**TASK # 02**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**class Employee {**

**private:**

**string name;**

**int employeeID;**

**float basicSalary;**

**float\* allowances;**

**public:**

**Employee(string empName, int empID, float salary, float\* quarterlyAllowances) {**

**name = empName;**

**employeeID = empID;**

**basicSalary = salary;**

**allowances = new float[3];**

**for (int i = 0; i < 3; i++) {**

**allowances[i] = quarterlyAllowances[i];**

**}**

**}**

**~Employee() {**

**delete[] allowances;**

**}**

**float calculateTotalEarnings() {**

**float totalEarnings = basicSalary;**

**for (int i = 0; i < 3; i++) {**

**totalEarnings += allowances[i];**

**}**

**return totalEarnings;**

**}**

**void displayDetails() {**

**cout << "Employee ID: " << employeeID << endl;**

**cout << "Name: " << name << endl;**

**cout << "Basic Salary: $" << basicSalary << endl;**

**cout << "Quarterly Allowances: ";**

**for (int i = 0; i < 3; i++) {**

**cout << "$" << allowances[i] << " ";**

**}**

**cout << endl;**

**cout << "Total Earnings: $" << calculateTotalEarnings() << endl;**

**cout << "-----------------------------" << endl;**

**}**

**};**

**int main() {**

**int numEmployees;**

**cout << "Enter the number of employees: ";**

**cin >> numEmployees;**

**Employee\*\* employees = new Employee\*[numEmployees];**

**for (int i = 0; i < numEmployees; i++) {**

**string name;**

**int id;**

**float salary;**

**float allowances[3];**

**cout << "Enter details for employee " << (i + 1) << ":" << endl;**

**cout << "Name: ";**

**cin.ignore();**

**getline(cin, name);**

**cout << "Employee ID: ";**

**cin >> id;**

**cout << "Basic Salary: ";**

**cin >> salary;**

**cout << "Enter Quarterly Allowances (3 values): ";**

**for (int j = 0; j < 3; j++) {**

**cin >> allowances[j];**

**}**

**employees[i] = new Employee(name, id, salary, allowances);**

**}**

**cout << "\nEmployee Details:\n";**

**for (int i = 0; i < numEmployees; i++) {**

**employees[i]->displayDetails();**

**}**

**for (int i = 0; i < numEmployees; i++) {**

**delete employees[i];**

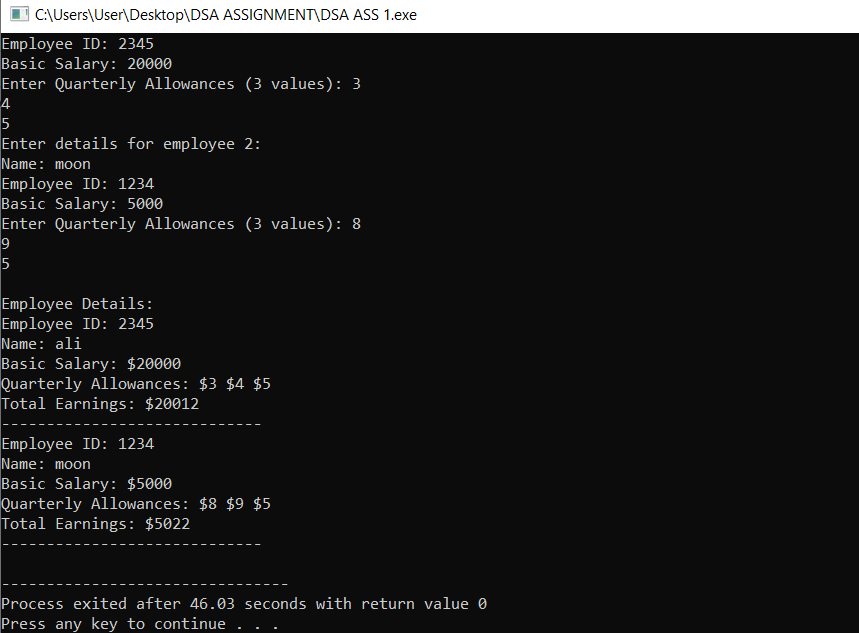
**}**

**delete[] employees;**

**return 0;**

**}**

**OUTPUT**



**TASK 02**

**#include <iostream>**

**#include <string>**

**#include <iomanip>**

**using namespace std;**

**class Student {**

**private:**

**string name;**

**int rollNumber;**

**int\* marks;**

**int numSubjects;**

**public:**

**Student(string studentName, int studentRollNumber, int\* studentMarks, int subjects) {**

**name = studentName;**

**rollNumber = studentRollNumber;**

**numSubjects = subjects;**

**marks = new int[numSubjects];**

**for (int i = 0; i < numSubjects; i++) {**

**marks[i] = studentMarks[i];**

**}**

**}**

**~Student() {**

**delete[] marks;**

**}**

**float calculateAverage() {**

**int total = 0;**

**for (int i = 0; i < numSubjects; i++) {**

**total += marks[i];**

**}**

**return static\_cast<float>(total) / numSubjects;**

**}**

**void displayDetails() {**

**cout << "Name: " << name << ", Roll Number: " << rollNumber << ", Marks: ";**

**for (int i = 0; i < numSubjects; i++) {**

**cout << marks[i] << " ";**

**}**

**cout << ", Average: " << fixed << setprecision(2) << calculateAverage() << endl;**

**}**

**float getAverage() {**

**return calculateAverage();**

**}**

**};**

**void sortStudents(Student\*\* students, int count) {**

**for (int i = 0; i < count - 1; i++) {**

**for (int j = 0; j < count - i - 1; j++) {**

**if (students[j]->getAverage() < students[j + 1]->getAverage()) {**

**swap(students[j], students[j + 1]);**

**}**

**}**

**}**

**}**

**int main() {**

**int numStudents;**

**cout << "Enter the number of students: ";**

**cin >> numStudents;**

**Student\*\* students = new Student\*[numStudents];**

**for (int i = 0; i < numStudents; i++) {**

**string name;**

**int rollNumber;**

**int marks[5];**

**cout << "Enter details for student " << (i + 1) << ":" << endl;**

**cout << "Name: ";**

**cin.ignore();**

**getline(cin, name);**

**cout << "Roll Number: ";**

**cin >> rollNumber;**

**cout << "Enter marks for 5 subjects: ";**

**for (int j = 0; j < 5; j++) {**

**cin >> marks[j];**

**}**

**students[i] = new Student(name, rollNumber, marks, 5);**

**}**

**sortStudents(students, numStudents);**

**cout << "\nStudent Details (Sorted by Average Marks):" << endl;**

**for (int i = 0; i < numStudents; i++) {**

**students[i]->displayDetails();**

**}**

**for (int i = 0; i < numStudents; i++) {**

**delete students[i];**

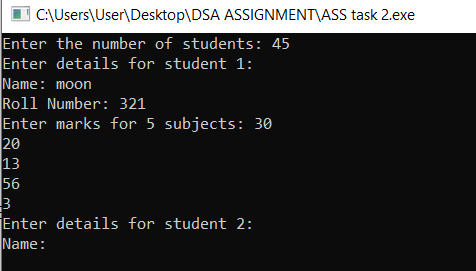
**}**

**delete[] students;**

**return 0;**

**}**

**OUTPUT**



**TASK # 03**

**#include <iostream>**

**#include <vector>**

**#include <string>**

**using namespace std;**

**class Book {**

**private:**

**string title;**

**string author;**

**string isbn;**

**int availableCopies;**

**public:**

**Book(string title, string author, string isbn, int availableCopies)**

**: title(title), author(author), isbn(isbn), availableCopies(availableCopies) {}**

**~Book() {}**

**string getTitle() { return title; }**

**string getAuthor() { return author; }**

**string getISBN() { return isbn; }**

**int getAvailableCopies() { return availableCopies; }**

**bool issueBook() {**

**if (availableCopies > 0) {**

**--availableCopies;**

**return true;**

**}**

**return false;**

**}**

**void returnBook() {**

**++availableCopies;**

**}**

**void display() {**

**cout << "Title: " << title << ", Author: " << author**

**<< ", ISBN: " << isbn << ", Available Copies: " << availableCopies << endl;**

**}**

**};**

**class Library {**

**private:**

**vector<Book\*> books;**

**public:**

**~Library() {**

**for (Book\* book : books) {**

**delete book;**

**}**

**}**

**void addBook(string title, string author, string isbn, int availableCopies) {**

**books.push\_back(new Book(title, author, isbn, availableCopies));**

**cout << "Book added: " << title << endl;**

**}**

**void issueBook(string title) {**

**for (Book\* book : books) {**

**if (book->getTitle() == title) {**

**if (book->issueBook()) {**

**cout << "Book issued: " << title << endl;**

**} else {**

**cout << "No copies available for: " << title << endl;**

**}**

**return;**

**}**

**}**

**cout << "Book not found: " << title << endl;**

**}**

**void returnBook(string title) {**

**for (Book\* book : books) {**

**if (book->getTitle() == title) {**

