TASK NO 1:

#include<iostream>

using namespace std;

class Appartment {

int ID;

string address;

string\* name;

public:

Appartment() : name(nullptr) {}

Appartment(int i, string ad, string n) {

ID = i;

address = ad;

name = new string(n);

}

Appartment(const Appartment& obj) {

if (obj.name != nullptr) {

name = new string(\*obj.name);

} else {

name = nullptr;

}

ID = obj.ID;

address = obj.address;

}

~Appartment() {

delete name;

cout << "APPARTMENT DESTROYED" << endl;

}

void display() {

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

cout << "ID : " << ID << endl;

cout << "ADDRESS: " << address << endl;

if (name) {

cout << "OWNER'S NAME: " << \*name << endl;

} else {

cout << "OWNER'S NAME: (None)" << endl;

}

}

};

class Agent {

string Agname;

Appartment\* appartments[10];

int count = 0;

public:

Agent(string n) {

Agname = n;

}

void addApart(Appartment\* apt) {

if (count < 10) {

appartments[count++] = apt;

} else {

cout << "THE LIMIT IS FULL" << endl;

}

}

void displayAgent() {

cout << "AGENT NAME: " << Agname << endl;

cout << "APARTMENTS:" << endl;

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

appartments[i]->display();

}

}

};

int main() {

Appartment\* a1 = new Appartment(101, "Street 1, City A", "John");

Appartment\* a2 = new Appartment(102, "Street 2, City B", "Alice");

Agent agent1("Michael");

agent1.addApart(a1);

agent1.addApart(a2);

agent1.displayAgent();

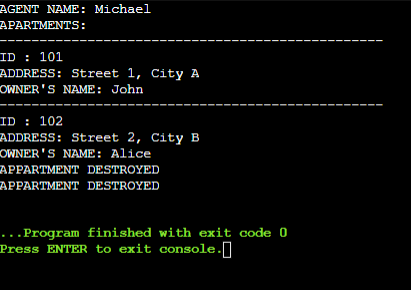
delete a1;

delete a2;

return 0;

}

OUTPUT:



TASK NO 2:

#include <iostream>

using namespace std;

class Marks {

float scores[3];

public:

Marks(float s[]) {

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

scores[i] = s[i];

}

}

Marks(const Marks &m) {

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

scores[i] = m.scores[i];

}

}

void showMarks() const {

cout << "---- Marks Details ----" << endl;

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

cout << "Test " << i + 1 << ": " << scores[i] << endl;

}

}

};

class Pupil {

string regNo;

string fullName;

Marks marks;

public:

Pupil(string r, string n, float s[]) : regNo(r), fullName(n), marks(s) {}

Pupil(const Pupil &p) : regNo(p.regNo), fullName(p.fullName), marks(p.marks) {}

void displayInfo() const {

cout << "============================" << endl;

cout << "Registration No: " << regNo << endl;

cout << "Name: " << fullName << endl;

marks.showMarks();

}

};

int main() {

float scoresArray[] = {88.5, 79.3, 92.0};

Pupil student1("S2023101", "ALISHBA AMIR", scoresArray);

student1.displayInfo();

Pupil student2(student1);

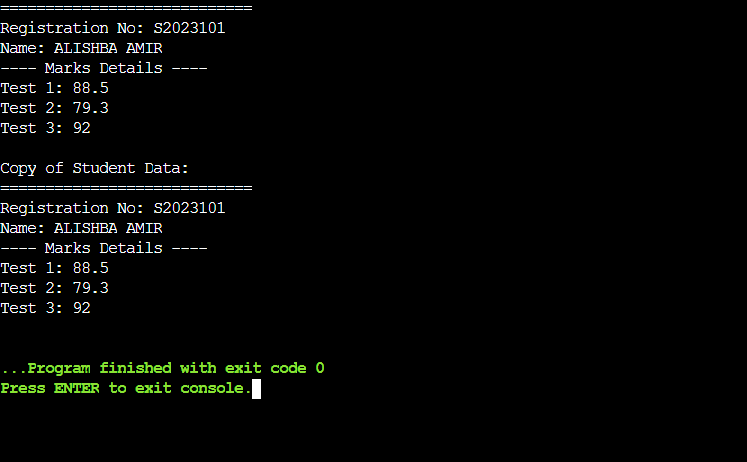
cout << endl << "Copy of Student Data:" << endl;

student2.displayInfo();

return 0;

