Q-1: You are developing a geometry calculator application that can calculate the area of different shapes. To achieve this, you create a base class Shape and derived classes Circle and Rectangle.

Sample test case:

|  |
| --- |
| Input: Circle(radius=5)  Output: Area of Circle: 78.5  Input: Rectangle(length= 4, width= 6)  Output: Area of Rectangle: 24 |

Constraints:

* The values of width, length and radius should be positive integers.

Q-2: Imagine you are organizing a grand Multimedia Festival, where various artists and creators will be showcasing their creative works, including music and videos. To manage the multimedia playback during the festival, you decide to create a C++ program that efficiently

handles the media players for different types of content such as audio or video.

Sample test case:

|  |
| --- |
| Input: AudioPlayer("Song1.mp3")  Output: Playing audio: Song1.mp3  Input: VideoPlayer("Movie1.mp4")  Output: Playing video: Movie1.mp4 |

Q-3: Imagine you are working on a smart home control system. The system allows users to control various devices remotely using a centralized control panel. As part of this project, you are tasked with creating a C++ program to implement the functionality of remote

controlling TVs and DVD players. To achieve this, you will define a base class called RemoteControl and two derived classes, TVRemote and DVDRemote.

Sample test case:

|  |
| --- |
| Input:  remote1 = new TVRemote()  remote1->turnOn()  remote1->turnOff()  Output:  TV: Turning ON  TV: Turning OFF  Input:  remote2 = new DVDRemote()  remote2->turnOn()  remote2->turnOff()  Output:  DVD Player: Turning ON  DVD Player: Turning OFF |
|  |

Q-4: Imagine you are working on a payroll management system for a company called "TechCorp." The company has different types of employees, including Regular Employees and Contract Employees. As part of the system, you need to create a C++ program to calculate the pay for each employee based on their employment type.

Sample test case:

|  |
| --- |
| Input:  RegularEmployee(name="John Doe",id=1001,rate=20.0,hours=160)  Output:  Regular Employee Pay: $3520  Input:  ContractEmployee(name="Jane Smith", id=2001, rate=2000, hours=0,contractRate=500)  Output:  Contract Employee Pay: $500 |

Q-5: Adventura town needs a virtual simulation of the thrilling experience of driving different vehicles on their scenic trails. Design a base class called "Vehicle," which will serve as a blueprint for all types of vehicles in Adventura. Additionally, craft two unique derived classes, "Car" and "Bike," to embody specific types of vehicles that adventurers frequently use. Each class in the program has a special function called "drive()." This function represents the exhilarating action of driving the corresponding

vehicle to experience a virtual journey through Adventura's captivating terrains.

Sample test case:

|  |
| --- |
| Input:Car()  Output:Driving car.  Input:Bike()  Output:Riding bike. |

Q-6: Imagine you are developing a virtual pet simulator where users can interact with different types of animals.

To achieve this, you decide to create a C++ program that simulates the behavior of various animals.

The program will include a base class called Animal and two derived classes, Dog and Cat.

In this virtual pet world, users can select a virtual pet of their choice, either a dog or a cat, and interact with them.

Each pet will have its unique characteristics, including the sounds they make. To represent the sounds made by the animals,

each class will have a function called makeSound().

Sample test case:

|  |
| --- |
| Input:  Dog(name="Buddy")  Cat(name="Whiskers")  Output:  Buddy barks: Woof Woof!  Whiskers meows: Meow Meow! |

Q-7: Write a program in which we have four classes: Person, Student, Employee, and Manager. Both Student and Employee inherit virtually from Person. The Manager class is derived from both Student and Employee. Without using a virtual base class, the Manager class would face the diamond problem, as there would be two copies of the Person class in the inheritance hierarchy. By making Person a virtual base class, we resolve the diamond problem, and the manager.display() function correctly calls the Student and Employee versions of display().

Sample test case:

|  |
| --- |
| Input:  manager(1001, 3.8, 5000);  manager.display()  Output: Manager ID: 1001, GPA: 3.8, Salary: 5000 |

Q-8: Develop a graphical user interface (GUI) library with a virtual base class Widget and a derived class Button. Implement a virtual function draw() in the base class and override it in the Button class to display the button on the screen.

Sample test case:

|  |
| --- |
| Input: Widget\* widget = new Button(100, 50, "Click Me!");  widget->draw();  Output: Drawing a button with label "Click Me!" at (100, 50) |

Q-9: Create a program for managing restaurant orders. Design a virtual base class Order and derived class Invoice. Implement a virtual function calculateTotal() in the Order class, and override it in the Invoice class to calculate the total

amount to be paid including tax rate.

Sample test case:

|  |
| --- |
| Input:  Order("Pizza", 2, 12.50)  Order("Burger", 3, 8.75)  Invoice("Pasta", 4, 9.25, 0.1)  Output:  Item: Pizza, Quantity: 2, Price Per Item: $12.5, Total Amount: $25  Item: Burger, Quantity: 3, Price Per Item: $8.75, Total Amount: $26.25  Item: Pasta, Quantity: 4, Price Per Item: $9.25, Total Amount: $40.7 |

Q-10: Create a game development framework with a virtual base class GameObject and derived class Character. Implement a virtual function update() in the GameObject class, and override it in the Character class to update the character's position and other attributes.

Sample test case:

|  |
| --- |
| GameObject\* obj1 = new GameObject(5, 5);  obj1->update(); // Output: Updating GameObject: (5, 5)  Character\* char1 = new Character(10, 10, 100);  char1->update(); // Output: Updating Character: (11, 11), Health: 110 |

Q-11: Create a virtual base class Animal with pure virtual functions displayInfo() and makeSound(). Implement derived classes Lion, Elephant, and Giraffe that inherit from Animal. In a zoo management system, use these classes to display information and make sounds of different animals.

Sample test case:

|  |
| --- |
| Input:  Lion lion("Simba");  animal->displayInfo();  animal->makeSound();  Output:  Name: Simba  Species: Lion  The lion roars! |

Q-12: Create a virtual base class Travel with pure virtual functions bookTicket() and calculateFare(). Implement derived classes Flight, Train, and Bus that inherit from Travel. In a travel booking system, use these classes to book tickets for different modes of travel and calculate the fare accordingly.

Sample test case:

|  |
| --- |
| Input:  Travel\* flight = new Flight("Flight", 2, 300.0);  flight->bookTicket();  flight->calculateFare();  Output:  Booking 2 flight ticket(s) for Flight mode.  Total fare for 2 passengers in Flight mode: $600  Input:  Travel\* train = new Train("Train", 3, 50.0);  train->bookTicket();  train->calculateFare();  Output:  Booking 3 train ticket(s) for Train mode.  Total fare for 3 passengers in Train mode: $150  Input:  Travel\* bus = new Bus("Bus", 4, 20.0);  bus->bookTicket();  bus->calculateFare();  Output:  Booking 4 bus ticket(s) for Bus mode.  Total fare for 4 passengers in Bus mode: $80 |