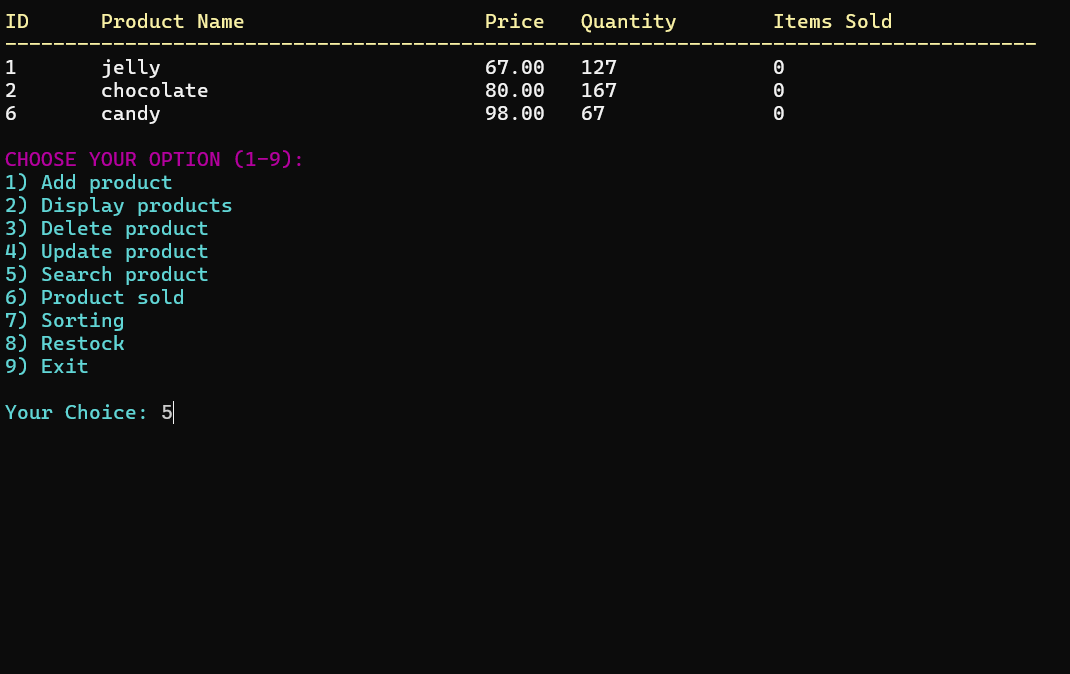
**6:-** Product has been deleted and now choosing update option.



**7:-** Updated item with respect to price and choosing display option again.



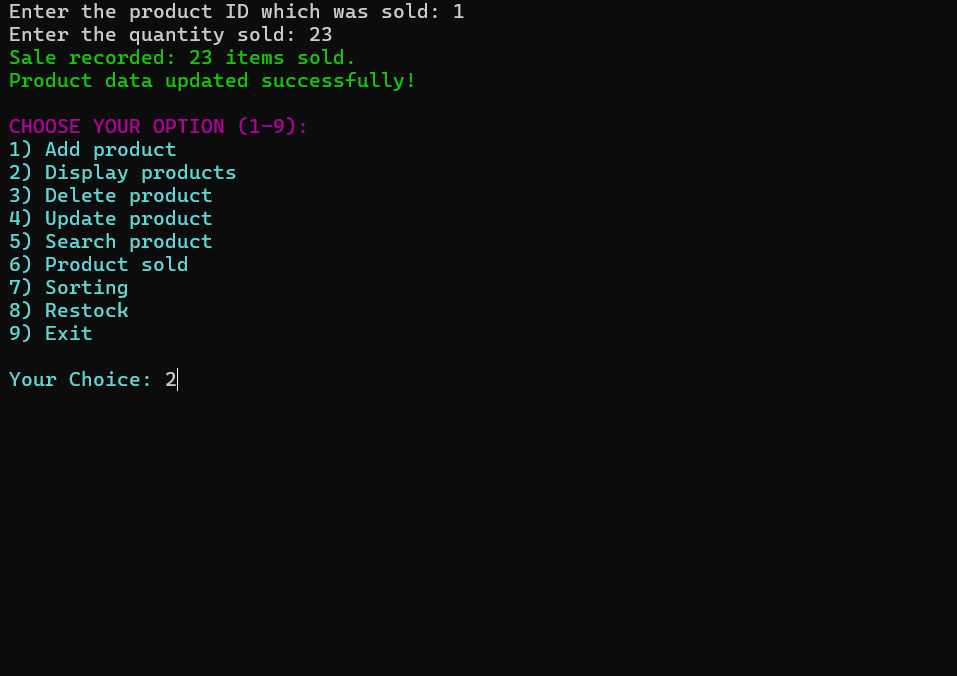
**8:-** products displayed and choosing search option.



