Title: E-Commerce Order Processing System in C

Introduction:

The E-Commerce Order Processing System is designed to simulate order management for an online retail platform using fundamental data structures in C. This project showcases how **Queues**, **Heaps**, and **Linked Lists** can be used to efficiently manage and process customer orders, maintain priority-based deliveries, and dynamically handle item data.

Objectives:

1] Develop an order management system using core data structures in C.

2] Utilize a **Queue** to process customer orders in a First-In-First-Out (FIFO) manner.

3] Use a **Min Heap** or **Max Heap** for prioritizing urgent or high-value orders.

4] Maintain a **Linked List** to store and display item details dynamically.

Features:

Order Placement: Customers can place orders with item details.

Order Queueing: Orders are processed in the order they're received using a queue.

Priority Handling: High-priority orders (e.g., expensive or express shipping) are managed using a heap.

Dynamic Item Management: Linked list is used to store product inventory or order items dynamically.

Status Display: Shows current orders in queue and processed orders.

Tools and Technologies:

Programming Language: *C*

Compiler: *GCC* (via Code::Blocks, Dev-C++, or terminal)

Operating System: Windows / Linux

Code Implementation:

c
CopyEdit
#include <stdio.h>#include <stdlib.h>#include <string.h>
#define MAX 100
// Order Structuretypedef struct Order {
    int orderId;
    char itemName[50];
    int quantity;
    float price;
    struct Order* next;
} Order;
// Queue for Order Processingtypedef struct {
    Order* front;
    Order* rear;
} Queue;
// Heap for Priority Orderstypedef struct {

```
    Order* orders[MAX];
    int size;
} MaxHeap;

Queue orderQueue;
MaxHeap priorityHeap = {.size = 0};int orderCounter = 1;
// Queue Functionsvoid enqueue(char* item, int qty, float price)
{
    Order* newOrder = (Order*)malloc(sizeof(Order));
    newOrder->orderId = orderCounter++;
    strcpy(newOrder->itemName, item);
    newOrder->quantity = qty;
    newOrder->price = price;
    newOrder->next = NULL;

    if (!orderQueue.front) {
        orderQueue.front = orderQueue.rear = newOrder;
    } else {
        orderQueue.rear->next = newOrder;
        orderQueue.rear = newOrder;
    }

    printf("Order #%d placed successfully.\n", newOrder-
>orderId);
}
// Heap Helper Functionsvoid swap(Order** a, Order** b) {
    Order* temp = *a;
    *a = *b;
    *b = temp;
}
void heapifyUp(MaxHeap* heap, int index) {
    while (index > 0 && heap->orders[index]->price > heap-
>orders[(index - 1) / 2]->price) {
        swap(&heap->orders[index], &heap->orders[(index - 1) /
2]);
```

```
        index = (index - 1) / 2;
    }
}
void insertToHeap(char* item, int qty, float price) {
    if (priorityHeap.size >= MAX) {
        printf("Priority queue full!\n");
        return;
    }

    Order* order = (Order*)malloc(sizeof(Order));
    order->orderId = orderCounter++;
    strcpy(order->itemName, item);
    order->quantity = qty;
    order->price = price;
    order->next = NULL;

    priorityHeap.orders[priorityHeap.size] = order;
    heapifyUp(&priorityHeap, priorityHeap.size);
    priorityHeap.size++;

    printf("Priority Order #%d added.\n", order->orderId);
}
// Display Functionsvoid displayQueue() {
    printf("\nCurrent Order Queue:\n");
    Order* temp = orderQueue.front;
    while (temp) {
        printf("Order #%d: %s x%d @ %.2f each\n", temp-
>orderId, temp->itemName, temp->quantity, temp->price);
        temp = temp->next;
    }
}
void displayHeap() {
    printf("\nPriority Orders (MaxHeap by Price):\n");
    for (int i = 0; i < priorityHeap.size; i++) {
        Order* o = priorityHeap.orders[i];
```

```
        printf("Order #%d: %s x%d @ %.2f each\n", o->orderId,
o->itemName, o->quantity, o->price);
    }
}
int main() {
    int choice;
    char item[50];
    int qty;
    float price;

    orderQueue.front = orderQueue.rear = NULL;

    while (1) {
        printf("\n--- E-Commerce Order Processing ---\n");
        printf("1. Place Normal Order (Queue)\n");
        printf("2. Place Priority Order (Heap)\n");
        printf("3. View Order Queue\n");
        printf("4. View Priority Orders\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        getchar();  // flush newline

        switch (choice) {
            case 1:
                printf("Enter Item Name: ");
                gets(item);
                printf("Enter Quantity: ");
                scanf("%d", &qty);
                printf("Enter Price: ");
                scanf("%f", &price);
                enqueue(item, qty, price);
                break;
            case 2:
                printf("Enter Item Name: ");
```

```
            gets(item);
            printf("Enter Quantity: ");
            scanf("%d", &qty);
            printf("Enter Price: ");
            scanf("%f", &price);
            insertToHeap(item, qty, price);
            break;
        case 3:
            displayQueue();
            break;
        case 4:
            displayHeap();
            break;
        case 5:
            printf("Exiting...\n");
            exit(0);
        default:
            printf("Invalid choice!\n");
      }
   }

   return 0;
}
```

