Project: Struct-Based School Management System (SMS)

Project Introduction

Welcome ladies and gentlemen! In this project we would like to develop a school management system. This system is a simplified application aimed to provide a comprehensive method of managing school-related information. It is designed to provide a streamlined, automated system for administrators, teachers, and students. Our focus is on creating a struct-based system to handle different kinds of data related tasks to a school, such as courses, student records, etc.

Main Components

The project can be divided into three main components:

1. **Student Struct**: Each student in the school has their own unique record, containing details like name and ID (integer).

2. **Course Struct**: Each course offered in the school will have its own struct. This includes the course name, average grade, and a dynamic array of enrolled students.

3. **School Struct**: This struct encapsulates all the courses offered in the school, making up the whole school system. This struct should include a school name and a dynamic array of courses offered by the school.

Key Functions to Implement

1. **Create a Student**: Takes user input for student details, allocates a new student struct in memory, and populates the struct with the provided details.

2. **Create a Course**: Asks for the number of students in a course and their details. Allocates a new course struct in memory, and populates it with the provided course details and students.

3. **Create a School**: Takes user input for the number of courses and their details. Allocates a new school struct in memory, and populates it with the provided details.

4. **Print Student Details**: Takes a student struct and prints its details.

5. **Print Course Details**: Accepts a course struct and prints the details of the course and the students enrolled in it.

6. **Print School Details**: Takes a school struct and prints the school's details, all the courses offered, and the students in those courses.

7. **Check if student in course**: Takes a student's ID and a course struct, iterates over the course's student list, and checks if the student is enrolled.

8. **Check if student in school**: Accepts a student's ID and a school struct. It uses the previous function to check if the student is enrolled in any of the school's courses.

9. **Print all student courses**: Receives a student's ID and a school struct. It prints out all the courses that the student is enrolled in.

10. **Print all students who failed a course**: Accepts a course struct and a passing grade. Prints the details of students who didn't meet the passing grade.

11. **Print all students who passed a course**: Similar to the previous function but prints details of students who have met or exceeded the passing grade.

12. **Print all the courses with a passed average grade**: Receives a school struct and a passing grade. Prints out courses where the average grade is equal to or higher than the passing grade.

13. **Print all the courses with a failed average grade**: Similar to the previous function but prints out courses where the average grade is below the passing grade.

14. **Print the average grade between all the courses**: Accepts a school struct, calculates the average grade for each course, then calculates and prints the overall average of these averages.

15. **Print the course with the highest average grade**: Receives a school struct, finds the course with the highest average grade, and prints its details.

Ensure Proper Memory Management

Almost everything should be done here dynamically..! Make sure to free the dynamically allocated memory when you're done using it.

Summary

Your main program (main function) should create a school, populate it with courses and students, and then print the details of the entire school.

The purpose of this exercise is to help you understand and become comfortable with structs, pointers, and dynamic memory allocation. Remember to focus on writing clean, understandable code. Good luck!