**Mohammad Taha Anwer**

**24K-3033  
Lab 5 Tasks   
  
Task 1**

#include <iostream>

using namespace std;

class Vehicle {

private:

int regID;

string modelType;

string proprietor;

public:

Vehicle(int r, string m, string o) : regID(r), modelType(m), proprietor(o) {}

Vehicle(const Vehicle& v) {

regID = v.regID;

modelType = v.modelType;

cout << "Enter New Proprietor: ";

getline(cin, proprietor);

}

void showDetails() {

cout << "Vehicle Info - Reg ID: " << regID << ", Model: " << modelType << ", Proprietor: " << proprietor << endl << endl;

}

};

int main() {

Vehicle v1(101, "BMW M5 Sedan", "Taha");

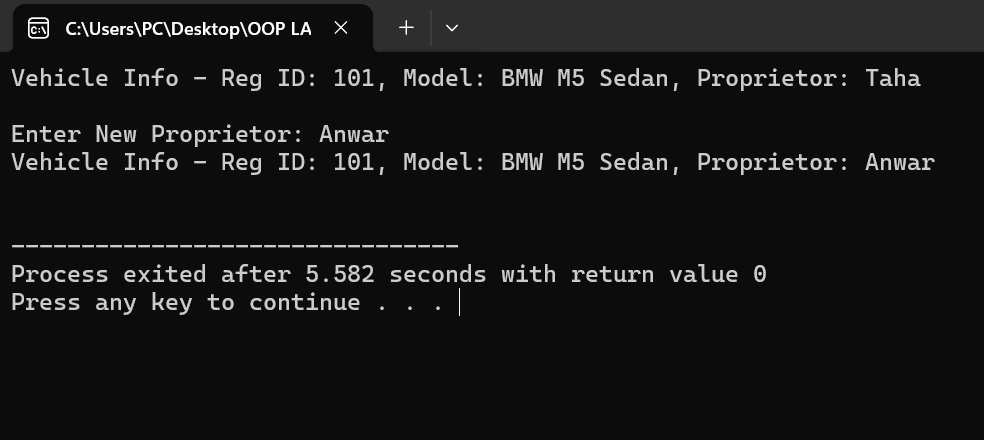
v1.showDetails();

Vehicle v2 = Vehicle(v1);

v2.showDetails();

return 0;

}



**Task 2**

#include <iostream>

using namespace std;

class MedicalRecord {

private:

int recordID;

string patientName;

float \*testScores;

int testCount;

public:

MedicalRecord(int id, string name, int num) : recordID(id), patientName(name), testCount(num) {

testScores = new float[testCount];

cout << "Enter test results:" << endl;

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

cout << "Result for Test " << i + 1 << ": ";

cin >> testScores[i];

}

}

MedicalRecord(MedicalRecord &obj) {

recordID = obj.recordID;

patientName = obj.patientName;

testCount = obj.testCount;

testScores = new float[testCount];

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

testScores[i] = obj.testScores[i];

}

}

~MedicalRecord() {

delete[] testScores;

}

void showInfo() {

cout << endl << "Record ID: " << recordID << ", Patient: " << patientName << endl << "Test Scores: ";

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

cout << testScores[i] << " ";

}

cout << endl;

}

};

int main() {

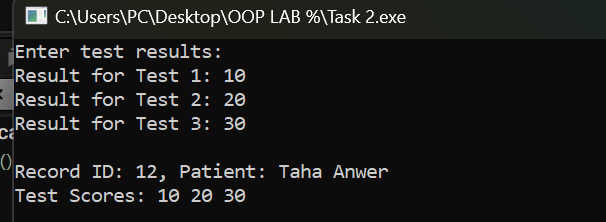
MedicalRecord taha(12, "Taha Anwer", 3);

taha.showInfo();

MedicalRecord record2 = taha;

record2.showInfo();

}



**Task 3**

#include <iostream>

using namespace std;

class Lecturer {

private:

string lecturerName;

int lecturerID;

string faculty;

public:

Lecturer(string n, int i, string f) : lecturerName(n), lecturerID(i), faculty(f) {}

string getName() {

return lecturerName;

}

string getFaculty() {

return faculty;

}

int getID() {

return lecturerID;

}

};

class Institution {

private:

string instituteName;

string location;

string\* faculties;

int facultyCount;

Lecturer \*\*lecturers;

int lecturerCount;

public:

Institution(string n, string loc) : instituteName(n), location(loc) {

lecturerCount = 0;

lecturers = nullptr;

cout << "Enter the number of faculties in " << n << ": ";

cin >> facultyCount;

cin.ignore();

faculties = new string[facultyCount];

