PROJECT LIST

Projects should create, read & update records using Arrays & Structures. Records should be stored in files.

1. Movie Ticket System

Objective:

Develop a movie ticket seller application using C programming. The application should allow users to view a seating arrangement, purchase tickets for available seats, and visually track sold and available seats using a 2D array.

Requirements:

1. Use a 2D array to represent the seating chart (e.g., 5 rows × 10 columns for 50 seats).
    - Each element in the array should indicate whether the seat is available or sold (e.g., 0 for available, 1 for sold).
2. Functionalities:
    - Implement a function to display the current seat layout using a clear visual format (e.g., O for available, X for sold).
    - Implement a function to allow users to purchase a ticket by selecting a seat (row and column).
    - Implement a function to validate user input (e.g., valid row/column, checking if the seat is already taken).
    - Implement a function to display the total number of available and sold tickets.
3. Add the ability to reset or initialize the seating chart at the beginning of each session.
4. Optionally, implement a function to save the current seating status to a file and load it on the next run.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper input validation and user-friendly messages (e.g., if the user tries to buy a seat that’s already sold).
- Display row and column indices clearly to make seat selection easier.
- Keep the code modular, clean, and easy to read and maintain.

Deliverables:

- Source code files.
- A brief report explaining the design choices, such as the seating array size, the display method, and how ticket purchases are handled.

2. CGPA Counter System

Objective:
Develop a CGPA counter application using C programming. The application should allow students to input their course marks semester-wise and calculate the SGPA (Semester GPA) for each semester and the cumulative CGPA up to the current semester.

Requirements:

1. Define a structure to store course information, including course name, credit hours, and obtained marks (or grade point).
2. Use a 2D array or array of structures to manage multiple semesters and courses per semester.
3. Functionalities:
    - Implement a function to input course details and marks for each semester.
    - Implement a function to calculate SGPA for a given semester using the formula:
      SGPA = (sum of (grade point × credit) for all courses) / (total credits of the semester)
      (Assume a grade point mapping like: A = 4.0, B = 3.0, etc., or based on numeric marks ranges).
    - Implement a function to calculate CGPA up to the current semester:
      CGPA = (sum of all (grade point × credit) for all semesters) / (sum of all credits)
    - Implement a function to display the SGPA of each semester and the overall CGPA.
4. Allow input for multiple semesters and multiple courses per semester (e.g., up to 8 semesters, 6 courses per semester).
5. Optionally, add a function to save and load the data from a file to continue in the future.
6. Create a simple text-based menu to guide users through input and viewing options.

Guidelines:

- Ensure proper input validation (e.g., valid grade point, credit range, number of courses).
- Display calculated results with proper formatting (e.g., SGPA up to 2 decimal places).
- Maintain a clean and modular codebase to improve readability and debugging.

Deliverables:

- Source code files.

- A brief report explaining the design decisions, such as grade mapping, data structure choice, and CGPA/SGPA calculation methodology.

3. Tic Tac Toe Game

Objective:
Develop a simple two-player Tic Tac Toe game using C programming. The application should allow two users to play against each other in multiple rounds, track individual player wins, and offer rematches after each game.

Requirements:

1. Use a 2D array (3×3) to represent the game board.
2. Functionalities:
    - Implement a function to display the current game board clearly using characters like X and O.
    - Implement a function to allow players to make a move by selecting a row and column.
    - Validate input to ensure it's within range and the selected cell is empty.
    - Alternate turns between Player 1 (X) and Player 2 (O).
    - Implement a function to check for a winner (three identical symbols in a row, column, or diagonal).
    - Implement a function to detect a draw when all cells are filled and no winner is found.
    - Keep count of how many times each player has won.
    - After each game, offer a rematch option:
        - If the players agree, reset the board and continue.
        - If they decline, display the final win counts and end the program.
3. Display a summary after each game, showing the current win tally for both players.

