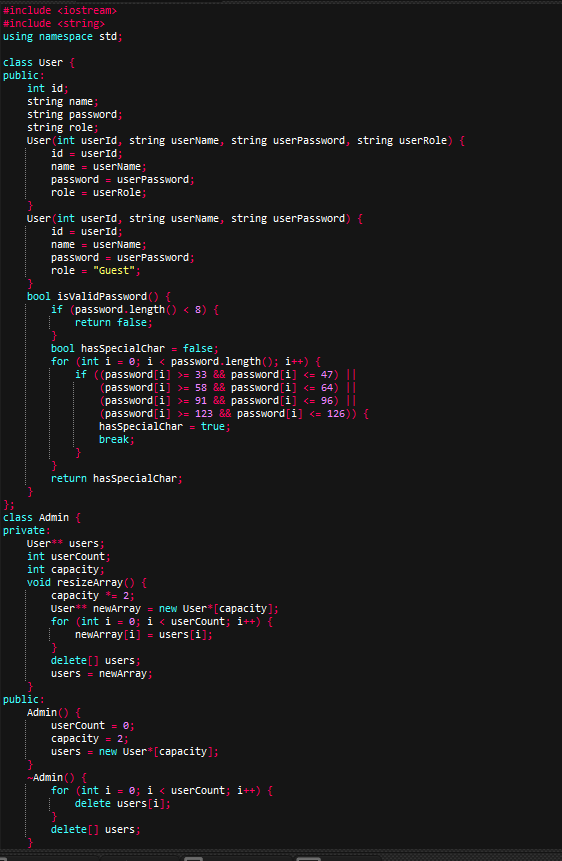
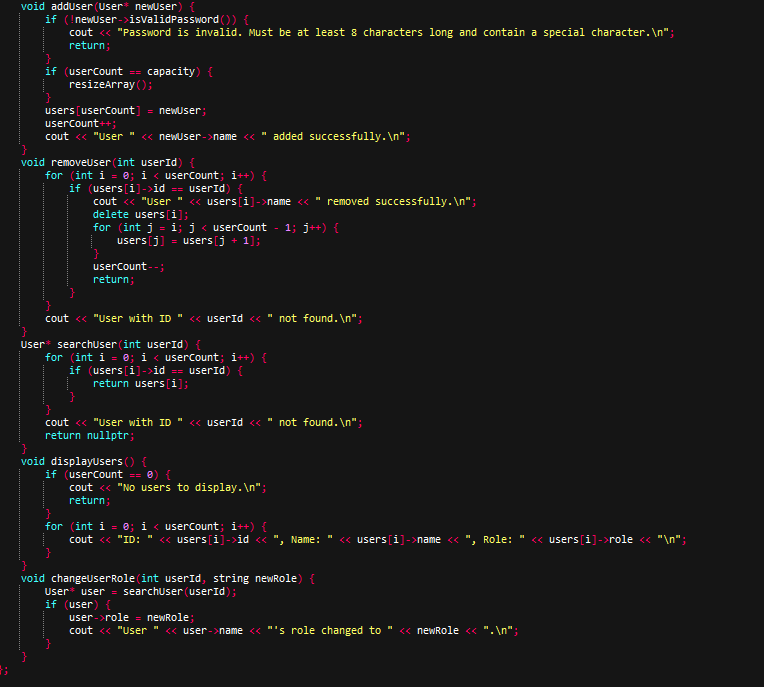
OOPs LAB 04

Name:AbdulRahimDawra RollNo:24K-0633

## TASK 01





## TASK 02

#include <iostream>

#include <cstring>

#include <iomanip>

using namespace std;

class Product {

public:

int id;

string name;

double price;

int stockQuantity;

Product() : id(0), name(""), price(0.0), stockQuantity(0) {}

Product(int \_id, const string& \_name, double \_price, int \_stock)

: id(\_id), name(\_name), price(\_price), stockQuantity(\_stock) {}

};

class CartItem {

public:

int productId;

string name;

double price;

int quantity;

CartItem() : productId(0), name(""), price(0.0), quantity(0) {}

CartItem(int pid, const string& n, double p, int q) : productId(pid), name(n), price(p),

quantity(q) {}

};

class ShoppingCart {

private:

CartItem\* items;

int count;

int capacity;

void ensureCapacity() {

if (count < capacity) return;

int newCap = (capacity == 0) ? 2 : capacity \* 2;

CartItem\* tmp = new CartItem[newCap];

for (int i = 0; i < count; ++i) tmp[i] = items[i];

delete[] items;

items = tmp;

capacity = newCap;

}

public:

ShoppingCart() : items(nullptr), count(0), capacity(0) {}

ShoppingCart(CartItem arr[], int n) : items(nullptr), count(0), capacity(0) {

for (int i = 0; i < n; ++i) addProduct(arr[i].productId, arr[i].name, arr[i].price,

arr[i].quantity);

}

~ShoppingCart() { delete[] items; }

void addProduct(int pid, const string& name, double price, int qty) {

if (qty <= 0) return;

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

if (items[i].productId == pid) {

items[i].quantity += qty;

return;

}

}

ensureCapacity();

items[count++] = CartItem(pid, name, price, qty);

}

bool removeProductById(int pid) {

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

if (items[i].productId == pid) {

for (int j = i; j + 1 < count; ++j) items[j] = items[j + 1];

--count;

return true;

}

}

return false;

}

void displayCart() const {

if (count == 0) {

cout << "Cart is empty.\n";

return;

}

cout << left << setw(6) << "ID" << setw(20) << "Name" << setw(10) << "Price" <<

setw(8) << "Qty" << setw(10) << "Subtotal" << '\n';

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

double sub = items[i].price \* items[i].quantity;

cout << setw(6) << items[i].productId << setw(20) << items[i].name << setw(10) <<

items[i].price

<< setw(8) << items[i].quantity << setw(10) << fixed << setprecision(2) << sub <<

'\n';

}

cout << "Total: " << fixed << setprecision(2) << calculateTotal() << "\n";

}

double calculateTotal() const {

double total = 0.0;

for (int i = 0; i < count; ++i) total += items[i].price \* items[i].quantity;

return total;

}

int getCount() const { return count; }

const CartItem\* getItems() const { return items; }

};

class Inventory {

private:

Product\* products;

int count;

int capacity;

void ensureCapacity() {

if (count < capacity) return;

int newCap = (capacity == 0) ? 4 : capacity \* 2;

Product\* tmp = new Product[newCap];

for (int i = 0; i < count; ++i) tmp[i] = products[i];

delete[] products;

products = tmp;

capacity = newCap;

}

public:

Inventory() : products(nullptr), count(0), capacity(0) {}

~Inventory() { delete[] products; }

void addProduct(const Product& p) {

ensureCapacity();

products[count++] = p;

}

Product\* findById(int id) {

for (int i = 0; i < count; ++i) if (products[i].id == id) return &products[i];

return nullptr;

}

void listProducts() const {

if (count == 0) { cout << "No products in inventory.\n"; return; }

cout << left << setw(6) << "ID" << setw(20) << "Name" << setw(10) << "Price" <<

setw(10) << "Stock" << '\n';

for (int i = 0; i < count; ++i)

cout << setw(6) << products[i].id << setw(20) << products[i].name << setw(10) <<

products[i].price << setw(10) << products[i].stockQuantity << '\n';

}

};

int main() {

Inventory store;

store.addProduct(Product(1, "T-shirt", 12.99, 10));

store.addProduct(Product(2, "Sneakers", 59.99, 5));

store.addProduct(Product(3, "Jeans", 34.50, 8));

store.addProduct(Product(4, "Hat", 8.75, 20));

ShoppingCart cart;

int choice;

do {

cout << "\n=== Online Store Menu ===\n";

cout << "1. List products\n2. Add product to cart\n3. Remove product from cart\n4.View cart\n5. Checkout (reduces stock)\n0. Exit\nChoice: ";

cin >> choice;

if (choice == 1) {

store.listProducts();

} else if (choice == 2) {

int pid, qty;

cout << "Enter product ID: "; cin >> pid;

Product\* p = store.findById(pid);

if (!p) { cout << "Product not found.\n"; continue; }

cout << "Enter quantity: "; cin >> qty;

if (qty <= 0) { cout << "Invalid quantity.\n"; continue; }

if (qty > p->stockQuantity) {

cout << "Not enough stock. Available: " << p->stockQuantity << '\n';

continue;

}

cart.addProduct(p->id, p->name, p->price, qty);

cout << "Added to cart.\n";

} else if (choice == 3) {

int pid; cout << "Enter product ID to remove from cart: "; cin >> pid;

if (cart.removeProductById(pid)) {

cout << "Removed.";} else cout << "Product not in cart.";

} else if (choice == 4) {

cart.displayCart();

} else if (choice == 5) {

bool ok = true;

for (int i = 0; i < cart.getCount(); ++i) {

const CartItem &ci = cart.getItems()[i];

Product\* p = store.findById(ci.productId);

if (!p || p->stockQuantity < ci.quantity) { ok = false; break; }

}

if (!ok) {

cout << "Checkout failed: one or more items insufficient stock.\n";

} else {

for (int i = 0; i < cart.getCount(); ++i) {

const CartItem &ci = cart.getItems()[i];

Product\* p = store.findById(ci.productId);

p->stockQuantity -= ci.quantity;

}

cout << "Checkout successful. Total charged: " << fixed << setprecision(2) <<

cart.calculateTotal() << "\n";

cart = ShoppingCart();

}

} else if (choice != 0) {

cout << "Invalid choice.\n";

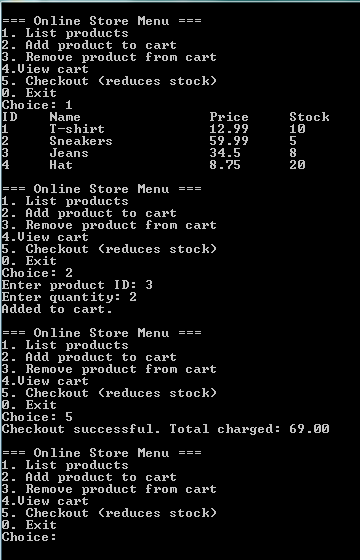
}

} while (choice != 0);

cout << "Goodbye.\n";

return 0;

}



## TASK 03

#include <iostream>

#include <string>

using namespace std;

struct Doctor {

string name;

string department;

};

class Patient {

public:

int id;

string name;

int age;

string disease;

Doctor doctorAssigned;

Patient() : id(0), name(""), age(0), disease(""), doctorAssigned({"",""}) {}

Patient(int \_id, string \_name, int \_age, string \_disease)

: id(\_id), name(\_name), age(\_age), disease(\_disease), doctorAssigned({"",""}) {}

Patient(int \_id, string \_name, int \_age, string \_disease, Doctor doc)

: id(\_id), name(\_name), age(\_age), disease(\_disease), doctorAssigned(doc) {}

};

class Hospital {

private:

Patient\* patients;

int count;

int capacity;

int nextId;

void ensureCapacity() {

if (count < capacity) return;

int newCap = (capacity == 0) ? 4 : capacity \* 2;

Patient\* tmp = new Patient[newCap];

for (int i = 0; i < count; ++i) tmp[i] = patients[i];

delete[] patients;

patients = tmp;

capacity = newCap;

}

public:

Hospital() : patients(nullptr), count(0), capacity(0), nextId(1) {}

~Hospital() { delete[] patients; }

void addPatient(string name, int age, string disease) {

ensureCapacity();

patients[count++] = Patient(nextId++, name, age, disease);

}

void addPatient(string name, int age, string disease, Doctor doc) {

ensureCapacity();

patients[count++] = Patient(nextId++, name, age, disease, doc);

}

void assignDoctor(int pid, Doctor doc) {

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

if (patients[i].id == pid) {

patients[i].doctorAssigned = doc;

return;

}

}

cout << "Patient not found.\n";

}

void removePatient(int pid) {

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

if (patients[i].id == pid) {

for (int j = i; j + 1 < count; ++j) patients[j] = patients[j + 1];

--count;

return;

}

}

cout << "Patient not found.\n";

}

void displayAll() const {

if (count == 0) { cout << "No patients in record.\n"; return; }

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

cout << "ID: " << patients[i].id

<< ", Name: " << patients[i].name

<< ", Age: " << patients[i].age

<< ", Disease: " << patients[i].disease;

if (!patients[i].doctorAssigned.name.empty())

cout << ", Doctor: " << patients[i].doctorAssigned.name << " (" <<

patients[i].doctorAssigned.department << ")";

cout << "\n";

}

}

Patient\* searchById(int pid) {

for (int i = 0; i < count; ++i)

if (patients[i].id == pid) return &patients[i];

return nullptr;

}

};

int main() {

Hospital h;

int choice;

do {

cout << "\n=== Hospital System ===\n";

cout << "1. Add patient\n2. Add patient with doctor\n3. Assign doctor\n4. Remove patient\n5. Display all\n6. Search by ID\n0. Exit\nChoice: ";

cin >> choice;

if (choice == 1) {

string n, dis; int age;

cout << "Enter name age disease: ";

cin >> n >> age >> dis;

h.addPatient(n, age, dis);

} else if (choice == 2) {

string n, dis, dn, dep; int age;

cout << "Enter name age disease doctorName department: ";

cin >> n >> age >> dis >> dn >> dep;

h.addPatient(n, age, dis, {dn, dep});

} else if (choice == 3) {

int id; string dn, dep;

cout << "Enter patient ID, doctor name, department: ";

cin >> id >> dn >> dep;

h.assignDoctor(id, {dn, dep});

} else if (choice == 4) {

int id; cout << "Enter patient ID: "; cin >> id;

h.removePatient(id);