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

cout << "Enter faculty #" << i + 1 << ": ";

getline(cin, faculties[i]);

}

}

void showLecturers() {

if (lecturerCount == 0) {

cout << "No lecturers found in " << instituteName << endl;

return;

}

cout << endl << "List of lecturers in " << instituteName << ":" << endl;

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

cout << "ID: " << lecturers[i]->getID() << endl

<< "Name: " << lecturers[i]->getName() << endl

<< "Faculty: " << lecturers[i]->getFaculty() << endl;

}

cout << endl;

}

string getInstituteName() {

return instituteName;

}

string getLocation() {

return location;

}

void enrollLecturer(Lecturer \* newLecturer) {

string faculty = newLecturer->getFaculty();

bool found = false;

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

if (faculty == faculties[i]) {

found = true;

}

}

if(!found) {

cout << "Faculty mismatch for lecturer " << newLecturer->getName()

<< " in " << instituteName << ". Enrollment failed!" << endl;

return;

}

Lecturer\*\* tempLecturers = new Lecturer\*[lecturerCount + 1];

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

tempLecturers[i] = lecturers[i];

}

delete[] lecturers;

tempLecturers[lecturerCount] = newLecturer;

lecturers = tempLecturers;

lecturerCount++;

}

void removeLecturer() {

int ID;

cout << "Enter the ID of the lecturer to be removed from " << instituteName << ": ";

cin >> ID;

cin.ignore();

int index;

bool found = false;

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

if(lecturers[i]->getID() == ID) {

index = i;

found = true;

break;

}

}

if(!found) {

cout << "Lecturer with ID " << ID << " not found. Cannot proceed with removal!" << endl;

return;

}

Lecturer\*\* tempLecturers = new Lecturer\*[lecturerCount - 1];

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

tempLecturers[i] = lecturers[i];

}

for (int i = index + 1; i < lecturerCount; i++) {

tempLecturers[i - 1] = lecturers[i];

}

delete[] lecturers;

lecturers = tempLecturers;

lecturerCount--;

}

~Institution() {

delete[] lecturers;

delete[] faculties;

}

};

int main() {

Institution fast("FAST-NUCES", "Shah Latif Town, Karachi");

Institution iba("IBA", "Inside Karachi University, Karachi");

Lecturer\* taha = new Lecturer("Taha Ahmed", 201, "Computer Science");

Lecturer\* danial = new Lecturer("Danial Khan", 202, "Computer Science");

Lecturer\* ayesha = new Lecturer("Ayesha Tariq", 203, "AI & DS");

Lecturer\* rehan = new Lecturer("Rehan Ali", 204, "Software Engineering");

Lecturer\* sara = new Lecturer("Sara Malik", 205, "Management");

Lecturer\* fahad = new Lecturer("Fahad Javed", 206, "Accounting");

Lecturer\* laiba = new Lecturer("Laiba Noor", 207, "Finance");

Lecturer\* hammad = new Lecturer("Hammad Saeed", 208, "Economics");

fast.enrollLecturer(taha);

fast.enrollLecturer(danial);

fast.enrollLecturer(ayesha);

fast.enrollLecturer(rehan);

fast.enrollLecturer(sara);

iba.enrollLecturer(fahad);

iba.enrollLecturer(laiba);

iba.enrollLecturer(hammad);

cout << "Initial list of lecturers:" << endl;

fast.showLecturers();

iba.showLecturers();

cout << "Lecturer transferring from FAST-NUCES to IBA..." << endl;

fast.removeLecturer();

iba.enrollLecturer(taha);

cout << "Updated lecturer lists after transfer:" << endl;

fast.showLecturers();

iba.showLecturers();

delete taha;

delete danial;

delete ayesha;

delete rehan;

delete sara;

delete fahad;

delete laiba;

delete hammad;

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.

**Task 4**

#include <iostream>

using namespace std;

class PowerUnit {

private:

int energy;

float level;

public:

PowerUnit() : energy(1000), level(100.0) { }

PowerUnit(int e) : energy(e), level(100.0) { }

int getEnergy() {

return energy;

}

float getLevel() {

return level;

}

void setLevel(float l) {

level = l;

}

void refill() {

if (level == 100) {

cout << "Power already at max. No need to refill!" << endl;

return;

}

level = 100.0;

cout << "Power fully restored!" << endl;

}

void deplete() {

if (level <= 0) {

cout << "Device has shut down... Recharge needed!" << endl;

return;

}

if (level <= 20) {

cout << "Alert: Low power! Current level: " << level << "%." << endl;