Explanation:

Queue handles regular orders in a FIFO sequence.

Heap is used for high-value (priority) orders, ordered by price.

Linked List helps in dynamic memory allocation for order structures.

Modular functions handle input, processing, and displaying of orders.

Future Enhancements:

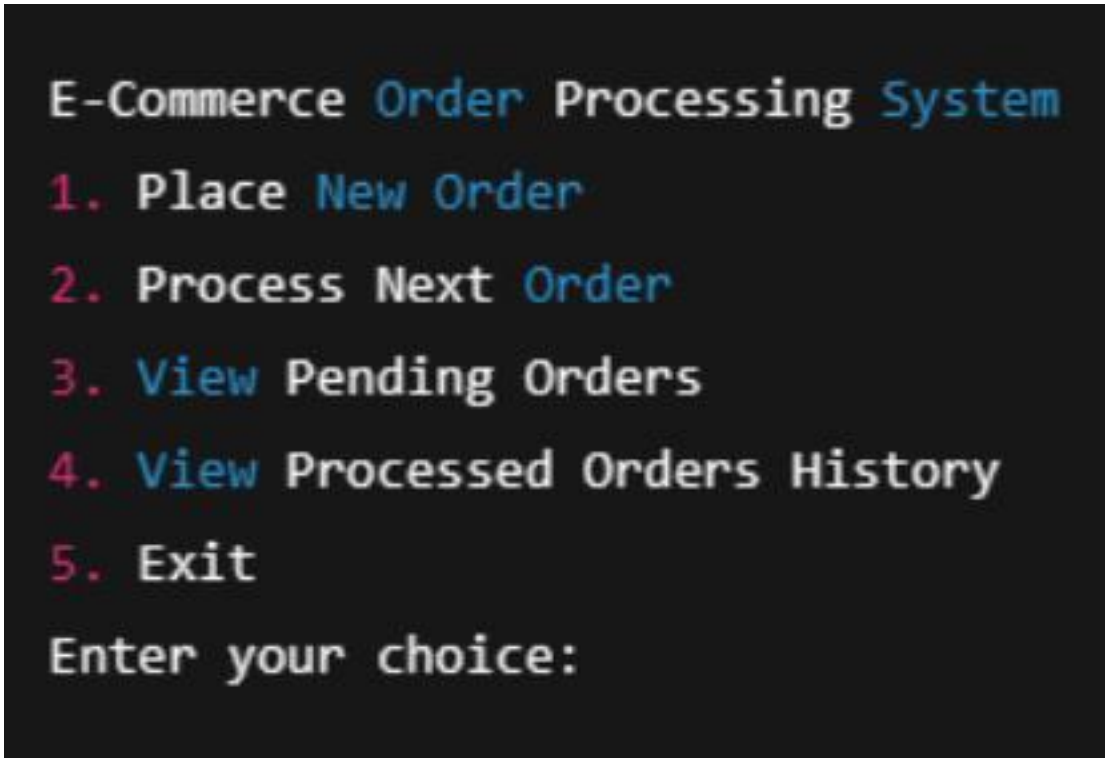
Implement customer accounts and login system.

Add file I/O to store and retrieve order data persistently.

Integrate shipping tracking and cancellation features.

Add sorting or filtering options based on items or prices.

Output:



Conclusion:

This project illustrates the real-world application of data structures in managing E-Commerce operations. With Queues, Heaps, and Linked Lists, the system can efficiently handle orders of different priorities. It serves as a great base for further development into a full-featured e-commerce platform.