**Slip 24: Sample Solutions and Explanations**

**Q1. Employee Payroll System (Inheritance: FullTime, PartTime)**

**Approach**

* Use a base class Employee for common data (name, id).
* Two derived classes: FullTimeEmployee and PartTimeEmployee, each with their specifics.
* Accept and display details for both types of employees.

**Code**

#include <iostream>  
using namespace std;  
  
// [Employee Base Class]  
class Employee {  
protected:  
 string name; int id;  
public:  
 void accept() { cout << "Name: "; cin >> name; cout << "ID: "; cin >> id; }  
 void display() { cout << name << " " << id << endl; }  
};  
  
// [FullTimeEmployee Derived Class]  
class FullTimeEmployee : public Employee {  
 double salary;  
public:  
 void accept() { Employee::accept(); cout << "Salary: "; cin >> salary; }  
 void display() { Employee::display(); cout << "FullTime Salary: " << salary << endl; }  
};  
  
// [PartTimeEmployee Derived Class]  
class PartTimeEmployee : public Employee {  
 double rate;  
public:  
 void accept() { Employee::accept(); cout << "Hourly Rate: "; cin >> rate; }  
 void display() { Employee::display(); cout << "PartTime Rate: " << rate << endl; }  
};  
  
int main() {  
 FullTimeEmployee f; PartTimeEmployee p;  
 f.accept(); p.accept();  
 f.display(); p.display();  
 return 0;  
}

**Explanation**

* Each derived class adds its own data but calls base class function for common attributes.
* Demonstrates inheritance and method overriding for payroll management.

**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.

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

**Approach**

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

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
// [Student Class Definition]  
class Student {  
 int roll;  
 string name, empType;  
public:  
 void accept() {  
 cout << "Roll: "; cin >> roll;  
 cout << "Name: "; cin >> name;  
 cout << "Employee Type: "; cin >> empType;  
 }  
 void display() { cout << roll << " " << name << " " << empType << endl; }  
 string getEmpType() { return empType; }  
};  
  
int main() {  
 int n;  
 cout << "Number of students: ";  
 cin >> n;  
 vector<Student> students(n);  
 for(auto &s : students) s.accept();  
 string searchType;  
 cout << "Enter employee type to search: "; cin >> searchType;  
 cout << "Students with employee type " << searchType << ":\n";  
 for(auto &s : students)  
 if(s.getEmpType() == searchType) 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 employee type.
* The getEmpType 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. Employee Payroll System (Inheritance Case Study)**

**Approach**

* Use a base class Employee for common data (name, id).
* Derive FullTimeEmployee and PartTimeEmployee classes for specific attributes.
* Accept and display details for both types of employees.

**Code**

#include <iostream>  
using namespace std;  
  
class Employee {  
protected:  
 string name; int id;  
public:  
 void accept() { cout << "Name: "; cin >> name; cout << "ID: "; cin >> id; }  
 void display() { cout << name << " " << id << endl; }  
};  
class FullTimeEmployee : public Employee {  
 double salary;  
public:  
 void accept() { Employee::accept(); cout << "Salary: "; cin >> salary; }  
 void display() { Employee::display(); cout << "FullTime Salary: " << salary << endl; }  
};  
class PartTimeEmployee : public Employee {  
 double rate;  
public:  
 void accept() { Employee::accept(); cout << "Hourly Rate: "; cin >> rate; }  
 void display() { Employee::display(); cout << "PartTime Rate: " << rate << endl; }  
};  
  
int main() {  
 FullTimeEmployee f; PartTimeEmployee p;  
 f.accept(); p.accept();  
 f.display(); p.display();  
 return 0;  
}

**Explanation**

* Each derived class adds its own data but calls base class function for common attributes.
* Demonstrates inheritance and method overriding for payroll management.

**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.