}

OUTPUT:



TASK NO 03:

#include <iostream>

#include <string>

using namespace std;

class Staff {

string empName, jobTitle;

public:

Staff(string n, string j) : empName(n), jobTitle(j) {}

void showDetails() const {

cout << "Employee Name: " << empName << endl;

cout << "Position: " << jobTitle << endl;

}

};

class Assignment {

string projectName, dueDate;

Staff\* team[10];

int teamSize = 0;

public:

Assignment(string p, string d) : projectName(p), dueDate(d) {}

void assignStaff(Staff\* s) {

if (teamSize < 10) {

team[teamSize++] = s;

} else {

cout << "Maximum team limit reached!" << endl;

}

}

void showProjectInfo() const {

cout << "======= Project Information =======" << endl;

cout << "Project: " << projectName << endl;

cout << "Completion Date: " << dueDate << endl;

cout << "Assigned Staff for " << projectName << ":" << endl;

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

team[i]->showDetails();

}

}

};

int main() {

Staff s1("Alex Carter", "Marketing Specialist");

s1.showDetails();

Assignment proj1("Social Media Strategy", "30-June-2025");

proj1.assignStaff(&s1);

cout << endl;

proj1.showProjectInfo();

Assignment proj2("Website Redesign", "15-Aug-2025");

proj2.assignStaff(&s1);

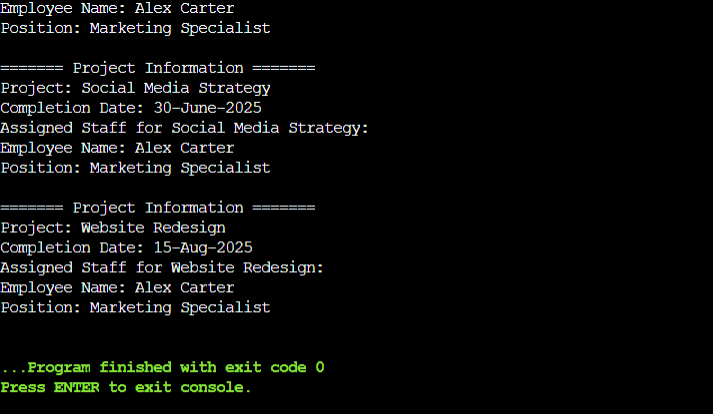
cout << endl;

proj2.showProjectInfo();

return 0;

}

OUTPUT:



TASK NO 04:

#include <iostream>

#include <string>

using namespace std;

class AlarmSystem {

string securityLevel;

public:

AlarmSystem(string level) : securityLevel(level) {}

void showDetails() const {

cout << "Security Level: " << securityLevel << endl;

}

};

class SmartHome {

string homeID;

AlarmSystem alarm;

public:

SmartHome(string id, string security) : homeID(id), alarm(security) {}

void display() const {

cout << "Smart Home ID: " << homeID << endl;

alarm.showDetails();

}

};

int main() {

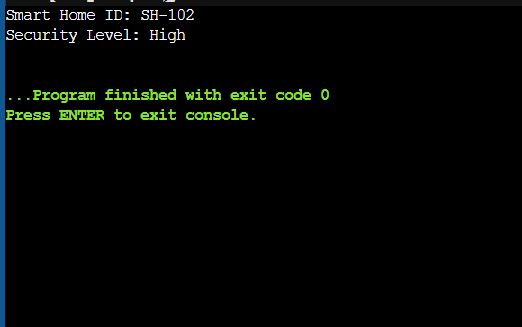
SmartHome home("SH-102", "High");

home.display();

return 0;

}

OUTPUT:



TASK NO 05:

#include <iostream>

#include <string>

using namespace std;

class Doctor {

string name, specialization;

int experience;

public:

Doctor(string n, string s, int e) : name(n), specialization(s), experience(e) {}

void showDetails() const {

cout << "Doctor: " << name << ", Specialization: " << specialization

<< ", Experience: " << experience << " years" << endl;

