#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

// Define a struct to represent an item record

struct ItemRecord {

std::string name;

int quantity;

double price;

// Constructor for initializing an item record

ItemRecord(const std::string& n, int q, double p) : name(n), quantity(q), price(p) {}

// Overload the less than operator for sorting

bool operator<(const ItemRecord& other) const {

return name < other.name;

}

};

int main() {

// Create a vector of ItemRecord objects

std::vector<ItemRecord> itemRecords;

// Add some sample records to the vector

itemRecords.push\_back(ItemRecord("ItemA", 10, 5.99));

itemRecords.push\_back(ItemRecord("ItemC", 5, 3.49));

itemRecords.push\_back(ItemRecord("ItemB", 8, 2.99));

itemRecords.push\_back(ItemRecord("ItemD", 15, 4.99));

// Sort the item records by name

std::sort(itemRecords.begin(), itemRecords.end());

// Display the sorted item records

std::cout << "Sorted Item Records by Name:\n";

for (const ItemRecord& item : itemRecords) {

std::cout << "Name: " << item.name << ", Quantity: " << item.quantity << ", Price: $" << item.price

<< "\n";

}

// Perform a binary search for a specific item by name

std::string searchName = "ItemC";

auto it = std::lower\_bound(itemRecords.begin(), itemRecords.end(), ItemRecord(searchName, 0, 0.0));

if (it != itemRecords.end() && it->name == searchName) {

std::cout << "\nFound " << searchName << " in the item records.\n";

std::cout << "Name: " << it->name << ", Quantity: " << it->quantity << ", Price: $" << it->price <<

"\n";

} else {

std::cout << "\nItem " << searchName << " not found in the item records.\n";

}

return 0;

}