**book->returnBook();**

**cout << "Book returned: " << title << endl;**

**return;**

**}**

**}**

**cout << "Book not found: " << title << endl;**

**}**

**void searchBooks(string query) {**

**cout << "Searching for books matching: " << query << endl;**

**bool found = false;**

**for (Book\* book : books) {**

**if (book->getTitle().find(query) != string::npos ||**

**book->getAuthor().find(query) != string::npos) {**

**book->display();**

**found = true;**

**}**

**}**

**if (!found) {**

**cout << "No books found matching: " << query << endl;**

**}**

**}**

**void displayBooks() {**

**if (books.empty()) {**

**cout << "No books available in the library." << endl;**

**return;**

**}**

**cout << "Library Books:" << endl;**

**for (Book\* book : books) {**

**book->display();**

**}**

**}**

**};**

**int main() {**

**Library library;**

**int choice;**

**do {**

**cout << "\nLibrary Menu:\n";**

**cout << "1. Add Book\n";**

**cout << "2. Issue Book\n";**

**cout << "3. Return Book\n";**

**cout << "4. Search Books\n";**

**cout << "5. Display All Books\n";**

**cout << "6. Exit\n";**

**cout << "Enter your choice: ";**

**cin >> choice;**

**cin.ignore(); // To ignore the newline character left in the input buffer**

**if (choice == 1) {**

**string title, author, isbn;**

**int availableCopies;**

**cout << "Enter book title: ";**

**getline(cin, title);**

**cout << "Enter author name: ";**

**getline(cin, author);**

**cout << "Enter ISBN: ";**

**getline(cin, isbn);**

**cout << "Enter number of available copies: ";**

**cin >> availableCopies;**

**library.addBook(title, author, isbn, availableCopies);**

**} else if (choice == 2) {**

**string title;**

**cout << "Enter book title to issue: ";**

**getline(cin, title);**

**library.issueBook(title);**

**} else if (choice == 3) {**

**string title;**

**cout << "Enter book title to return: ";**

**getline(cin, title);**

**library.returnBook(title);**

**} else if (choice == 4) {**

**string query;**

**cout << "Enter title or author to search: ";**

**getline(cin, query);**

**library.searchBooks(query);**

**} else if (choice == 5) {**

**library.displayBooks();**

**} else if (choice == 6) {**

**cout << "Exiting the program." << endl;**

**} else {**

**cout << "Invalid choice. Please try again." << endl;**

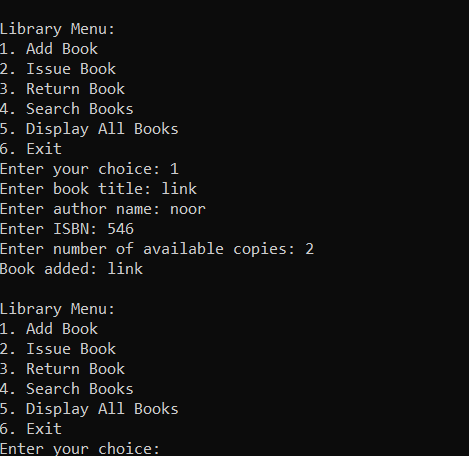
**}**

**} while (choice != 6);**

**return 0;**

**}**

**OUTPUT**



**TASK # 06**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**class Ticket {**

**private:**

**string passengerName;**

**int trainNumber;**

**int seatNumber;**

**bool isReserved;**

**public:**

**Ticket() : passengerName(""), trainNumber(0), seatNumber(0), isReserved(false) {}**

**void reserveTicket(string name, int trainNum, int seatNum) {**

**passengerName = name;**

**trainNumber = trainNum;**

**seatNumber = seatNum;**

**isReserved = true;**

**}**

**void cancelTicket() {**

**passengerName = "";**

**trainNumber = 0;**

**seatNumber = 0;**

**isReserved = false;**

**}**

**void displayTicket() {**

**if (isReserved) {**

**cout << "Passenger Name: " << passengerName << ", Train Number: " << trainNumber << ", Seat Number: " << seatNumber << endl;**

