Poetry Diary

```
#include <stdio.h>
#include<stdlib.h>
#include <string.h>

#define MAX_POEMS 100
#define MAX_TITLE 100
#define MAX_CONTENT
1000 #define FILENAME
"poems.dat"

struct Poem {
    int id;
    char title[MAX_TITLE];
    char
    content[MAX_CONTEN
    T];
};

struct Poem
poems[MAX_POEMS]; int
poemCount = 0;

void displayMenu();
void addPoem();
void
viewAllPoems();
void searchByTitle();
void editPoem();
void deletePoem();
void saveToFile();
void loadFromFile();
void clearInputBuffer();

int main() {
    int choice;
    loadFromFile();

    printf("\n========================================\n");
    printf("      Welcome to Your Poetry Diary!\n");
    printf("==================================
    =======\n");

    while(1) {
        displayMenu();
        printf("Enter your choice: ");
        scanf("%d", &choice);
        clearInputBuffer();

        switch(choice) {
            case 1: addPoem(); break;
            case 2: viewAllPoems(); break;
            case 3: searchByTitle(); break;
            case 4: editPoem(); break;
            case 5: deletePoem(); break;
            case 6:
                saveToFile();
                printf("\nPoems saved successfully!
                Goodbye!\n"); exit(0);
            default:
                printf("\nInvalid choice! Please try again.\n");
```

```
            }
        }
        return 0;
}

void displayMenu() {
        printf("\n=========================================
        ==\n");
        printf("                    MAIN MENU\n");
        printf("=================================
        ========\n");
        printf("1. Add a new poem\n");
        printf("2. View all poems\n");
        printf("3. Search poem by title\n");
        printf("4. Edit a poem\n"); printf("5.
        Delete a poem\n"); printf("6. Save
        and Exit\n");
        printf("=========================================\n");
```

```
}
void addPoem() {
    if(poemCount >= MAX_POEMS) {
        printf("\nDiary is full! Cannot add more poems.\n");
        return;
    }

    struct Poem newPoem;
    newPoem.id = poemCount +
    1;
    printf("\n--- Add New Poem ---\n");
    printf("Enter poem title: ");
    fgets(newPoem.title, MAX_TITLE,
    stdin);
    newPoem.title[strcspn(newPoem.title, "\n")] = '\0';
    printf("Enter your poem (end with a line containing only 'END'):\n");
    char line[MAX_CONTENT];
    newPoem.content[0] = '\0';
    while(1) {
        fgets(line, MAX_CONTENT, stdin);
        if(strcmp(line, "END\n") == 0) break;
        if(strlen(newPoem.content) + strlen(line) < MAX_CONTENT - 1)
            strcat(newPoem.content, line);
        else { printf("\nPoem too long! Truncated.\n"); break; }
    }
    poems[poemCount++] = newPoem;
    printf("\n✓ Poem added successfully! (ID: %d)\n", newPoem.id);
}

void viewAllPoems() {
    if(poemCount == 0) {
        printf("\nNo poems in your diary yet!\n");
        return;
    }
    printf("\n==================================
    =======\n"); printf("         ALL POEMS IN YOUR
    DIARY\n");
    printf("==================================
    ======\n");
    for(int i = 0; i < poemCount; i++) {
        printf("\n[Poem ID: %d]\n", poems[i].id);
        printf("Title: %s\n", poems[i].title);
        printf("Content:\n%s\n", poems[i].content);
        printf(" ---------------------------------------------- \n");
    }
    printf("\nTotal poems: %d\n", poemCount);
}

void searchByTitle() {
    if(poemCount == 0) {
        printf("\nNo poems to search!\n");
        return;
    }
    char searchTitle[MAX_TITLE];
    printf("\nEnter title to search: ");
    fgets(searchTitle, MAX_TITLE,
    stdin);
    searchTitle[strcspn(searchTitle, "\n")] = '\0'; int
    found = 0;
    for(int i = 0; i < poemCount; i++) {
        if(strstr(poems[i].title, searchTitle) != NULL) {
```

```
            printf("\n[Found - Poem ID: %d]\n", poems[i].id);
            printf("Title: %s\n", poems[i].title);
            printf("Content:\n%s\n", poems[i].content);
            printf("----------------------------------------------\n");
            found = 1;
        }
    }
    if(!found) printf("\nNo poems found with title containing '%s'\n", searchTitle);
}

void editPoem() {
    if(poemCount == 0)
    {
        printf("\nNo poems to edit!\n");
        return;
    }
    int id;
    printf("\nEnter poem ID to edit: ");
    scanf("%d", &id);
    clearInputBuffer();
    int index = -1;
```

```
    for(int i = 0; i < poemCount; i++) if(poems[i].id == id) index = i;
    if(index == -1) { printf("\nPoem with ID %d not found!\n", id); return; }
    printf("\n--- Editing Poem: %s ---\n", poems[index].title);
    printf("1. Edit title\n2. Edit content\nEnter choice: "); int
    choice; scanf("%d", &choice); clearInputBuffer(); char
    line[MAX_CONTENT];
    switch(choice) {
        case 1:
            printf("Enter new title: ");
            fgets(poems[index].title, MAX_TITLE,
            stdin);
            poems[index].title[strcspn(poems[index].title, "\n")] = '\0';
            break;
        case 2:
            printf("Enter new content (end with 'END'):\n");
            poems[index].content[0] = '\0';
            while(1) {
                fgets(line, MAX_CONTENT, stdin);
                if(strcmp(line, "END\n") == 0) break;
                if(strlen(poems[index].content) + strlen(line) < MAX_CONTENT -
                    1) strcat(poems[index].content, line);
                else { printf("\nPoem too long! Truncated.\n"); break; }
            }
            break;
        default: printf("Invalid choice!\n"); return;
    }
    printf("\n✓ Poem updated successfully!\n");
}
void deletePoem() {
    if(poemCount == 0) { printf("\nNo poems to delete!\n"); return; }
    int id;
    printf("\nEnter poem ID to delete: ");
    scanf("%d", &id);
    clearInputBuffer();
    int index = -1;
    for(int i = 0; i < poemCount; i++) if(poems[i].id == id) index = i;
    if(index == -1) { printf("\nPoem with ID %d not found!\n", id); return; }
    printf("\nAre you sure you want to delete '%s'? (y/n): ", poems[index].title);
    char confirm; scanf("%c", &confirm); clearInputBuffer();
    if(confirm == 'y' || confirm == 'Y') {
        for(int i = index; i < poemCount - 1; i++) poems[i] = poems[i + 1];
        poemCount--; printf("\n✓ Poem deleted successfully!\n");
    } else printf("\nDeletion cancelled.\n");
}
void saveToFile() {
    FILE *file = fopen(FILENAME, "wb");
    if(file == NULL) { printf("\nError saving file!\n"); return; }
    fwrite(&poemCount, sizeof(int), 1, file);
    fwrite(poems, sizeof(struct Poem), poemCount, file);
    fclose(file);
    printf("\n✓ All poems saved to file!\n");
}
void loadFromFile() {
    FILE *file = fopen(FILENAME, "rb");
    if(file == NULL) return;
    fread(&poemCount, sizeof(int), 1, file);
    fread(poems, sizeof(struct Poem), poemCount, file);
    fclose(file);
    printf("\n✓ Loaded %d poem(s) from file.\n", poemCount);
```

```
}
void clearInputBuffer() {
	int c; while ((c = getchar()) != '\n' && c != EOF);
}
```