Guidelines:

- Ensure proper input validation and user-friendly prompts.
- Maintain game state in a loop to support multiple matches until players choose to exit.
- Use modular code with separate functions for game logic, board display, input handling, and win checking.
- Keep the interface simple, text-based, and intuitive for smooth gameplay.

Deliverables:

- Source code files.
- A brief report explaining:
    - The game logic and structure,
    - How player turns and winning conditions are handled,

        - How the rematch system and win tracking are implemented.

4. Scientific Calculator

Objective:
Develop a scientific calculator application using C programming. The calculator should support a wide range of arithmetic and scientific operations, providing a comprehensive set of functions similar to a physical scientific calculator.

Requirements:

1. Design a user-friendly text-based interface that allows users to choose from a list of operations.
2. Functionalities:
   Implement functions to perform the following operations:
   - Basic Arithmetic:
     - Addition, Subtraction, Multiplication, Division
   - Advanced Mathematical Functions:
     - Power (x^y), Square root, Cube root
     - Logarithm (log, ln)
     - Exponential (e^x)
   - Trigonometric Functions:
     - Sine (sin), Cosine (cos), Tangent (tan)
     - Inverse Trigonometric functions (asin, acos, atan)
     - Hyperbolic functions (sinh, cosh, tanh)
   - Factorial Calculation
   - Modulo Operation
   - Absolute Value
3. Use the math.h library for trigonometric and logarithmic calculations.
4. Input Handling:
   - Prompt users to enter required values based on the selected operation.
   - Validate input (e.g., no division by zero, valid domain for logarithmic and trigonometric functions).
5. Allow the user to perform multiple calculations in a loop until they choose to exit.
6. Optionally, maintain a simple history of the last few operations (in memory).

Guidelines:

- Keep the code modular: use separate functions for each category of operations.
- Use clear menus and instructions to guide users through operation selection and input.

- Handle mathematical edge cases gracefully (e.g., invalid inputs, overflow).
- Use switch or if-else statements for menu and operation selection.

Deliverables:

- Source code files.
- A brief report explaining:
    - The overall program structure,
    - How each operation is implemented,
    - Any input validation or error handling techniques used**.**

5. Simplified Mobile Phonebook

Objective:

Develop a simplified mobile phonebook application using C programming. The application should allow users to create, read, update, and store contact records.

Requirements:

1. Define a structure to store contact details, including name, phone number, and email.
2. Use an array to manage multiple contacts.
3. **Functionalities**:
   - Implement a function to add a new contact to the array.
   - Implement a function to display all contacts.
   - Implement a function to search for a contact by name or phone number.
   - Implement a function to update an existing contact's details.
4. Implement functions to save the contact list to a file and load it from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

Ensure proper error handling and input validation.

Design the user interface to be intuitive and user-friendly.

Optimize the code for readability and maintainability.

Deliverables:

Source code files.

A brief report explaining the design choices and implementation details.

6. Patient Records Management System

Objective:

Develop a patient records management system using C programming. The application should allow users to create, read, update, and store patient records.

Requirements:

1. **Define a structure** to store patient details, including name, age, gender, contact number, and medical history.
2. **Use an array** to manage multiple patient records.
3. **Functionalities:**
    - Implement a function to add a new patient record to the array.
    - Implement a function to display all patient records.
    - Implement a function to search for a patient by name or contact number.
    - Implement a function to update an existing patient's details.
4. Implement functions to save the patient records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

7. Library Book Database Management System

Objective:

Develop a library book database management system using C programming. The application should allow users to create, read, update, and store book records.

Requirements:

1. **Define a structure** to store book details, including title, author, ISBN, publication year, and availability status.
2. **Use an array** to manage multiple book records.
3. **Functionalities:**
    - Implement a function to add a new book record to the array.
    - Implement a function to display all book records.