**}**

**}**

**bool isTicketReserved() {**

**return isReserved;**

**}**

**int getSeatNumber() {**

**return seatNumber;**

**}**

**};**

**class ReservationSystem {**

**private:**

**Ticket\* tickets;**

**int totalSeats;**

**public:**

**ReservationSystem(int seats) {**

**totalSeats = seats;**

**tickets = new Ticket[totalSeats];**

**}**

**~ReservationSystem() {**

**delete[] tickets;**

**}**

**void reserveTicket(string name, int trainNum, int seatNum) {**

**if (seatNum < 1 || seatNum > totalSeats) {**

**cout << "Invalid seat number." << endl;**

**return;**

**}**

**if (tickets[seatNum - 1].isTicketReserved()) {**

**cout << "Seat " << seatNum << " is already reserved." << endl;**

**} else {**

**tickets[seatNum - 1].reserveTicket(name, trainNum, seatNum);**

**cout << "Ticket reserved successfully for " << name << " on train " << trainNum << " at seat " << seatNum << "." << endl;**

**}**

**}**

**void cancelTicket(int seatNum) {**

**if (seatNum < 1 || seatNum > totalSeats) {**

**cout << "Invalid seat number." << endl;**

**return;**

**}**

**if (tickets[seatNum - 1].isTicketReserved()) {**

**tickets[seatNum - 1].cancelTicket();**

**cout << "Ticket canceled for seat " << seatNum << "." << endl;**

**} else {**

**cout << "No reservation found for seat " << seatNum << "." << endl;**

**}**

**}**

**void displayReservedTickets() {**

**cout << "Reserved Tickets:" << endl;**

**for (int i = 0; i < totalSeats; i++) {**

**tickets[i].displayTicket();**

**}**

**}**

**};**

**int main() {**

**int totalSeats;**

**cout << "Enter total number of seats: ";**

**cin >> totalSeats;**

**ReservationSystem reservationSystem(totalSeats);**

**int choice;**

**do {**

**cout << "\nTicket Reservation Menu:\n";**

**cout << "1. Reserve Ticket\n";**

**cout << "2. Cancel Ticket\n";**

**cout << "3. Display Reserved Tickets\n";**

**cout << "4. Exit\n";**

**cout << "Enter your choice: ";**

**cin >> choice;**

**if (choice == 1) {**

**string name;**

**int trainNum, seatNum;**

**cout << "Enter passenger name: ";**

**cin.ignore();**

**getline(cin, name);**

**cout << "Enter train number: ";**

**cin >> trainNum;**

**cout << "Enter seat number: ";**

**cin >> seatNum;**

**reservationSystem.reserveTicket(name, trainNum, seatNum);**

**} else if (choice == 2) {**

**int seatNum;**

**cout << "Enter seat number to cancel: ";**

**cin >> seatNum;**

**reservationSystem.cancelTicket(seatNum);**

**} else if (choice == 3) {**

**reservationSystem.displayReservedTickets();**

**} else if (choice == 4) {**

**cout << "Exiting the program." << endl;**

**} else {**

**cout << "Invalid choice. Please try again." << endl;**

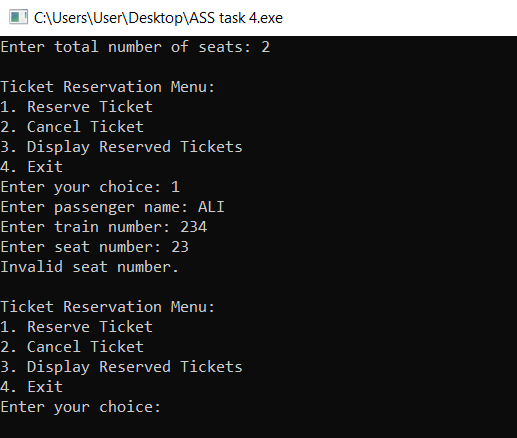
**}**

**} while (choice != 4);**

**return 0;**

**}**

**OUTPUT**



**TASK # 08**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**class Vehicle {**

**public:**

**string registrationNumber;**

**int parkingSlotNumber;**

**Vehicle() : parkingSlotNumber(-1) {}**

**};**

**class ParkingLot {**

**private:**

**Vehicle\* vehicles;**

**int totalSlots;**

**int occupiedSlots;**

**public:**

**ParkingLot(int slots) {**

**totalSlots = slots;**

**occupiedSlots = 0;**

**vehicles = new Vehicle[totalSlots];**

**}**

**~ParkingLot() {**

**delete[] vehicles;**

**}**

**void addVehicle(const string& regNumber) {**

**if (occupiedSlots < totalSlots) {**

**vehicles[occupiedSlots].registrationNumber = regNumber;**

**vehicles[occupiedSlots].parkingSlotNumber = occupiedSlots + 1;**

**cout << "Vehicle added: " << regNumber << " in slot " << vehicles[occupiedSlots].parkingSlotNumber << endl;**

**occupiedSlots++;**

**} else {**

