Vehicle Inspection Management System with Detailed Explanation

This project implements a Vehicle Inspection Management System using C programming.

It uses key concepts like file handling, dynamic memory allocation, pointers,

linked lists, stacks, queues, and modular functions to simulate and organize the

process of vehicle inspections.

Below, the complete code is provided with comments explaining the functionality of each part.

File: inspection_system.h

```c
/*
    inspection_system.h - Header File

    This file declares the data structures and function prototypes used in the
    Vehicle Inspection Management System.
*/

#ifndef INSPECTION_SYSTEM_H
#define INSPECTION_SYSTEM_H

// Structures
typedef struct Vehicle {
    char licensePlate[10]; // License plate of the vehicle
    char ownerName[50];    // Name of the owner
    char model[30];        // Model of the vehicle
    int year;              // Year of manufacture
    struct Vehicle* next;  // Pointer to the next vehicle in the linked list
} Vehicle;
```

```c
typedef struct StackNode {
    char licensePlate[10]; // License plate of the vehicle
    struct StackNode* next; // Pointer to the next stack node
} StackNode;

typedef struct QueueNode {
    char licensePlate[10]; // License plate of the vehicle
    struct QueueNode* next; // Pointer to the next queue node
} QueueNode;

typedef struct Queue {
    QueueNode* front; // Front of the queue
    QueueNode* rear;  // Rear of the queue
} Queue;

// Function prototypes
Vehicle* addVehicle(Vehicle* head, char* licensePlate, char* ownerName, char* model, int
year);
void displayVehicles(Vehicle* head);
void saveToFile(Vehicle* head, const char* filename);
Vehicle* loadFromFile(const char* filename);
void scheduleInspectionLIFO(StackNode** top, char* licensePlate);
void processInspectionLIFO(StackNode** top);
void scheduleInspectionFIFO(Queue* queue, char* licensePlate);
void processInspectionFIFO(Queue* queue);

#endif
```

File: inspection_system.c

```c
/*
```

```
    inspection_system.c - Implementation File

    This file implements the functions declared in the header file.
    These functions handle operations for managing vehicles,
    scheduling inspections using LIFO and FIFO, and file handling.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "inspection_system.h"

// Add vehicle to the linked list
Vehicle* addVehicle(Vehicle* head, char* licensePlate, char* ownerName, char* model, int
year) {
    Vehicle* newVehicle = (Vehicle*)malloc(sizeof(Vehicle));
    if (newVehicle == NULL) {
        printf("Memory allocation failed!\n");
        return head;
    }
    strcpy(newVehicle->licensePlate, licensePlate);
    strcpy(newVehicle->ownerName, ownerName);
    strcpy(newVehicle->model, model);
    newVehicle->year = year;
    newVehicle->next = head; // Insert at the beginning of the list
    return newVehicle;
}

// Display all vehicles in the linked list
void displayVehicles(Vehicle* head) {
    if (head == NULL) {
        printf("No vehicles to display.\n");
        return;
    }
    Vehicle* current = head;
    while (current != NULL) {
        printf("License Plate: %s, Owner: %s, Model: %s, Year: %d\n",
                                     current->licensePlate,  current->ownerName,  current->model,
current->year);
        current = current->next;
    }
}

// Save vehicle data to a file
void saveToFile(Vehicle* head, const char* filename) {
    FILE* file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error opening file for writing.\n");
        return;
    }

    Vehicle* current = head;
    while (current != NULL) {
        fprintf(file, "%s %s %s %d\n",
                                     current->licensePlate,  current->ownerName,  current->model,
current->year);
        current = current->next;
    }

    fclose(file);
    printf("Data saved successfully to %s.\n", filename);
}

// Load vehicle data from a file
Vehicle* loadFromFile(const char* filename) {
    FILE* file = fopen(filename, "r");
    if (file == NULL) {
        printf("Error opening file for reading.\n");
        return NULL;
    }

    Vehicle* head = NULL;
    char licensePlate[10], ownerName[50], model[30];
    int year;

    while (fscanf(file, "%s %s %s %d", licensePlate, ownerName, model, &year) == 4) {
        head = addVehicle(head, licensePlate, ownerName, model, year);
    }

    fclose(file);
    printf("Data loaded successfully from %s.\n", filename);
    return head;
}

// Schedule inspection (LIFO)
void scheduleInspectionLIFO(StackNode** top, char* licensePlate) {
    StackNode* newNode = (StackNode*)malloc(sizeof(StackNode));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }
    strcpy(newNode->licensePlate, licensePlate);
    newNode->next = *top;
    *top = newNode;
    printf("Scheduled %s for inspection (LIFO).\n", licensePlate);
}

// Process inspection (LIFO)
void processInspectionLIFO(StackNode** top) {
    if (*top == NULL) {
        printf("No inspections to process (LIFO).\n");
        return;
    }

    StackNode* temp = *top;
    printf("Processing inspection for %s (LIFO).\n", temp->licensePlate);
    *top = temp->next;
    free(temp);
}

// Schedule inspection (FIFO)
void scheduleInspectionFIFO(Queue* queue, char* licensePlate) {
    QueueNode* newNode = (QueueNode*)malloc(sizeof(QueueNode));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }
    strcpy(newNode->licensePlate, licensePlate);
    newNode->next = NULL;

    if (queue->rear == NULL) {
        queue->front = queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }

    printf("Scheduled %s for inspection (FIFO).\n", licensePlate);
}

// Process inspection (FIFO)
void processInspectionFIFO(Queue* queue) {
    if (queue->front == NULL) {
        printf("No inspections to process (FIFO).\n");
        return;
    }

    QueueNode* temp = queue->front;
    printf("Processing inspection for %s (FIFO).\n", temp->licensePlate);
    queue->front = temp->next;

    if (queue->front == NULL) {
        queue->rear = NULL;
    }

    free(temp);
}
```

