In stock market, the user buys/sells stocks, and each buy/sell action is called an `order`. An order includes an user id, the stock symbol the user wants to trade, the trading amount (how many shares of stocks) with what price. For example an order:

```
"user1", "AAPL", 10, 150.0
```

denotes user1 wants to trade 10 shares of "AAPL" stock with the price 150.0.

We organize one order information as an order object below.

```
class Order {
public:
    std::string user_id;
    std::string symbol;
    int quantity;
    double price;
    Order* next;

    Order(const std::string& uid, const std::string& sym, int qty, double
pr)
        : user_id(uid), symbol(sym), quantity(qty), price(pr),
next(nullptr) {}
};
```

There are two categories of orders, the `ask` (sell) order and the `bid` (buy) order. In assignment 1, we save the orders into a linked list. In this assignment, we save the ask/bid orders into the ask/bid queue. Once a sell/buy order is created, the order is enqueued into the ask/bid queue. This assignment asks to implement a **queue** where **each element in the queue is an Order object**. The queue has the same ADT methods learned in class, as below:

```
class Queue {
private:
    Order* front;
    Order* rear;
    int counter;

public:
    Queue() : front(nullptr), rear(nullptr), counter(0) {}
    ~Queue();
    bool IsEmpty() const;
    bool Enqueue(const string& user_id, const string& symbol, int
quantity, double price);
    bool Dequeue();
    void DisplayQueue() const;
};
```

You can either use circular array or linkedlist to implement the queue. Although the above ADT uses LinkedList implementation as the example, you may change accordingly to fit your Circular Array implementation. The main function demonstrates adding orders to the queue, dequeue, displayQueue:

```
int main() {
    // Create a queue of orders
    Queue askQueue, bidQueue;

    // Enqueue the asks
    askQueue.Enqueue("user1", "AAPL", 10, 150.0);
    askQueue.Enqueue("user1", "AAPL", 5, 160.0);
    askQueue.Enqueue("user1", "GOOG", 8, 200.0);

    // Enqueue the bids
    bidQueue.Enqueue("user2", "AAPL", 15, 160.0);
    bidQueue.Enqueue("user2", "GOOG", 10, 210.0);

    // Display the ask orders
    cout << "Ask orders:" << endl;
    askQueue.DisplayQueue();

    // Display the bid orders
    cout << "Bid orders:" << endl;
    bidQueue.DisplayQueue();

    // Dequeue an order
    askQueue.Dequeue();

    // Display the updated queue
    cout << "\nUpdated orders in the queue:" << endl;
    askQueue.DisplayQueue();

    return 0;
}
```

and the desired output is:

```
Ask orders:
Order - User ID: user1, Symbol: AAPL, Quantity: 10, Price: 150
Order - User ID: user1, Symbol: AAPL, Quantity: 5, Price: 160
Order - User ID: user1, Symbol: GOOG, Quantity: 8, Price: 200
Bid orders:
Order - User ID: user2, Symbol: AAPL, Quantity: 15, Price: 160
Order - User ID: user2, Symbol: GOOG, Quantity: 10, Price: 210

Updated orders in the queue:
Order - User ID: user1, Symbol: AAPL, Quantity: 5, Price: 160
Order - User ID: user1, Symbol: GOOG, Quantity: 8, Price: 200
```