} else if (choice == 5) {

h.displayAll();

} else if (choice == 6) {

int id; cout << "Enter patient ID: "; cin >> id;

Patient\* p = h.searchById(id);

if (p) cout << "Found: " << p->name << ", Disease: " << p->disease << "\n";

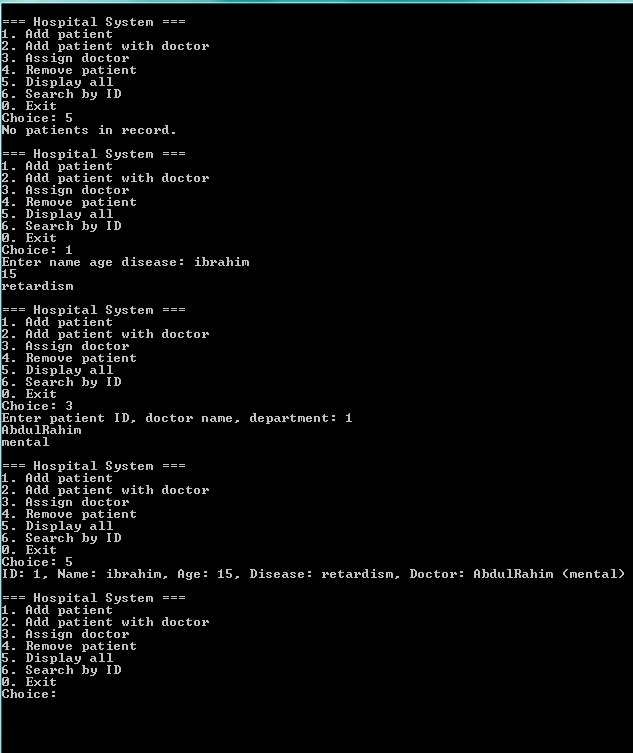
else cout << "Not found.\n";

}

} while (choice != 0);

return 0;

}



## TASK 04

#include <iostream>

#include <string>

using namespace std;

class Employee {

public:

int id;

string name;

string designation;

int hoursWorked;

Employee() : id(0), name(""), designation(""), hoursWorked(0) {}

Employee(int \_id, string \_name, string \_designation)

: id(\_id), name(\_name), designation(\_designation), hoursWorked(0) {}

Employee(int \_id, string \_name, string \_designation, int hw)

: id(\_id), name(\_name), designation(\_designation), hoursWorked(hw) {}

};

class Admin {

private:

Employee\* employees;

int count;

int capacity;

double hourlyRate;

void ensureCapacity() {

if (count < capacity) return;

int newCap = (capacity == 0) ? 4 : capacity \* 2;

Employee\* tmp = new Employee[newCap];

for (int i = 0; i < count; ++i) tmp[i] = employees[i];

delete[] employees;

employees = tmp;

capacity = newCap;

}

public:

Admin(double hr) : employees(nullptr), count(0), capacity(0), hourlyRate(hr) {}

~Admin() { delete[] employees; }

void addEmployee(Employee e) {

ensureCapacity();

employees[count++] = e;

}

void recordAttendance(int eid, int hours) {

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

if (employees[i].id == eid) {

employees[i].hoursWorked += hours;

return;

}

}

cout << "Employee not found.\n";

}

double calculateSalary(int eid) {

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

if (employees[i].id == eid) {

return employees[i].hoursWorked \* hourlyRate;

}

}

cout << "Employee not found.\n";

return 0.0;

}

void displayAll() const {

if (count == 0) { cout << "No employees.\n"; return; }

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

cout << "ID: " << employees[i].id

<< ", Name: " << employees[i].name

<< ", Designation: " << employees[i].designation

<< ", Hours: " << employees[i].hoursWorked

<< ", Salary: " << employees[i].hoursWorked \* hourlyRate

<< "\n";

}

}

};

int main() {

Admin admin(15.0);

int choice;

do {

cout << "\n=== Employee Management ===\n";

cout << "1. Add employee\n2. Record attendance\n3. Calculate salary\n4. Display all\n0. Exit\nChoice: ";

cin >> choice;

if (choice == 1) {

int id; string n, d;

cout << "Enter id, name, designation: ";

cin >> id >> n >> d;

admin.addEmployee(Employee(id, n, d));

} else if (choice == 2) {

int id, h;

cout << "Enter employee id, hours: ";

cin >> id >> h;

admin.recordAttendance(id, h);

} else if (choice == 3) {

int id; cout << "Enter employee id: "; cin >> id;

double sal = admin.calculateSalary(id);

cout << "Salary = " << sal << "\n";

} else if (choice == 4) {

admin.displayAll();

}

} while (choice != 0);

return 0;

}

