Finding alphabet occupancy

```
1. Initialize an array of size 26 to store alphabet occupancy.
2. Iterate through the input string.
3. For each character in the string:
   a. Convert it to lowercase.
   b. If it's an alphabet character ('a' to 'z'), increment the corresponding count in the array.
4. Display the alphabet occupancy array.
5. Exit.
```

Finding duplicate elements

```
1. Create an empty set or dictionary to store unique elements.
2. Iterate through the input list or array.
3. For each element in the list:
   a. If the element is already in the set/dictionary, it's a duplicate.
   b. Otherwise, add it to the set/dictionary.
4. Display or return the duplicate elements found.
5. Exit.
```

File management system

1. Initialize a file management system with root directory.
2. Define functions for file creation, deletion, moving, and listing.
3. Implement a menu-driven interface for users to interact with the system.
4. Handle user commands to perform file operations.
5. Exit.

Fast tag working

1. Initialize a data structure (e.g., a hash table) for storing tags and associated data.
2. Implement functions for adding, updating, or retrieving data based on tags.
3. Create a user interface to allow users to perform tag-related actions.
4. Handle user commands to manipulate tags and data.
5. Exit.

Music playlist

1. Create a playlist data structure, e.g., a linked list or array.
2. Implement functions to add songs, remove songs, shuffle, and play the playlist.
3. Build a user interface to interact with the playlist.
4. Handle user commands for managing and playing songs.
5. Exit.

Student entry management

1. Define a student data structure with attributes (e.g., name, ID, grades).
2. Implement functions for adding, updating, and deleting student records.
3. Create a user interface for administrators to manage student entries.
4. Handle user commands to perform student record operations.
5. Exit.

Book recommendation system

1. Initialize a database of books with attributes (e.g., title, author, genre).
2. Implement a user profile system to track user preferences.
3. Analyze user behavior to recommend books based on their preferences and reading history.
4. Present recommended books to users.
5. Handle user feedback to improve recommendations.
6. Exit.

Vending machines
1. Initialize the vending machine with available products and their prices.
2. Create a user interface for selecting products and processing payments.
3. Accept user input for product selection and payment.
4. Dispense the selected product if payment is sufficient.
5. Return change if necessary.
6. Update inventory after each transaction.
7. Exit.

Attendance management
1. Create a database of students and classes with attendance records.
2. Implement functions for taking attendance, marking absentees, and generating reports.
3. Develop a user interface for teachers or administrators to manage attendance.
4. Handle attendance-related operations and reporting.
5. Exit.

Undo redo

1. Initialize data structures to store the state of an application or system.
2. Implement functions for performing actions and recording their state.
3. Create user interface controls for undo and redo functionality.
4. Handle user commands to undo or redo actions.
5. Exit.

Browser history

1. Maintain a data structure (e.g., a stack or list) to store visited URLs.
2. Implement functions for adding, navigating, and managing the browsing history.
3. Create user interface controls for navigating back and forth.
4. Handle user interactions to navigate the browsing history.
5. Exit.

Phone directory
1. Build a database of contacts with attributes (e.g., name, phone number).
2. Implement functions for adding, editing, deleting, and searching for contacts.
3. Create a user interface for managing contacts.
4. Handle user commands to perform contact-related operations.
5. Exit.

Patient management system
1. Create a database of patient records with attributes (e.g., name, medical history).
2. Implement functions for adding, updating, and retrieving patient information.
3. Develop a user interface for healthcare professionals to manage patient records.
4. Handle user commands for patient management and data access.
5. Exit.

customer support system management
1. Establish a system for tracking customer support requests.
2. Implement functions for creating, updating, and resolving support tickets.
3. Build a user interface for support agents to manage customer issues.
4. Handle customer support requests and agent interactions.
5. Exit.

Cart management for ecomerce
1. Create a shopping cart data structure with items and quantities.
2. Implement functions for adding, removing, and updating cart items.
3. Develop a user interface for online shoppers to manage their carts.
4. Handle user interactions for shopping cart operations.
5. Exit.

College allotment system
1. Design a system for allocating courses, classes, or dormitories to college students.
2. Implement functions for student registration and allocation.
3. Build a user interface for administrators and students to interact with the allocation system.
4. Handle student registration and allocation processes.
5. Exit.

Car reccommendation system
1. Create a database of car models with attributes (e.g., make, price, features).
2. Implement functions for recommending cars based on user preferences.
3. Develop a user interface for users to specify their preferences.
4. Provide car recommendations based on user input.
5. Exit.

Seminar hall management system
1. Maintain a database of seminar halls with details (e.g., capacity, availability).
2. Implement functions for booking, canceling, and scheduling events in halls.
3. Create a user interface for event organizers to manage seminar hall bookings.
4. Handle booking and scheduling of seminar halls.
5. Exit.

Team nagement system

1. Establish a system for organizing and managing teams within an organization.
2. Implement functions for creating, modifying, and assigning team members.
3. Build a user interface for team leaders or managers to oversee team activities.
4. Handle team creation, member assignments, and team-related tasks.
5. Exit.

Hotel booking
1. Set up a hotel reservation system with information about rooms, rates, and availability.
2. Implement functions for booking rooms, checking in, and checking out.
3. Create a user interface for guests to reserve rooms.
4. Handle room bookings, check-ins, and check-outs.
5. Exit.

Parking allotment

1. Create a parking management system with information about parking spots and availability.
2. Implement functions for reserving, releasing, and tracking parking spots.
3. Build a user interface for users to reserve parking spaces.
4. Handle parking spot reservations and releases.
5. Exit.

Restaurant billing system

1. Design a restaurant billing system with menus, orders, and pricing information.
2. Implement functions for taking orders, calculating bills, and processing payments.
3. Develop a user interface for restaurant staff to manage orders and bills.
4. Handle order placement, bill calculation, and payment processing.
5. Exit.

Emailing app

1. Create an email application with features for composing, sending, receiving, and managing emails.
2. Implement functions for composing, sending, and receiving emails.
3. Build a user interface for email interactions.
4. Handle email composition, sending, receiving, and organization.
5. Exit.