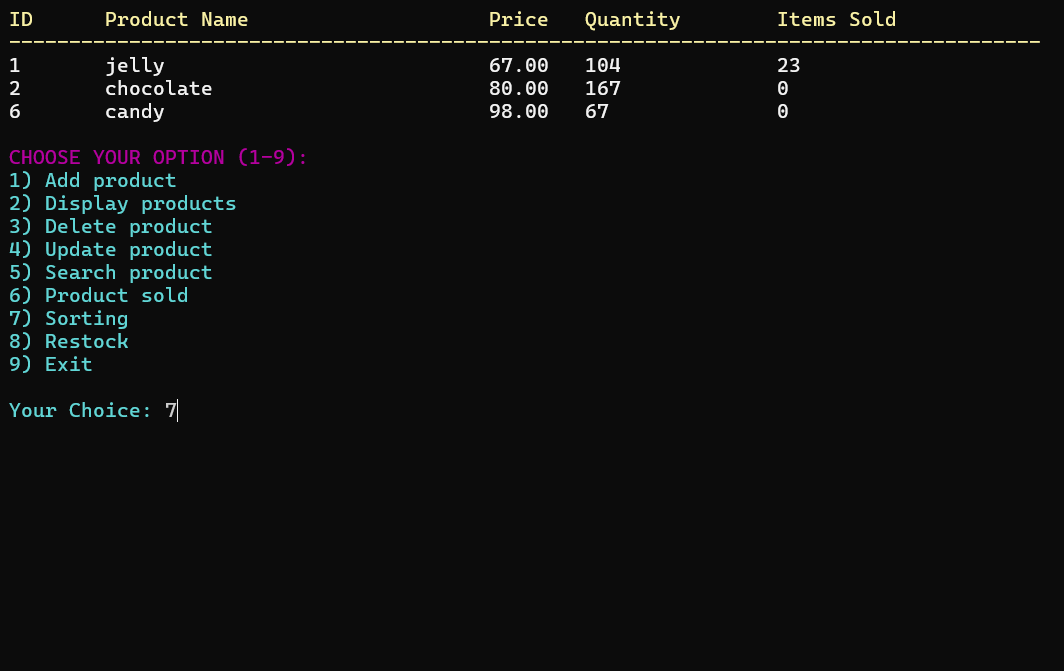
**9:-** searching ID in inventory and proceeding towards product sold function.



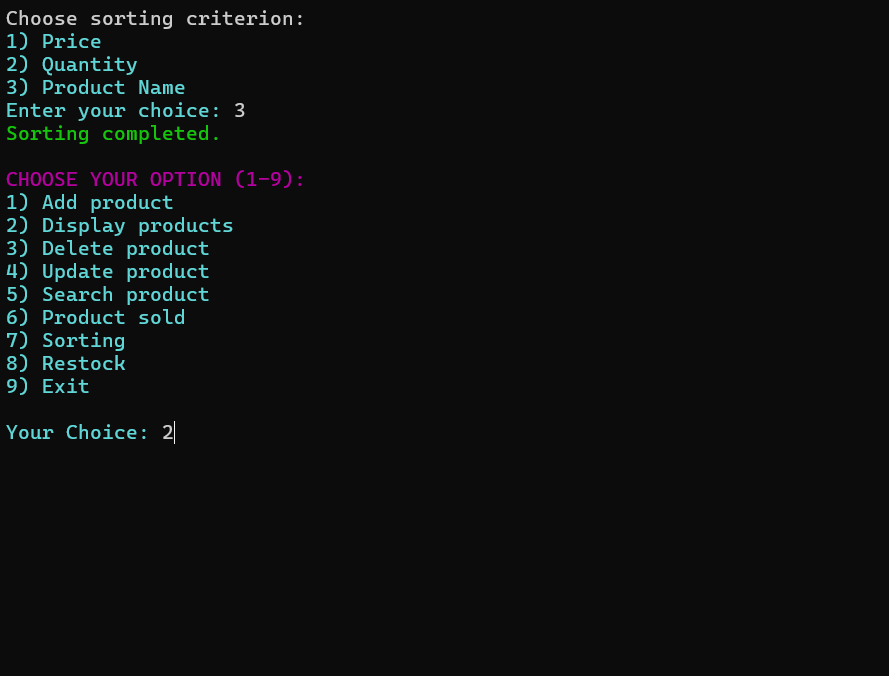
**10:-** Entering id and amount of item sold and choosing display option again.



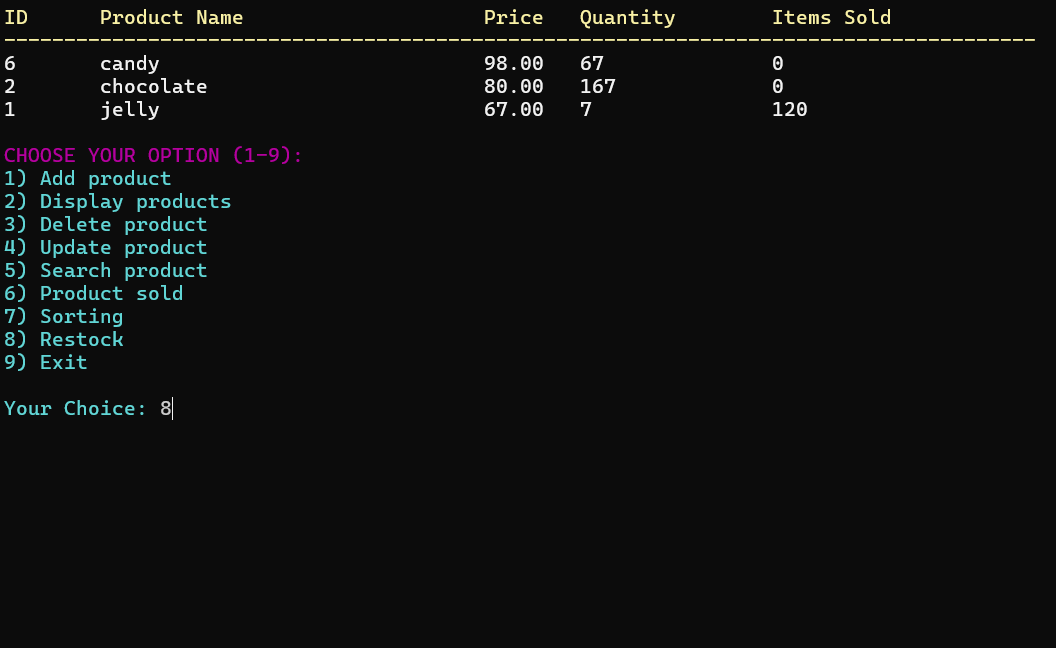
**11:-** Products are displayed and now choosing sorting option.



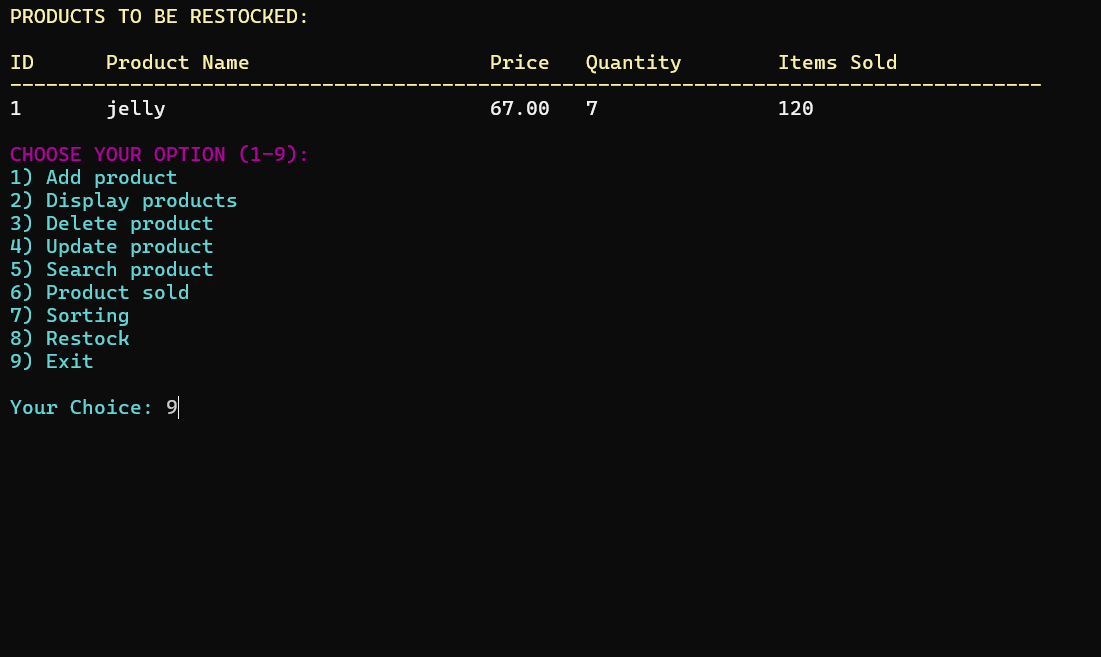
**12:-** Sorting completed and displaying products again.



**13:-** Products displayed and choosing restock option (restock set to quantity<10)



**14:-** Products needed to Restock are displayed and now choosing the exit option.



**15:-** Exiting the inventory.

