Structure Tasks

1. **University Database Management**
   Create a program to manage a university database where each department has multiple professors. Define a `Professor` structure with details like `name`, `ID`, `designation`, `publications`, and `salary`. Use an array of structures to store professor records. Implement functions to:
   - Add a new professor to a department.
   - Update a professor's details.
   - Sort professors by the number of publications.
   - Find and display the professor with the highest salary.
2. **Hospital Patient Management System**
   Design a program that keeps track of patients admitted to a hospital. Define a `Patient` structure with attributes like `name`, `age`, `disease`, `admission_date`, and `bill_amount`. Use an array of structures to store patient records. Implement functions to:
   - Admit a new patient.
   - Discharge a patient and generate a bill summary.
   - Search for a patient by name or disease.
   - Sort patients based on admission date.
3. **Bank Account Management System**
   Implement a banking system where each customer has a `BankAccount` structure containing `account_number`, `name`, `balance`, `account_type`, and `last_transaction`. Use an array of structures and implement functions to:
   - Create a new account.
   - Deposit and withdraw money while ensuring balance constraints.
   - Transfer money between two accounts.
   - Display all customers with an account balance above a given threshold.
4. **Employee Payroll System**
   Develop a payroll management system using a `Employee` structure with fields such as `employee_id`, `name`, `designation`, `basic_salary`, `allowances`, and `tax_deductions`. Use an array of structures and implement functions to:
   - Calculate the net salary of each employee.
   - Display the top 3 highest-paid employees.
   - Sort employees by their salary in descending order.
   - Generate a monthly payroll report.

5. **Library Book Management System**
   Design a library management system using a `Book` structure containing `ISBN`, `title`, `author`, `publisher`, `copies_available`, and `issued_copies`. Implement functions to:

   - Add a new book to the library.
   - Issue a book to a student while checking availability.
   - Return a book and update the count.
   - Display books sorted by the number of copies available.

6. **Online Shopping Cart System**
   Implement an online shopping system using a `Product` structure containing `product_id`, `name`, `category`, `price`, `stock_quantity`, and `rating`. Use an array of structures and implement functions to:

   - Add a new product to the inventory.
   - Purchase a product, ensuring stock is updated accordingly.
   - Find and display the highest-rated product in a given category.
   - Sort products by price in ascending order.

7. **Flight Reservation System**
   Develop a flight reservation system using a `Passenger` structure with details like `passenger_id`, `name`, `flight_number`, `seat_number`, and `ticket_price`. Use an array of structures and implement functions to:

   - Book a ticket and assign a seat.
   - Cancel a ticket and free up a seat.
   - Find all passengers booked for a given flight number.
   - Calculate total revenue generated by a specific flight.

8. **Stock Market Portfolio Management**
   Create a stock portfolio management system where an investor can track multiple stocks. Define a `Stock` structure containing `stock_id`, `company_name`, `shares_held`, `purchase_price`, and `current_price`. Implement functions to:

   - Buy new shares and update the portfolio.
   - Sell shares and calculate profit or loss.
   - Find the stock with the highest gain or loss.
   - Sort stocks based on percentage profit or loss.

9. **Smart Home Device Control System**
   Implement a smart home system using a `Device` structure with attributes `device_id`, `device_name`, `room_name`, `power_status`, and `energy_consumed`. Use an array of structures and functions to:

   - Turn a device on/off.
   - Display total energy consumption of all devices.
   - Find the most power-hungry device.
   - Sort devices based on their energy usage.

10. **Railway Reservation System**
    Develop a railway reservation system where passengers book train tickets. Define a `Passenger` structure with `passenger_id`, `name`, `train_number`, `seat_class`, and `ticket_price`. Implement functions to:

- Reserve a seat based on availability.
- Cancel a reservation and free up the seat.
- Find all passengers traveling on a specific train.
- Calculate total revenue generated by a train.