    - Implement a function to search for a book by title or ISBN.
    - Implement a function to update an existing book's details.
4. Implement functions to save the book records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

8. Payroll Record Management System

Objective:

Develop a payroll record management system using C programming. The application should allow users to create, read, update, and store employee payroll records.

Requirements:

1. **Define a structure** to store payroll details, including employee ID, name, position, salary, and hours worked.
2. **Use an array** to manage multiple payroll records.
3. **Functionalities:**
    - Implement a function to add a new payroll record to the array.
    - Implement a function to display all payroll records.
    - Implement a function to search for a payroll record by employee ID or name.
    - Implement a function to update an existing payroll record's details.
4. Implement functions to save the payroll records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.

- A brief report explaining the design choices and implementation details.

9. Hotel Room Reservation System

Objective:
Develop a hotel room reservation system using C programming. The application should allow users to create, read, update, and store room reservation records.

Requirements:

1. **Define a structure** to store reservation details, including guest name, room number, check-in date, check-out date, and reservation status.
2. **Use an array** to manage multiple reservation records.
3. **Functionalities:**
    - Implement a function to add a new reservation record to the array.
    - Implement a function to display all reservation records.
    - Implement a function to search for a reservation by guest name or room number.
    - Implement a function to update an existing reservation's details.
4. Implement functions to save the reservation records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

10. Restaurant Order Management System

Objective:
Develop a restaurant order management system using C programming. The application should allow users to create, read, update, and store order records.

Requirements:

1. **Define a structure** to store order details, including order ID, table number, items ordered, quantity, and total price.
2. **Use an array** to manage multiple order records.
3. **Functionalities:**
    - Implement a function to add a new order record to the array.
    - Implement a function to display all order records.
    - Implement a function to search for an order by order ID or table number.
    - Implement a function to update an existing order's details.
4. Implement functions to save the order records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

11. Flight Reservation System

Objective:

Develop a flight reservation system using C programming. The application should allow users to create, read, update, and store flight reservation records.

Requirements:

1. **Define a structure** to store reservation details, including passenger name, flight number, departure date, destination, and seat number.
2. **Use an array** to manage multiple reservation records.
3. **Functionalities:**
    - Implement a function to add a new reservation record to the array.
    - Implement a function to display all reservation records.
    - Implement a function to search for a reservation by passenger name or flight number.
    - Implement a function to update an existing reservation's details.
4. Implement functions to save the reservation records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

12. E-commerce Order Management System

Objective:

Develop an e-commerce order management system using C programming. The application should allow users to create, read, update, and store order records.

Requirements:

1. **Define a structure** to store order details, including order ID, customer name, product name, quantity, price, and order status.
2. **Use an array** to manage multiple order records.
3. **Functionalities:**
    - Implement a function to add a new order record to the array.
    - Implement a function to display all order records.
    - Implement a function to search for an order by order ID or customer name.
    - Implement a function to update an existing order's details.
4. Implement functions to save the order records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

13. Cricket Team Players’ Performance Record System

Objective:

Develop a cricket team players’ performance record system using C programming. The application should allow users to create, read, update, and store performance records of players.

Requirements:

1. **Define a structure** to store player performance details, including player name, matches played, runs scored and wickets taken.
2. **Use an array** to manage multiple player performance records.
3. **Functionalities:**
    - Implement a function to add a new player performance record to the array.
    - Implement a function to display all player performance records.
    - Implement a function to search for a player by name.
    - Implement a function to update an existing player’s performance details.
4. Implement functions to save the player performance records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

14. Movie Rating Database Management System

Objective:

Develop a movie rating database management system using C programming. The application should allow users to create, read, update, and store movie rating records.

Requirements:

1. **Define a structure** to store movie rating details, including movie title, director, release year and rating.
2. **Use an array** to manage multiple movie rating records.
3. **Functionalities:**
    - Implement a function to add a new movie rating record to the array.

