Q1: You are a theme park manager, and you want to create a program to manage and simulate different **attractions** in your park. You have **four** types of **attractions**: **`RollerCoaster`**, **`WaterRide`**, **`Show`**, and a base class **`Attraction`**. Each attraction has specific properties and behaviors.

Task:

- Create a program that creates instances of **`RollerCoaster`**, **`WaterRide`**, and **`Show`** attractions, adds them to a **`ThemePark`** object, displays information about all attractions in the theme park, and simulates a day in the theme park by simulating a ride for each attraction.

Data Members:

- Attraction class has data members: **name** (string), **`thrillLevel`** (int), **`capacity`** (int), **`duration`** (int). **`RollerCoaster`** has an additional data member: **`drops`** (int). **`WaterRide`** has an additional data member: **`waterUsage`** (double). Show has an additional data member: **`performer`** (string).

Member Functions:

- Attraction class has member functions: **`displayInfo()`** and **`simulateRide()`**. **`RollerCoaster`**, **`WaterRide`**, and **`Show`** classes override these functions. **`ThemePark`** class has member functions: **`addAttraction(related_parameters)`**, **`displayAllAttractions()`**, and **`simulateDay()`**.

Pointers:

- **`rollerCoaster`**, **`waterRide`**, and **`show`** objects **must be pointer objects**. **ThemePark's** attractions array holds pointers to **`Attraction`** objects.

`main()` Function:

- Creates a **`ThemePark`** object, creates instances of **`RollerCoaster`**, **`WaterRide`**, and **`Show`** attractions using dynamic memory allocation, adds these attractions to the **`ThemePark`** object, calls **`displayAllAttractions()`** to display information about all attractions, calls **`simulateDay()`** to simulate a day in the theme park, and **`deletes`** the attraction objects.

Q2: You are a software developer at a real estate company. Your manager has asked you to develop a property bidding system to manage the bidding process for a new property. The system should allow multiple bidders to place bids on the property, and it should display the winning bidder's information if the bid amount is greater than or equal to the reserve price. **However, due to limited resources, only 3 bidders are allowed to participate in the bidding process.**

- The system should have a **`Property`** class with attributes **`address`** (string) and **`reservePrice`** (double). The system should also have a **`Bidder`** class with attributes **`name`** (string) and **`bidAmount`** (double). The system should have a **`PropertyBiddingSystem`** class that manages the bidding process for a property. The **`PropertyBiddingSystem`** class should have an array of **3 Bidder** objects and a **`numBidders`** attribute to keep track of the number of bidders.
- The **`addBidder`** function should add a **`Bidder`** object to the system if there is space available **(i.e., if the number of bidders is less than 3)**. The **`displayWinningBidder`** function should display the winning bidder's name and bid amount if the bid amount is **greater than or equal to the reserve price.** The **displayBiddingSystemInfo** function should display the property address, reserve price, and the information of all bidders.
- Implement a **`friend`** function **`displayPropertyInfo`** in the **`Property`** class to display the property address and reserve price. Create an **abstract** class Bid with pure virtual functions **`placeBid`** and

displayBidInfo. The **Bidder** class should inherit from the **Bid** class and implement the **placeBid** and **displayBidInfo** functions.

Write a C++ program to implement the property bidding system as per the requirements. Use the following input data:

- A Property object with address **"Shah Latif Town"** and reserve price **100000.0**
- Four Bidder objects with names **"Mr. Ubaid Khan"**, **"Mr. Muhammad Khalid"**, **"Mr. Waseem Rauf"**, and **"Mr. Sameer"** and bid amounts **150000.0**, **110000.0**, **130000.0**, and **130000.0** respectively

Note: Only 3 bidders are allowed, so the fourth bidder should not be added to the system.

Q3: **Part A:**

You are a software developer for a transportation company that manages various modes of transportation, including buses, trains, and airplanes. The company has decided to introduce an online booking system that allows customers to book tickets for their preferred mode of transportation. The system should be able to handle different types of vehicles, each with its own attributes (e.g., number of seats, route, altitude). The system should also be able to provide details such as departure time, arrival time, and cost of the ticket.

Write a C++ program that implements the following:

- A base class **Vehicle** with data members **make**, **model**, and **maxSpeed**, and methods **getMake()**, **getModel()**, and **getMaxSpeed()**. The first two returns strings and the third returns and integer to the **main.**
- Derived classes **Bus**, **Train**, and **Airplane** that inherit from **Vehicle** and have additional data members and methods specific to each mode of transportation. The **Bus** class should have data members **numSeats** and **route**, and methods **int getNumSeats()** and **string getRoute()**. The classes **Train** possess the same data member and member functions. The **Airplane** has **numSeats (int)** and **altitude (int)** and the member functions int **getNumSeats()** and int **getAltitude()**.
- A **generic** class **BookingSystem** that can handle any type of **vehicle**, with data members **vehicle**, **departureTime (string)**, **arrivalTime (string)**, and **cost (double)**, and methods **getVehicle()**, **getDepartureTime()**, **getArrivalTime()**, and **getCost()**. All the member functions **returns** respected data members to **main()**.
- In the **main()** function, create instances of **Bus**, **Train**, and **Airplane** and create an **explicit** booking systems for each vehicle using the generic **BookingSystem** class. Print the booking details for each vehicle, including the **make**, **model**, **maxSpeed**, **numSeats**, **route /altitude**, **departure time**, **arrival time**, and **cost**.

Part B:

Additionally, the company has partnered with a travel agency that specializes in booking hotel rooms and resort stays. The agency wants to develop a generic booking system that can handle different types of hotels and resorts. The system should allow customers to book a room and display the details of the booked room.

- Your system should have a base class, **Hotel**, with essential attributes like the hotel's **name**, **location**, and **starRating**. Additionally, there are three derived classes: **LuxuryHotel**, **BudgetHotel**, and **Resort**, each tailored to specific accommodation types.
- The **LuxuryHotel** class extends **Hotel** and includes details such as the **numberOfRooms**, **amenities** like swimming pools and gyms, and **cuisine** options.

- The **`BudgetHotel`** class, another subclass of **`Hotel`**, also contains attributes like **`numberOfRooms`** and **`amenities`**, but it's designed for travelers seeking affordable options, thus offering fewer amenities.

- Lastly, the **`Resort`** class, inheriting from **`Hotel`**, focuses on recreational **`activities`** along with **`numberOfRooms`**, making it suitable for vacationers looking for a leisurely stay.
- Moreover, you need a **generic** booking system, **BookingSystem**, which can handle bookings for any type of hotel. It includes a data member that **`booking`** and respective methods to book a room and display booking information. The method that books the room using **`booking`** must receive a **pointer-based entity from the `main` function.**
- In the **`main`** function, you'll create instances of each type of hotel, specifying their details such as name, location, star rating, number of rooms, amenities, and activities. Then, you'll instantiate **explicit** booking systems tailored to each hotel type and book rooms accordingly, demonstrating the functionality of your system.