**Slip 22: Sample Solutions and Explanations**

**Q1. Vehicle Management System with Inheritance**

**Approach**

* Use a base class Vehicle with common attributes and methods.
* Derive a Car class from Vehicle that adds car-specific features (model, doors, maker).
* Demonstrate inheritance and method calls in main.

**Code**

#include <iostream>  
using namespace std;  
// [Base Class: Vehicle]  
class Vehicle {  
protected:  
 int speed, passengers;  
 string fuel;  
public:  
 void go(){ cout << "Vehicle going.\n"; }  
 void stop(){ cout << "Vehicle stopped.\n"; }  
 void changeDirection(){ cout << "Direction changed.\n"; }  
};  
// [Derived Class: Car]  
class Car : public Vehicle {  
 string model; int doors; string maker;  
public:  
 void radio(){ cout << "Radio ON.\n"; }  
 void ac(){ cout << "AC ON.\n"; }  
 void wiper(){ cout << "Wiper ON.\n"; }  
};  
  
int main() {  
 Car c;  
 c.go(); c.radio(); c.ac(); c.stop();  
 return 0;  
}

**Explanation**

* The Car class inherits all methods and data from Vehicle, and adds car-specific features.
* Demonstrates calling both inherited and new methods.

**Syntax Definitions**

* **protected**: Members are accessible in the class and its derived classes.
* **public inheritance**: Derived class inherits all public and protected members of the base class.

**Q2. Student Class: Accept, Display, and Search by Fuel Type**

**Approach**

* Create a Student class with attributes: roll number, name, and fuel type.
* Accept details for n students and store them in a vector.
* Display details of students with a specific fuel type (e.g., "Petrol").

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
// [Student Class Definition]  
class Student {  
 int roll;  
 string name, fuelType;  
public:  
 void accept() {  
 cout << "Roll: "; cin >> roll;  
 cout << "Name: "; cin >> name;  
 cout << "Fuel Type: "; cin >> fuelType;  
 }  
 void display() { cout << roll << " " << name << " " << fuelType << endl; }  
 string getFuelType() { return fuelType; }  
};  
  
int main() {  
 int n;  
 cout << "Number of students: ";  
 cin >> n;  
 vector<Student> students(n);  
 for(auto &s : students) s.accept();  
 string searchFuel;  
 cout << "Enter fuel type to search: "; cin >> searchFuel;  
 cout << "Students with fuel type " << searchFuel << ":\n";  
 for(auto &s : students)  
 if(s.getFuelType() == searchFuel) s.display();  
 return 0;  
}

**Explanation**

* The Student class encapsulates student data and provides methods to accept and display it.
* The program reads n students, then displays those with the specified fuel type.
* The getFuelType method is used for filtering.

**Syntax Definitions**

* **class**: A user-defined type that groups data and functions.
* **vector**: A dynamic array from the C++ Standard Library.

**Q3. Vehicle Management System with Inheritance (Case Study)**

**Approach**

* Use base class Vehicle; derived class Car adds car-specific features.
* Demonstrate inheritance and method calls in main.

**Code**

#include <iostream>  
using namespace std;  
class Vehicle {  
protected:  
 int speed, passengers;  
 string fuel;  
public:  
 void go(){ cout << "Vehicle going.\n"; }  
 void stop(){ cout << "Vehicle stopped.\n"; }  
 void changeDirection(){ cout << "Direction changed.\n"; }  
};  
class Car : public Vehicle {  
 string model; int doors; string maker;  
public:  
 void radio(){ cout << "Radio ON.\n"; }  
 void ac(){ cout << "AC ON.\n"; }  
 void wiper(){ cout << "Wiper ON.\n"; }  
};  
  
int main() {  
 Car c;  
 c.go(); c.radio(); c.ac(); c.stop();  
 return 0;  
}

**Explanation**

* Inherits all methods and data from Vehicle, adds Car-specific features.
* Demonstrates inheritance and method calls for a vehicle management system.

**Syntax Definitions**

* **Inheritance**: Mechanism by which one class acquires the properties and behaviors of another class.
* **Method Overriding**: Redefining a base class method in a derived class.