**cout << "Parking lot is full." << endl;**

**}**

**}**

**void removeVehicle(const string& regNumber) {**

**for (int i = 0; i < occupiedSlots; i++) {**

**if (vehicles[i].registrationNumber == regNumber) {**

**cout << "Removing vehicle: " << regNumber << " from slot " << vehicles[i].parkingSlotNumber << endl;**

**vehicles[i] = vehicles[occupiedSlots - 1];**

**occupiedSlots--;**

**return;**

**}**

**}**

**cout << "Vehicle not found: " << regNumber << endl;**

**}**

**void displayAvailableSlots() {**

**cout << "Available slots: " << (totalSlots - occupiedSlots) << endl;**

**for (int i = 0; i < totalSlots; i++) {**

**bool isOccupied = false;**

**for (int j = 0; j < occupiedSlots; j++) {**

**if (vehicles[j].parkingSlotNumber == i + 1) {**

**isOccupied = true;**

**break;**

**}**

**}**

**if (!isOccupied) {**

**cout << "Slot " << (i + 1) << " is available." << endl;**

**}**

**}**

**}**

**};**

**int main() {**

**int totalSlots;**

**cout << "Enter total parking slots: ";**

**cin >> totalSlots;**

**ParkingLot parkingLot(totalSlots);**

**int choice;**

**do {**

**cout << "\nParking Lot Menu:\n";**

**cout << "1. Add Vehicle\n";**

**cout << "2. Remove Vehicle\n";**

**cout << "3. Display Available Slots\n";**

**cout << "4. Exit\n";**

**cout << "Enter your choice: ";**

**cin >> choice;**

**cin.ignore();**

**if (choice == 1) {**

**string regNumber;**

**cout << "Enter vehicle registration number: ";**

**getline(cin, regNumber);**

**parkingLot.addVehicle(regNumber);**

**} else if (choice == 2) {**

**string regNumber;**

**cout << "Enter vehicle registration number to remove: ";**

**getline(cin, regNumber);**

**parkingLot.removeVehicle(regNumber);**

**} else if (choice == 3) {**

**parkingLot.displayAvailableSlots();**

**} else if (choice == 4) {**

**cout << "Exiting the program." << endl;**

**} else {**

**cout << "Invalid choice. Please try again." << endl;**

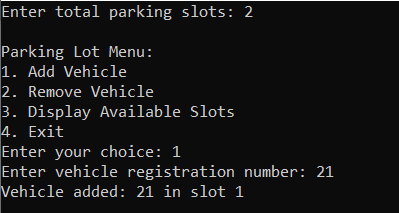
**}**

**} while (choice != 4);**

**return 0;**

**}**

**OUTPUT**



**TASK # 09**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**class Contact {**

**public:**

**string name;**

**string phoneNumber;**

**Contact\* next;**

**Contact(string name, string phoneNumber)**

**: name(name), phoneNumber(phoneNumber), next(nullptr) {}**

**};**

**class ContactBook {**

**private:**

**Contact\* head;**

**public:**

**ContactBook() : head(nullptr) {}**

**~ContactBook() {**

**while (head) {**

**Contact\* temp = head;**

**head = head->next;**

**delete temp;**

**}**

**}**

**void addContact(string name, string phoneNumber) {**

**Contact\* newContact = new Contact(name, phoneNumber);**

**newContact->next = head;**

**head = newContact;**

**cout << "Contact added: " << name << endl;**

**}**

**void deleteContact(string name) {**

**Contact\* current = head;**

**Contact\* previous = nullptr;**

**while (current != nullptr && current->name != name) {**

**previous = current;**

**current = current->next;**

**}**

**if (current == nullptr) {**

**cout << "Contact not found: " << name << endl;**

**return;**

**}**

**if (previous == nullptr) {**

**head = current->next;**

**} else {**

**previous->next = current->next;**

**}**

**delete current;**

**cout << "Contact deleted: " << name << endl;**

**}**

**void displayContacts() {**

**Contact\* current = head;**

**if (current == nullptr) {**

**cout << "No contacts available." << endl;**

**return;**

**}**

**cout << "Contact List:" << endl;**

**while (current != nullptr) {**

**cout << "Name: " << current->name << ", Phone: " << current->phoneNumber << endl;**

**current = current->next;**

**}**

**}**

