A detailed explanation of the code.

1. Structure Definition:

The code starts by defining a structure called `Cuisine`. This structure represents a cuisine and contains the following fields:

- `name`: A character array to store the name of the cuisine.

- `description`: A character array to store the description of the cuisine.

- `region`: A character array to store the region of the cuisine.

- `prev`: A pointer to the previous cuisine in the doubly linked list.

- `next`: A pointer to the next cuisine in the doubly linked list.

2. Global Variables:

The code declares a global variable `head` of type `Cuisine\*`. This variable serves as the head (starting point) of the doubly linked list of cuisines.

3. Function: `insertEnd`:

The `insertEnd` function is used to insert a new cuisine at the end of the doubly linked list. It takes the address of the `head` pointer, along with the name, description, and region of the cuisine to be inserted. The function performs the following steps:

- Allocates memory for a new `Cuisine` node.

- Copies the provided name, description, and region into the respective fields of the new node.

- If the `head` pointer is NULL (i.e., the list is empty), it sets the new node as the head and updates its `prev` and `next` pointers to point to itself, creating a circular list.

- If the list is not empty, it finds the last node in the list by accessing the `prev` pointer of the `head` node.

- Updates the pointers of the last node and the new node to insert the new cuisine at the end of the list.

4. Function: `displayCuisines`:

The `displayCuisines` function is responsible for displaying the details of all the cuisines in the doubly linked list. It takes the `head` of the list as input and performs the following steps:

- Checks if the `head` pointer is NULL, indicating an empty list. If so, it prints a message stating that no cuisines are available.

- If the list is not empty, it initializes a temporary pointer `temp` to the `head` node.

- Enters a loop that iterates through the list in a circular manner.

- Prints the name, description, and region of the current cuisine pointed to by `temp`.

- Moves `temp` to the next node in the list until it reaches the `head` node again, completing one full iteration of the circular list.

5. Function: `recommendCuisines`:

The `recommendCuisines` function reads cuisine details from a file and recommends cuisines based on the given filter options. It takes a filename, preference, region, and filter option as input and performs the following steps:

- Opens the specified file in read mode.

- Checks if the file opening was successful. If not, it prints an error message and returns.

- Initializes variables for storing cuisine details and a flag to track if any cuisines matching the filter criteria are found.

- Reads cuisine details from the file in a line-by-line manner using `fgets`.

- Processes each line of cuisine details and applies the specified filter option:

- Filter Option 1: Regional Filter

- Compares the read cuisine region with the provided region. If they match, it prints the cuisine details and sets the flag indicating a match.

- Filter Option 2: Preference Filter

- Searches for the preference string within the cuisine description. If found, it prints the

cuisine details and sets the flag indicating a match.

- Filter Option 3: Regional and Preference Filter

- Combines the previous two filter options, checking both the region and preference. If both match, it prints the cuisine details and sets the flag indicating a match.

- After processing all cuisines, if no matches were found (flag remains unset), it prints a message indicating that no cuisines matching the filter criteria were found.

- Closes the file.

6. Function: `deallocateCuisines`:

The `deallocateCuisines` function is responsible for deallocating the memory used by the doubly linked list of cuisines. It takes the address of the `head` pointer as input and performs the following steps:

- Checks if the `head` pointer is NULL, indicating an empty list. If so, it returns without performing any deallocation.

- Initializes two pointer variables, `current` and `next`, to keep track of the current and next nodes during the deallocation process.

- Enters a loop that iterates through the list in a circular manner.

- Updates `next` to store the pointer to the next node.

- Frees the memory occupied by the current node pointed to by `current`.

- Moves `current` to the next node (`next`) and repeats the process until it reaches the `head` node again, completing one full iteration of the circular list.

- Sets the `head` pointer to NULL, indicating an empty list.

7. Function: `manager`:

The `manager` function represents the manager role. It prompts the user to enter a username and password, performs a basic login check, and executes tasks related to the manager's role. The function performs the following steps:

- Prompts the user to enter a username and password.

- Performs a basic login check by comparing the entered username and password with hardcoded values. If the login fails, it prints an error message and returns.

- If the login is successful, the manager can enter the number of cuisines to be added.

- Opens a file in append mode to store the cuisine details.

- In a loop, prompts the manager to enter the name, description, and region of each cuisine.

- Writes the entered cuisine details to the file.

- Calls the `insertEnd` function to insert the cuisine into the doubly linked list.

- After adding all the cuisines, it closes the file.

- Displays all the cuisines using the `displayCuisines` function.

8. Function: `customer`:

The `customer` function represents the customer role. It prompts the user to select a filter option (by region, preference, or both) and recommends cuisines based on the selected filter. The function performs the following steps:

- Displays the available filter options.

- Prompts the user to enter the desired filter option.

- Based on the selected filter option, prompts the user to enter the region and/or preference for filtering the cuisines.

- Calls the `recommendCuisines` function, passing the appropriate filter option, region, and preference.

- The recommended cuisines are displayed on the console.

9. `main` Function:

The `main` function is the entry point of the program. It performs the following steps:

- Declares a variable `role` to store the user's role (manager, customer, or exit).

- Enters a loop that repeatedly prompts the user to enter their role until the user selects the exit option.

- Based on the selected role, it calls the corresponding function (`manager` or `customer`).

- After the loop ends, it calls the `de

allocateCuisines` function to free the memory used by the doubly linked list.

- Prints a goodbye message and terminates the program.