    - Implement a function to display all movie rating records.
    - Implement a function to search for a movie by title or director.
    - Implement a function to update an existing movie rating's details.
4. Implement functions to save the movie rating records to a file and load them from a file.
5. Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.
- A brief report explaining the design choices and implementation details.

15. Bank Management System

Objective:

Develop a bank management system using C programming. The application should allow users to manage customer accounts, perform transactions, and store account records.

Requirements:

- Define a structure to store customer account details such as account number, name, balance, and transaction history.
- Use an array to manage multiple customer accounts.

Functionalities:

- Implement a function to add a new account to the array.
- Implement a function to display all account details.
- Implement a function to search for an account by account number or name.
- Implement a function to deposit or withdraw money from an account.
- Implement functions to save account records to a file and load them from a file.
- Create a simple text-based menu to navigate the above functionalities.

Guidelines:

- Ensure proper error handling and input validation.
- Design the user interface to be intuitive and user-friendly.
- Optimize the code for readability and maintainability.

Deliverables:

- Source code files.

- A brief report explaining the design choices and implementation details.

16. Department Store Management System

Objective:
Create a department store management system using C that handles product inventory, sales, and billing operations.

Requirements:
- Define a structure to manage product details including name, price, and quantity.
- Use an array to handle multiple products.

Functionalities:
- Implement a function to add new products to the system.
- Implement a function to update stock quantities.
- Implement a function to display all product information.
- Implement a function to generate sales bills.
- Implement functions to save product records to a file and load them from a file.
- Provide a menu to navigate product and sales management.

Guidelines:
Focus on efficient inventory management and accurate billing.

Deliverables:
Source code and a brief explanation of the design.

17. Employee Record System

Objective:
Develop an employee record management system to store and manage employee information using C programming.

Requirements:
- Define a structure to store employee details such as ID, name, position, and salary.
- Use an array to manage multiple employee records.

Functionalities:
- Implement a function to add new employee records.
- Implement a function to display all employee records.
- Implement a function to search for employees by ID or name.
- Implement a function to update employee details.
- Implement functions to save employee records to a file and load them from a file.
- Provide a text-based menu for navigation.

Guidelines:
Ensure secure data management and efficient searching.

Deliverables:
Source code and an explanation of how the system works.

18. Student Record System

Objective:
Develop a student record system using C programming that allows managing student details, marks, and attendance.

Requirements:

- Define a structure to store student details such as ID, name, marks, and attendance.
- Use an array to manage multiple student records.

Functionalities:

- Implement a function to add and update student records.
- Implement a function to display student records.
- Implement a function to search for students by ID or name.
- Implement functions to save and load records from a file.
- Provide a menu for navigating through functionalities.

Guidelines:
Ensure efficient student data storage and quick retrieval.

Deliverables:
Source code files and a report explaining the design and functionality.

19. Bookshop Management System

Objective:
Create a bookshop management system using C programming to manage book inventory, sales, and billing.

Requirements:

- Define a structure to store book details such as title, author, price, and stock.
- Use an array to manage multiple books.

Functionalities:

- Implement a function to add new books to the inventory.
- Implement a function to display available books.
- Implement a function to generate sales bills.
- Implement a function to update book stock after sales.

- Implement functions to save and load inventory data from a file.
- Create a menu for bookshop management.

Guidelines:
Ensure proper inventory management and accurate billing.

Deliverables:
Source code files and a report explaining functionality.

20. Bus Reservation System

Objective:
Develop a bus reservation system using C programming to manage seat reservations, cancellations, and schedules.

Requirements:

- Define a structure to store bus details such as bus number, destination, and available seats.
- Use an array to manage multiple bus schedules.

Functionalities:

- Implement a function to reserve seats on a bus.
- Implement a function to cancel seat reservations.
- Implement a function to display bus schedules and available seats.
- Implement functions to save and load bus data from a file.
- Provide a menu for navigating the system.

Guidelines:
Ensure efficient seat management and accurate reservation data.

Deliverables:
Source code files and a report explaining the system.