**void searchContact(string name) {**

**Contact\* current = head;**

**while (current != nullptr) {**

**if (current->name == name) {**

**cout << "Contact found: " << current->name << " - " << current->phoneNumber << endl;**

**return;**

**}**

**current = current->next;**

**}**

**cout << "Contact not found: " << name << endl;**

**}**

**};**

**int main() {**

**ContactBook contactBook;**

**int choice;**

**do {**

**cout << "\nContact Book Menu:\n";**

**cout << "1. Add Contact\n";**

**cout << "2. Delete Contact\n";**

**cout << "3. Search Contact\n";**

**cout << "4. Display All Contacts\n";**

**cout << "5. Exit\n";**

**cout << "Enter your choice: ";**

**cin >> choice;**

**cin.ignore();**

**if (choice == 1) {**

**string name, phoneNumber;**

**cout << "Enter contact name: ";**

**getline(cin, name);**

**cout << "Enter phone number: ";**

**getline(cin, phoneNumber);**

**contactBook.addContact(name, phoneNumber);**

**} else if (choice == 2) {**

**string name;**

**cout << "Enter contact name to delete: ";**

**getline(cin, name);**

**contactBook.deleteContact(name);**

**} else if (choice == 3) {**

**string name;**

**cout << "Enter contact name to search: ";**

**getline(cin, name);**

**contactBook.searchContact(name);**

**} else if (choice == 4) {**

**contactBook.displayContacts();**

**} else if (choice == 5) {**

**cout << "Exiting the program." << endl;**

**} else {**

**cout << "Invalid choice. Please try again." << endl;**

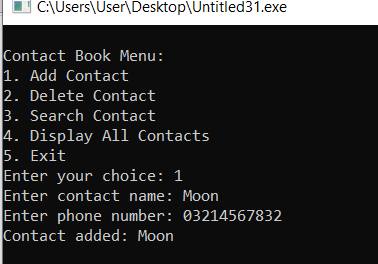
**}**

**} while (choice != 5);**

**return 0;**

**}**

**OUTPUT**



**TASK # 10**

#include <iostream>

#include <vector>

using namespace std;

class Product {

private:

int productID;

string name;

float price;

int stockQuantity;

public:

Product(int productID, string name, float price, int stockQuantity) {

this->productID = productID;

this->name = name;

this->price = price;

this->stockQuantity = stockQuantity;

}

~Product() {}

void sell(int quantity) {

if (quantity > stockQuantity) {

cout << "Not enough stock for product: " << name << endl;

} else {

stockQuantity -= quantity;

cout << "Sold " << quantity << " of " << name << endl;

}

}

void display() const {

cout << "Product ID: " << productID << ", Name: " << name

<< ", Price: $" << price << ", Stock: " << stockQuantity << endl;

if (stockQuantity < 5) { // Low stock warning threshold

cout << "Warning: Low stock for product: " << name << endl;

}

}

int getStockQuantity() const {

return stockQuantity;

}

int getProductID() const {

return productID;

}

};

class Inventory {

private:

vector<Product\*> products;

public:

~Inventory() {

for (Product\* product : products) {

delete product;

}

}

void addProduct(int id, const string& name, float price, int stock) {

Product\* newProduct = new Product(id, name, price, stock);

products.push\_back(newProduct);

cout << "Added product: " << name << endl;

}

void sellProduct(int id, int quantity) {

for (Product\* product : products) {

if (product->getProductID() == id) {

product->sell(quantity);

return;

}

}

cout << "Product ID not found." << endl;

}

void displayProducts() const {

for (const Product\* product : products) {

product->display();

}

}

};

int main() {

Inventory inventory;

inventory.addProduct(1, "Laptop", 999.99, 10);

inventory.addProduct(2, "Smartphone", 499.99, 3);

inventory.addProduct(3, "Tablet", 299.99, 5);

cout << "\nCurrent Inventory:\n";

inventory.displayProducts();

inventory.sellProduct(1, 2);

inventory.sellProduct(2, 1);

inventory.sellProduct(3, 6);

cout << "\nUpdated Inventory:\n";

inventory.displayProducts();

return 0;

}

Output:

