The Hotel Management System

The provided C code implements a simple hotel management system that allows for booking rooms, viewing, editing, deleting, and searching customer records. It includes user authentication and uses file handling for data persistence.

Key Components

1. Libraries Used

- **`stdio.h`**: Standard Input/Output functions.
- **`string.h`**: String manipulation functions.
- **`ctype.h`**: Character handling functions.
- **`stdlib.h`**: General utility functions.

2. Structure Definition

- **`struct CustomerDetails`**: Holds customer information.
  - `roomnumber[10]`
  - `name[20]`
  - `address[25]`
  - `phonenumber[15]`
  - `nationality[15]`
  - `email[20]`
  - `period[10]`
  - `arrivaldate[10]`

3. Function Declarations

- **`add()`**: Adds a new customer record.
- **`list()`**: Lists all customer records.
- **`edit()`**: Edits an existing customer record.
- **`delete1()`**: Deletes a customer record.
- **`search()`**: Searches for a customer record by room number.
- **`login()`**: Handles user login authentication.
- **`getCustomerDetails()`**: Collects customer details from the user.
- **`printCustomerDetails()`**: Displays customer details.
- **`clearBuffer()`**: Clears the input buffer.

Detailed Functionality

1. User Authentication (`login()`)

- Prompts for username and password.
- Allows up to three attempts.
- Checks credentials against hardcoded values (`user` and `pass`).
- Exits the program after three failed attempts.

2. Main Menu (`main()`)

- Displays options:
    1. Book a room
    2. View customer records
    3. Delete customer record
    4. Search customer record
    5. Edit record
    6. Exit
- Reads user choice and calls the corresponding function.
- Uses a loop to allow multiple operations until the user exits.

3. Adding a Customer (`add()`)

- Opens `add.txt` in append mode.
- Repeatedly collects customer details using `getCustomerDetails()`.
- Writes the details to the file.
- Allows the user to add multiple records or quit.

4. Listing Customers (`list()`)

- Opens `add.txt` in read mode.
- Displays all customer records in a formatted table.
- Uses `printf()` to align data in columns.

5. Deleting a Customer (`delete1()`)

- Opens `add.txt` and a temporary file `temp.txt`.
- Prompts for the room number to delete.
- Copies all records except the one to delete to `temp.txt`.
- Replaces `add.txt` with `temp.txt`.
- Informs the user whether the deletion was successful.

6. Searching for a Customer (`search()`)

- Opens `add.txt` in read mode.
- Prompts for the room number to search.
- Reads records until a match is found.
- Displays the customer's details if found.
- Notifies the user if the record is not found.

7. Editing a Customer Record (`edit()`)

- Opens `add.txt` in read/write mode.
- Prompts for the room number to edit.
- Reads records until a match is found.
- Uses `fseek()` to move the file pointer back one record.
- Overwrites the record with new details collected via `getCustomerDetails()`.
- Notifies the user whether the update was successful.

Helper Functions

`getCustomerDetails()`

- Collects customer details safely.
- Uses `fgets()` for string inputs to prevent buffer overflow.
- Removes the trailing newline character using `strtok()`.

`printCustomerDetails()`

- Displays the customer's information in a readable format.

`clearBuffer()`

- Flushes the input buffer.
- Ensures no residual input affects subsequent input operations.

Input Handling Improvements

- **Buffer Clearing**: Ensures that input buffers are cleared to prevent unintended input consumption.
- **Safe String Input**: Replaces `scanf()` with `fgets()` for strings to prevent buffer overflows.
- **Newline Removal**: Uses `strtok()` to remove the newline character from strings read with `fgets()`.

File Handling

- **Data Storage**: Customer records are stored in `add.txt`.
- **Temporary Files**: Uses `temp.txt` as a temporary file during deletion operations.
- **Binary File Mode**: Uses binary mode for file operations (`"a+"`, `"r"`, `"r+"`) with structures.

User Interface

- **Menus and Prompts**: Provides clear instructions and feedback to the user.
- **Formatting**: Aligns data in tables for better readability.
- **Feedback Messages**: Informs the user about the success or failure of operations.

Error Handling

- Checks if files open successfully before proceeding.
- Prints error messages if file operations fail.
- Uses return statements to exit functions when errors occur.

Security Considerations

- **Basic Authentication**: Uses hardcoded credentials, which is not secure for real applications.
- **Input Validation**: Limited input validation; assumes correct input format.

Potential Enhancements

- **Dynamic Memory Allocation**: To handle inputs of variable lengths.
- **Improved Authentication**: Replace hardcoded credentials with a secure authentication mechanism.
- **Data Validation**: Add checks to validate input data (e.g., valid email format, numeric values where expected).
- **Concurrency Handling**: Implement file locks for multi-user environments.
- **User Experience**: Enhance the interface with better error messages and possibly a GUI.

Conclusion

The code provides a foundational system for managing hotel customer records with basic functionalities. It demonstrates file handling, user input management, and struct usage in C. While functional, the system can be improved in terms of security, input validation, and user experience for practical deployment.