File: main.c

```
/*
    main.c - Entry Point

    This file handles user interaction and calls appropriate functions
    to perform operations on the Vehicle Inspection Management System.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "inspection_system.h"

// Display the main menu
void displayMenu() {
    printf("\n--- Vehicle Inspection Management System ---\n");

    printf("1. Add Vehicle\n");

    printf("2. Display Vehicles\n");

    printf("3. Save Vehicles to File\n");

    printf("4. Load Vehicles from File\n");

    printf("5. Schedule Inspection (LIFO)\n");

    printf("6. Process Inspection (LIFO)\n");

    printf("7. Schedule Inspection (FIFO)\n");

    printf("8. Process Inspection (FIFO)\n");

    printf("9. Exit\n");

    printf("Enter your choice: ");

}


// Main function

int main() {

    Vehicle* vehicleList = NULL;  // Head of the vehicle linked list

    StackNode* lifoStack = NULL; // Top of the LIFO stack

    Queue fifoQueue = {NULL, NULL}; // FIFO queue

    int choice;

    char licensePlate[10], ownerName[50], model[30];

    int year;


    do {

        displayMenu();

        scanf("%d", &choice);

        getchar(); // Consume newline


        switch (choice) {

        case 1:

            printf("Enter license plate: ");

            scanf("%s", licensePlate);

            printf("Enter owner name: ");

            scanf("%s", ownerName);
            printf("Enter model: ");

            scanf("%s", model);

            printf("Enter year: ");

            scanf("%d", &year);

            vehicleList = addVehicle(vehicleList, licensePlate, ownerName, model, year);

            break;


        case 2:

            displayVehicles(vehicleList);

            break;


        case 3:

            saveToFile(vehicleList, "vehicles.txt");

            break;


        case 4:

            vehicleList = loadFromFile("vehicles.txt");

            break;


        case 5:

            printf("Enter license plate to schedule (LIFO): ");

            scanf("%s", licensePlate);

            scheduleInspectionLIFO(&lifoStack, licensePlate);

            break;


        case 6:

            processInspectionLIFO(&lifoStack);

            break;


        case 7:

            printf("Enter license plate to schedule (FIFO): ");

            scanf("%s", licensePlate);

            scheduleInspectionFIFO(&fifoQueue, licensePlate);
```

```
                break;

            case 8:
                processInspectionFIFO(&fifoQueue);
                break;

            case 9:
                printf("Exiting program...\n");
                break;

            default:
                printf("Invalid choice! Try again.\n");
            }
        } while (choice != 9);

        return 0;
    }
```