}

level -= 5;

if (level < 0) level = 0;

cout << "Current power level: " << level << "%." << endl;

}

};

class Gadget {

private:

PowerUnit battery;

string brand;

string version;

public:

Gadget(string b, string v, int e) : brand(b), version(v), battery(e) { }

void showSpecs() {

cout << "\nDevice Information:" << endl;

cout << "Brand: " << brand << endl;

cout << "Version: " << version << endl;

cout << "Energy Storage: " << battery.getEnergy() << " mAh" << endl;

cout << "Charge Level: " << battery.getLevel() << "%" << endl;

}

void recharge() {

cout << "\nRecharging " << brand << "..." << endl;

battery.refill();

}

void operate() {

cout << "\nOperating " << brand << "..." << endl;

battery.deplete();

}

};

int main() {

Gadget device1("Apple", "iPhone 15", 4500);

device1.showSpecs();

device1.operate();

device1.recharge();

device1.showSpecs();

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.

**Task 5**

#include <iostream>

using namespace std;

class Guide {

private:

string guideName;

int experienceYears;

string expertise;

public:

Guide(string g, int e, string ex) : guideName(g), experienceYears(e), expertise(ex) {}

string getGuideName() { return guideName; }

int getExperienceYears() { return experienceYears; }

string getExpertise() { return expertise; }

};

class Agency {

private:

int guideCount;

Guide \*\* guides;

public:

Agency() {

guideCount = 0;

guides = nullptr;

}

void addGuide(Guide\* newGuide) {

cout << "\nRegistering new guide..." << endl;

Guide \*\* temp = new Guide\* [guideCount + 1];

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

temp[i] = guides[i];

}

temp[guideCount] = newGuide;

delete[] guides;

guides = temp;

guideCount++;

cout << "Guide " << newGuide->getGuideName() << " successfully registered!" << endl << endl;

}

void removeGuide() {

cout << "\nDeleting a guide..." << endl;

string gName;

cout << "Enter guide's name to remove: ";

getline(cin, gName);

bool found = false;

int idx;

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

if(gName == guides[i]->getGuideName()) {

found = true;

idx = i;

}

}

if(!found) {

cout << "No guide named " << gName << " found! Deletion failed." << endl;

return;

}

if(guideCount == 1) {

delete[] guides;

guides = nullptr;

} else {

Guide\*\* temp = new Guide\*[guideCount - 1];

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

temp[i] = guides[i];

}

for(int i = idx; i < guideCount; i++) {

temp[i - 1] = guides[i];

}

delete[] guides;

guides = temp;

}

guideCount--;

cout << "Guide " << gName << " has been removed!" << endl << endl;

}

void listGuides() {

cout << "\nGuide Directory:" << endl;

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

cout << i + 1 << ". Name: " << guides[i]->getGuideName() << ", Experience: " << guides[i]->getExperienceYears() << " years, Expertise: " << guides[i]->getExpertise() << endl;

}

cout << endl;

}

~Agency() {

delete[] guides;

}

};

int main() {

Agency DreamTours;

Guide\* hassan = new Guide("Hassan", 5, "Cultural Tours");

Guide\* ahmed = new Guide("Ahmed", 3, "Mountain Trekking");

Guide\* zain = new Guide("Zain", 7, "Desert Safaris");

DreamTours.addGuide(hassan);

DreamTours.addGuide(ahmed);

DreamTours.addGuide(zain);

DreamTours.listGuides();

DreamTours.removeGuide();

DreamTours.listGuides();

delete hassan, ahmed, zain;

}

A screenshot of a computer program

AI-generated content may be incorrect.

**Task 6**

#include <iostream>

using namespace std;

class Film {

private:

string filmTitle, filmDirector;

float filmDuration;

public:

Film(string fT, string fD, float fDu) : filmTitle(fT), filmDirector(fD), filmDuration(fDu) {}

string getFilmTitle() { return filmTitle; }

string getFilmDirector() { return filmDirector; }

float getFilmDuration() { return filmDuration; }

};

class Theatre {

private:

int totalFilms;

Film\*\* filmList;

bool\* currentlyPlaying;

public:

Theatre() {

cout << "How many films are there? ";

cin >> totalFilms;

cin.ignore();

filmList = new Film\*[totalFilms];

currentlyPlaying = new bool[totalFilms];

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

string fT, fD;

float fDu;

cout << "Film Title: ";

getline(cin, fT);

cout << "Directed by: ";

getline(cin, fD);

cout << "Duration (hours): ";

cin >> fDu;

cin.ignore();

filmList[i] = new Film(fT, fD, fDu);

string playing;

cout << "Currently Showing? (yes/no): ";

getline(cin, playing);

currentlyPlaying[i] = ("yes" == playing) ? true : false;

}

}

~Theatre() {

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

delete filmList[i];

}

delete[] filmList;

delete[] currentlyPlaying;

}

void showFilms() {

cout << "\nNow Showing:" << endl;

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

cout << i + 1 << ". Name: " << filmList[i]->getFilmTitle() << ", Director: " << filmList[i]->getFilmDirector() << ", Duration: " << filmList[i]->getFilmDuration() << " hours, Status: " << ((currentlyPlaying[i]) ? "Playing" : "Not Playing") << endl;

}

cout << endl;

}

};

int main() {

Theatre venue;

venue.showFilms();

}

A screenshot of a computer

AI-generated content may be incorrect.

**Task 7**

#include <iostream>

using namespace std;

class Product {

private:

static int IDCounter;

int productID;

string name;

float price;

public:

Product(string n, float p) :name(n), price(p), productID(IDCounter++) {}

int getID() { return productID; }

string getName() { return name; }

float getPrice() { return price; }

};

int Product::IDCounter = 0;

class GroceryStore {

private:

int numProducts;

Product\*\* products;

public:

GroceryStore() {

numProducts = 0;

products = nullptr;

}

void addProduct(Product\* product) {

cout << endl << "Adding product..." << endl;

Product \*\* tempProducts = new Product\* [numProducts + 1];

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

tempProducts[i] = products[i];

}

tempProducts[numProducts] = product;

delete[] products;

products = tempProducts;

numProducts++;

cout << "Product " << product->getName() << " added!" << endl;

}

void displayProducts() {

if (numProducts == 0) {

cout << "No products in the store." << endl;

return;

}

cout << "\nDisplaying Products:\n";

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

cout << i + 1 << ". ID: " << products[i]->getID() << ", Name: " << products[i]->getName() << ", Price: $" << products[i]->getPrice() << endl;

}

cout << endl;

}

void removeProduct() {

int ID;

cout << "Enter ID of product to be removed: ";

cin >> ID;

cin.ignore();

int index = -1;

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

if (products[i]->getID() == ID) {

index = i;

break;

}

}

if (index == -1) {

cout << "Product " << ID << " not found! Cannot remove." << endl;

return;

}

cout << "Removing product " << ID << ": " << products[index]->getName() << "..." << endl;

Product\*\* tempProducts = new Product\* [numProducts - 1];

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

tempProducts[i] = products[i];

}

for (int i = index; i < numProducts - 1; i++) {

tempProducts[i] = products[i + 1];

}

delete[] products;

products = tempProducts;

numProducts--;

cout << "Product " << ID << " removed!" << endl << endl;

}

void searchProduct() {

cout << endl << "Searching for product..." << endl;

string name;

cout << "Enter name of product: ";

getline(cin, name);

bool found = false;

int index;

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

if(name == products[i]->getName()) {

found = true;

index = i;

}

}

if(!found) {

cout << "Product " << name << " not found!" << endl;

return;

}

cout << "Product Found!" << endl;

cout << "ID: " << products[index]->getID() << ", Name: " << products[index]->getName() << ", Price: $" << products[index]->getPrice() << endl;

}

void sortProducts() {

cout << "Sorting products as per price..." << endl;

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

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

if(products[j]->getPrice() > products[j+1]->getPrice()) {

Product \* temp = products[j];

products[j] = products[j+1];

products[j+1] = temp;

}

}

}

cout << "Products sorted!" << endl;

cout << "Cheapest Product; " << "ID: " << products[0]->getID() << ", Name: " << products[0]->getName() << ", Price: $" << products[0]->getPrice() << endl;

}

~GroceryStore() {

delete[] products;

}

};

int main() {

GroceryStore store;

Product\* AC = new Product("AC", 500.0);

Product\* sugar = new Product("Sugar", 2.0);

Product\* rice = new Product("Rice", 3.5);

Product\* milk = new Product("Milk", 1.8);

Product\* drinks = new Product("Drinks", 2.5);

Product\* chips = new Product("Chips", 1.2);

store.addProduct(AC);

store.addProduct(sugar);

store.addProduct(rice);

store.addProduct(milk);

store.addProduct(drinks);

store.addProduct(chips);

store.displayProducts();

store.removeProduct();

store.searchProduct();

store.sortProducts();

store.displayProducts();

delete AC;

delete sugar;

delete rice;

delete milk;

delete drinks;

delete chips;

return 0;

}

A screenshot of a computer program

AI-generated content may be incorrect.