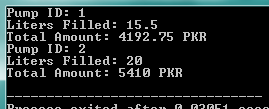
OOPs LAB 05

NAME: Abdul Rahim Dawra RollNo: 24K-0633

## Task 01:

#include <iostream>  
using namespace std;  
  
class PetrolPump {  
 static float pricePerLiter;  
 int pumpID;  
 float litersFilled;  
  
public:  
 PetrolPump(int id, float liters) {  
 pumpID = id;  
 litersFilled = liters;  
 }  
  
 void printReceipt() {  
 float total = litersFilled \* pricePerLiter;  
 cout << "Pump ID: " << pumpID << endl;  
 cout << "Liters Filled: " << litersFilled << endl;  
 cout << "Total Amount: " << total << " PKR" << endl;  
 }  
};  
  
float PetrolPump::pricePerLiter = 270.50;  
  
int main() {  
 PetrolPump p1(1, 15.5);  
 PetrolPump p2(2, 20);  
 p1.printReceipt();  
 p2.printReceipt();  
 return 0;  
}



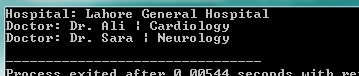
## Task 02:

#include <iostream>  
#include <string>  
using namespace std;  
  
class MobileDevice {  
 string modelName;  
 const string IMEINumber;  
  
public:  
 MobileDevice(string name, string imei) : modelName(name), IMEINumber(imei) {}  
  
 void verifyDevice() const {  
 cout << "Model: " << modelName << endl;  
 cout << "IMEI: " << IMEINumber << endl;  
 }  
};  
  
int main() {  
 MobileDevice m1("Samsung Galaxy A55", "358901245678901");  
 m1.verifyDevice();  
 return 0;  
}



## Task 03:

#include <iostream>  
#include <vector>  
using namespace std;  
  
class Doctor {  
public:  
 string name, specialization;  
 Doctor(string n, string s) : name(n), specialization(s) {}  
};  
  
class Hospital {  
 string hospitalName;  
 vector<Doctor\*> doctors;  
public:  
 Hospital(string name) : hospitalName(name) {}  
  
 void addDoctor(Doctor\* d) {  
 doctors.push\_back(d);  
 }  
  
 void showDoctors() {  
 cout << "Hospital: " << hospitalName << endl;  
 for (auto d : doctors) {  
 cout << "Doctor: " << d->name << " | " << d->specialization << endl;  
 }  
 }  
};  
  
int main() {  
 Doctor d1("Dr. Ali", "Cardiology");  
 Doctor d2("Dr. Sara", "Neurology");  
  
 Hospital h1("Lahore General Hospital");  
 h1.addDoctor(&d1);  
 h1.addDoctor(&d2);  
  
 h1.showDoctors();  
 return 0;  
}



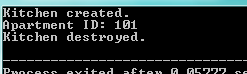
## Task 04:

#include <iostream>  
using namespace std;  
  
class CallRecord {  
 string callerID;  
 int durationMinutes;  
 static int totalCalls;  
  
public:  
 CallRecord(string id = "", int duration = 0) {  
 callerID = id;  
 durationMinutes = duration;  
 totalCalls++;  
 }  
  
 static int getTotalCalls() {  
 return totalCalls;  
 }  
};  
  
int CallRecord::totalCalls = 0;  
  
int main() {  
 CallRecord calls[20];  
 cout << "Total calls logged: " << CallRecord::getTotalCalls() << endl;  
 return 0;  
}



## Task 05:

#include <iostream>  
#include <string>  
using namespace std;  
  
class Kitchen {  
public:  
 Kitchen() { cout << "Kitchen created." << endl; }  
 ~Kitchen() { cout << "Kitchen destroyed." << endl; }  
};  
  
class Apartment {  
 const int apartmentID;  
 Kitchen kitchen;  
  
public:  
 Apartment(int id) : apartmentID(id) {}  
 void showApartment() const {  
 cout << "Apartment ID: " << apartmentID << endl;  
 }  
};  
  
int main() {  
 Apartment a1(101);  
 a1.showApartment();  
 return 0;  
}



## Task 06:

#include <iostream>  
#include <vector>  
using namespace std;  
  
class Employee {  
 string name;  
 static int totalEmployees;  
  
public:  
 Employee(string n) : name(n) { totalEmployees++; }  
 static int getTotalEmployees() { return totalEmployees; }  
};  
  
int Employee::totalEmployees = 0;  
  
class ClientProject {  
 vector<Employee\*> team;  
public:  
 void assignEmployee(Employee\* e) {  
 team.push\_back(e);  
 }  
  
 void showTeam() {  
 cout << "Project Team Members: " << team.size() << endl;  
 }  
};  
  
int main() {  
 Employee e1("Ali"), e2("Sara"), e3("Ahmed");  
  
 ClientProject p1;  
 p1.assignEmployee(&e1);  
 p1.assignEmployee(&e2);  
 p1.assignEmployee(&e3);  
  
 cout << "Total Employees in Company: " << Employee::getTotalEmployees() << endl;  
 return 0;  
}



## Task 07:

#include <iostream>  
#include <vector>  
using namespace std;  
  
class Student {  
public:  
 string name;  
 const int enrollmentID;  
 Student(string n, int id) : name(n), enrollmentID(id) {}  
};  
  
class CourseSection {  
 string sectionName;  
 static int totalSections;  
 vector<Student\*> students;  
  
public:  
 CourseSection(string name) : sectionName(name) { totalSections++; }  
  
 void addStudent(Student\* s) {  
 students.push\_back(s);  
 }  
  
 static void showTotalSections() {  
 cout << "Total Course Sections: " << totalSections << endl;  
 }  
};  
  
int CourseSection::totalSections = 0;  
  
int main() {  
 Student s1("Ali", 101), s2("Sara", 102);  
 CourseSection cs1("BCS-A");  
 cs1.addStudent(&s1);  
 cs1.addStudent(&s2);  
  
 CourseSection::showTotalSections();  
 return 0;  
}