}

};

class Hospital {

string hospitalName;

Doctor\* doctors[10];

int count;

public:

Hospital(string name) : hospitalName(name), count(0) {}

void addDoctor(Doctor\* doc) {

if (count < 10) {

doctors[count++] = doc;

}

}

void showHospitalDetails() const {

cout << "Hospital: " << hospitalName << endl;

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

doctors[i]->showDetails();

}

}

};

int main() {

Doctor d1("Alice", "Cardiology", 10);

Doctor d2("Bob", "Neurology", 8);

Hospital h("City Hospital");

h.addDoctor(&d1);

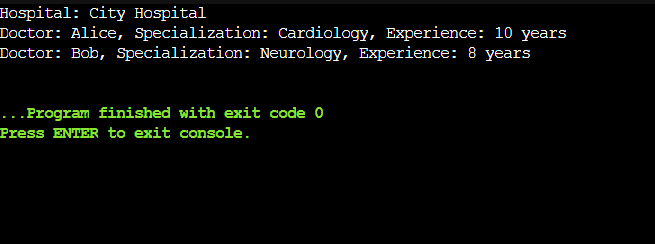
h.addDoctor(&d2);

h.showHospitalDetails();

return 0;

}

OUTPUT:



TASK NO 06:

#include <iostream>

#include <string>

using namespace std;

class Level {

string levelName;

int difficulty;

public:

Level(string name, int diff) : levelName(name), difficulty(diff) {}

void showDetails() const {

cout << "Level: " << levelName << ", Difficulty: " << difficulty << endl;

}

};

class VideoGame {

string title, genre;

Level\* levels[5];

int levelCount;

public:

VideoGame(string t, string g) : title(t), genre(g), levelCount(0) {}

void addLevel(Level\* lvl) {

if (levelCount < 5) {

levels[levelCount++] = lvl;

}

}

void showGameDetails() const {

cout << "Game: " << title << ", Genre: " << genre << endl;

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

levels[i]->showDetails();

}

}

};

int main() {

Level l1("Forest", 3);

Level l2("Desert", 5);

VideoGame game("Quest World", "Adventure");

game.addLevel(&l1);

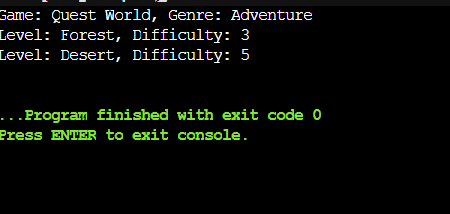
game.addLevel(&l2);

game.showGameDetails();

return 0;

}

OUTPUT:



TASK NO 07:

#include <iostream>

#include <string>

using namespace std;

class Product {

int id;

string name;

int quantity;

public:

Product() {}

Product(int i, string n, int q) : id(i), name(n), quantity(q) {}

string getName() const {

return name;

}

int getID() const {

return id;

}

void display() const {

cout << "Product ID: " << id << ", Name: " << name << ", Quantity: " << quantity << endl;

}

};

class Inventory {

Product products[10];

int count;

public:

Inventory() : count(0) {}

void addProduct(int id, string name, int quantity) {

if (count < 10) {

products[count++] = Product(id, name, quantity);

}

}

void sortByName() {

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

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

if (products[i].getName() > products[j].getName()) {

Product temp = products[i];

products[i] = products[j];

products[j] = temp;

}

}

}

}

void searchByID(int id) const {

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

if (products[i].getID() == id) {

products[i].display();

return;

}

}

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

}

void displayInventory() const {

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

products[i].display();

}

}

};

int main() {

Inventory inventory;

inventory.addProduct(101, "Table", 5);

inventory.addProduct(103, "Laptop", 2);

inventory.addProduct(102, "Chair", 8);

cout << "Before Sorting:\n";

inventory.displayInventory();

inventory.sortByName();

cout << "\nAfter Sorting:\n";

inventory.displayInventory();

cout << "\nSearching for Product ID 102:\n";

inventory.searchByID(102);

return 0;

}

OUTPUT:

