#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_TASKS 10

#define MAX\_DESCRIPTION\_LENGTH 50

#define MAX\_COURSES 5

struct Task

{

char description[MAX\_DESCRIPTION\_LENGTH];

int completed;

};

struct Task tasks[MAX\_TASKS];

int taskCount = 0;

struct Course {

char name[20];

int credits;

char grade;

};

struct Course courses[MAX\_COURSES];

int courseCount = 0;

void addTask() {

if (taskCount < MAX\_TASKS) {

printf("Enter task description: ");

scanf(" %[^\n]", tasks[taskCount].description);

tasks[taskCount].completed = 0;

taskCount++;

printf("Task added successfully!\n");

} else {

printf("Task limit reached. Cannot add more tasks.\n");

}

}

void markComplete() {

int taskIndex;

printf("Enter task number to mark as complete: ");

scanf("%d", &taskIndex);

if (taskIndex >= 1 && taskIndex <= taskCount) {

tasks[taskIndex - 1].completed = 1;

printf("Task marked as complete!\n");

} else {

printf("Invalid task number.\n");

}

}

void editTask() {

int taskIndex;

printf("Enter task number to edit: ");

scanf("%d", &taskIndex);

if (taskIndex >= 1 && taskIndex <= taskCount) {

printf("Enter new task description: ");

scanf(" %[^\n]", tasks[taskIndex - 1].description);

printf("Task edited successfully!\n");

} else {

printf("Invalid task number.\n");

}

}

void sortTasks() {

// Sort tasks alphabetically based on descriptions

for (int i = 0; i < taskCount - 1; i++) {

for (int j = i + 1; j < taskCount; j++) {

if (strcmp(tasks[i].description, tasks[j].description) > 0) {

// Swap tasks

struct Task temp = tasks[i];

tasks[i] = tasks[j];

tasks[j] = temp;

}

}

}

printf("Tasks sorted alphabetically!\n");

}

void saveTasksToFile() {

FILE \*file = fopen("tasks.txt", "w");

if (file != NULL) {

for (int i = 0; i < taskCount; i++) {

fprintf(file, "%s %d\n", tasks[i].description, tasks[i].completed);

}

fclose(file);

printf("Tasks saved to file!\n");

} else {

printf("Error saving tasks to file.\n");

}

}

void loadTasksFromFile() {

FILE \*file = fopen("tasks.txt", "r");

if (file != NULL) {

taskCount = 0;

while (fscanf(file, "%s %d", tasks[taskCount].description, &tasks[taskCount].completed) == 2 && taskCount < MAX\_TASKS) {

taskCount++;

}

fclose(file);

printf("Tasks loaded from file!\n");

} else {

printf("No saved tasks found.\n");

}

}

void addCourse() {

if (courseCount < MAX\_COURSES) {

printf("Enter course name: ");

scanf(" %[^\n]", courses[courseCount].name);

printf("Enter course credits: ");

scanf("%d", &courses[courseCount].credits);

printf("Enter grade (A, B, C, D, F): ");

scanf(" %c", &courses[courseCount].grade);

courseCount++;

printf("Course added successfully!\n");

} else {

printf("Course limit reached. Cannot add more courses.\n");

}

}

float calculateCGPA() {

float totalCredits = 0.0;

float weightedSum = 0.0;

for (int i = 0; i < courseCount; i++) {

totalCredits += courses[i].credits;

switch (courses[i].grade) {

case 'A':

weightedSum += 4.0 \* courses[i].credits;

break;

case 'B':

weightedSum += 3.0 \* courses[i].credits;

break;

case 'C':

weightedSum += 2.0 \* courses[i].credits;

break;

case 'D':

weightedSum += 1.0 \* courses[i].credits;

break;

case 'F':

// No points for a failing grade

break;

default:

printf("Invalid grade entered for course %s. Skipping calculation for this course.\n", courses[i].name);

}

}

if (totalCredits == 0.0) {

return 0.0; // Avoid division by zero

}

return weightedSum / totalCredits;

}

// ... (Continuation from the first part)

void viewTasks() {

printf("\nTask List:\n");

for (int i = 0; i < taskCount; i++) {

printf("%d. %s - %s\n", i + 1, tasks[i].description, tasks[i].completed ? "Completed" : "Incomplete");

}

printf("\n");

}

void deleteTask() {

int taskIndex;

printf("Enter task number to delete: ");

scanf("%d", &taskIndex);

if (taskIndex >= 1 && taskIndex <= taskCount) {

for (int i = taskIndex - 1; i < taskCount - 1; i++) {

tasks[i] = tasks[i + 1];

}

taskCount--;

printf("Task deleted successfully!\n");

}

else

{

printf("Invalid task number.\n");

}

}

void clearCompletedTasks()

{

for (int i = 0; i < taskCount; )

{

if (tasks[i].completed) {

for (int j = i; j < taskCount - 1; j++) {

tasks[j] = tasks[j + 1];

}

taskCount--;

} else {

i++;

}

}

printf("Completed tasks cleared!\n");

}

int main()

{

int choice;

do

{

printf("Task Manager & CGPA Calculator Menu:\n");

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

printf("2. Mark Task as Complete\n");

printf("3. Edit Task\n");

printf("4. Sort Tasks\n");

printf("5. Save Tasks to File\n");

printf("6. Load Tasks from File\n");

printf("7. View Tasks\n");

printf("8. Delete Task\n");

printf("9. Clear Completed Tasks\n");

printf("10. Add Course for CGPA Calculation\n");

printf("11. Calculate CGPA\n");

printf("12. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

addTask();

break;

case 2:

markComplete();

break;

case 3:

editTask();

break;

case 4:

sortTasks();

break;

case 5:

saveTasksToFile();

break;

case 6:

loadTasksFromFile();

break;

case 7:

viewTasks();

break;

case 8:

deleteTask();

break;

case 9:

clearCompletedTasks();

break;

case 10:

addCourse();

break;

case 11:

printf("CGPA: %.2f\n", calculateCGPA());

break;

case 12:

printf("Exiting Task Manager & CGPA Calculator. Goodbye!\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

}

while (choice != 12);

return 0;

}