Cần tính GPA trung bình lớp. Nhiệm vụ:

1. Sử dụng Cursor AI *viết hàm tính GPA trung bình.*
2. *Nhờ AI sinh 10 sinh viên giả lập với GPA khác nhau để kiểm thử.*
3. *Prompt AI viết test case: trường hợp danh sách rỗng.*

Output:

```
GPA trung bình của lớp là: 8.25
```

Prompt :

"Hãy đóng vai một chuyên gia lập trình C để viết một chương trình quản lý sinh viên hoàn chỉnh với các yêu cầu sau:

1. Cấu trúc dữ liệu:

- Định nghĩa `struct Student` gồm: `id` (int), `name` (chuỗi), `age` (int), và `gpa` (float).

2. Khởi tạo và Dữ liệu giả lập (Simulation):

- Khởi tạo sẵn một mảng chứa 3 sinh viên mẫu.
- Yêu cầu AI: Hãy tạo thêm dữ liệu giả lập cho 10 sinh viên khác với các giá trị GPA khác nhau để mở rộng bộ dữ liệu kiểm thử.
- In toàn bộ danh sách sinh viên ban đầu ra màn hình dưới dạng bảng có tiêu đề.

3. Chức năng Nhập liệu:

- Viết hàm cho phép người dùng nhập thêm tối đa 5 sinh viên từ bàn phím. Sau khi nhập, in lại danh sách tổng hợp.

4. Xử lý File:

- Hàm `saveToFile`: Lưu toàn bộ danh sách sinh viên hiện có vào file `students.txt`.
- Hàm `readFromFile`: Đọc dữ liệu từ `students.txt` vào mảng `Student` và in ra màn hình.
- Định dạng Output: In đúng mẫu `[id] [name] [age] [gpa]` với GPA lấy 1 chữ số thập phân. Tiêu đề: `Danh sách đọc từ file:`.

5. Hàm Tính toán & Tìm kiếm (Có Docstring):

- Tìm kiếm: Viết hàm `findStudentByID`. Thêm Docstring giải thích: chức năng, tham số, giá trị trả về.
- Tính GPA: Viết hàm `calculateAverageGPA` để tính điểm trung bình của toàn bộ danh sách.

6. Kiểm thử (Unit Test):

- Test Case cho Tìm kiếm: Viết đoạn code kiểm thử tự động cho 3 trường hợp: ID tồn tại, ID không tồn tại, và ID âm.
- Test Case danh sách rỗng: Hãy viết một hàm kiểm thử hoặc đoạn logic xử lý đặc biệt cho trường hợp danh sách sinh viên rỗng (ví dụ: khi file `students.txt` không có dữ liệu hoặc mảng chưa có người nào) để chương trình không bị crash khi tính GPA hoặc tìm kiếm.

7. Yêu cầu kỹ thuật:

- Sử dụng hằng số `#define` cho `MAX_STUDENTS` và `FILENAME`.
- Sử dụng con trỏ khi cần thiết để tối ưu bộ nhớ.
- Có thông báo lỗi rõ ràng nếu không mở được file."

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

#define MAX_STUDENTS 18
#define FILENAME "students.txt"

struct Student {
    int id;
    char name[50];
    int age;
    float gpa;
};

// In bảng tiêu đề
void printHeader() {
    printf("\n%-10s %-30s %-10s %-10s\n", "ID", "Ten", "Tuoi", "GPA");
    printf("-------------------------------------------------------------------\n");
}

// In thông tin 1 sinh viên
void printStudent(struct Student s) {
    printf("%-10d %-30s %-10d %-10.1f\n", s.id, s.name, s.age, s.gpa);
}

// In danh sách sinh viên dạng bảng
void printStudentList(struct Student students[], int count) {
    if (count <= 0) {
        printf("Danh sach sinh vien rong!\n");
        return;
    }
    printHeader();
    for (int i = 0; i < count; i++) {
        printStudent(students[i]);
    }
    printf("\n");
}
```

```c
// Lưu danh sách sinh viên vào file
void saveToFile(struct Student students[], int count, const char* filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        printf("Khong the mo file de ghi!\n");
        return;
    }
    for (int i = 0; i < count; i++) {
        fprintf(file, "%d %s %d %.1f\n",
                students[i].id,
                students[i].name,
                students[i].age,
                students[i].gpa);
    }
    fclose(file);
    printf("Da luu %d sinh vien vao file %s\n", count, filename);
}

// Đọc dữ liệu sinh viên từ file vào mảng. Trả về số lượng sinh viên đã đọc.
int readFromFile(struct Student students[], int maxCount, const char* filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("Khong the mo file de doc!\n");
        return 0;
    }
    int count = 0;
    // Đọc lần lượt từng dòng
    while (count < maxCount &&
        fscanf(file, "%d %49s %d %f",
               &students[count].id, students[count].name,
               &students[count].age, &students[count].gpa) == 4)
    {
        count++;
    }
    fclose(file);
    return count;
}
```

```c
// Hàm in danh sách sinh viên dạng thô từ file
void printListFromFile(struct Student students[], int count) {
    printf("Danh sach doc tu file:\n");
    if (count <= 0) {
        printf("Khong co du lieu.\n");
        return;
    }
    for (int i = 0; i < count; i++) {
        printf("%d %s %d %.1f\n", students[i].id, students[i].name, students[i].age, students[i].gpa);
    }
}

// Nhập thêm sinh viên từ bàn phím
int inputStudents(struct Student students[], int currentCount, int maxCount) {
    int nAdd = 0;
    printf("\nNhap so luong sinh vien muon them (toi da %d): ", maxCount - currentCount);
    scanf("%d", &nAdd);
    if (nAdd < 1) {
        printf("Khong them sinh vien moi.\n");
        return 0;
    }
    if (nAdd > maxCount - currentCount) nAdd = maxCount - currentCount;
    for (int i = 0; i < nAdd; i++) {
        printf("\nNhap thong tin sinh vien thu %d:\n", currentCount + i + 1);
        printf("ID: ");
        scanf("%d", &students[currentCount + i].id);
        printf("Ten: ");
        scanf("%49s", students[currentCount + i].name);
        printf("Tuoi: ");
        scanf("%d", &students[currentCount + i].age);
        printf("GPA: ");
        scanf("%f", &students[currentCount + i].gpa);
    }
    return nAdd;
}

/**
 * Tim kiem sinh vien theo ID trong mang.
 * @param students Mang cac sinh vien
 * @param count So luong sinh vien trong mang
 * @param id ID cua sinh vien can tim
 * @return Con tro den sinh vien neu tim thay, NULL neu khong tim thay hoac id khong hop le
 */
```

```c
struct Student* findStudentByID(struct Student students[], int count, int id) {
    if (id < 0) return NULL;
    if (count <= 0) return NULL;
    for (int i = 0; i < count; i++) {
        if (students[i].id == id) {
            return &students[i];
        }
    }
    return NULL;
}

/**
 * Tinh GPA trung binh cua danh sach sinh vien.
 * @param students Mang cac sinh vien
 * @param count So luong sinh vien
 * @return Tra ve GPA trung binh (float), neu danh sach rong tra ve -1
 */
float calculateAverageGPA(struct Student students[], int count) {
    if (count <= 0) return -1.0f;
    float sum = 0.0f;
    for (int i = 0; i < count; i++)
        sum += students[i].gpa;
    return sum / count;
}
```

```c
// --- MAIN PROGRAM ---

int main() {
    // Khởi tạo 3 sinh viên mẫu
    struct Student students[MAX_STUDENTS] = {
        {1, "An", 20, 8.0},
        {2, "Binh", 21, 7.5},
        {3, "Cuong", 19, 8.5},
        // 10 sinh viên giả lập mở rộng
        {4, "Dung", 20, 6.7},
        {5, "Hoa", 22, 7.3},
        {6, "Khanh", 18, 5.9},
        {7, "Lam", 21, 9.1},
        {8, "Mai", 20, 8.3},
        {9, "Nhan", 23, 5.8},
        {10, "Oanh", 19, 7.0},
        {11, "Phuc", 21, 9.6},
        {12, "Quang", 20, 6.0},
        {13, "Son", 20, 8.9}
    };
    int studentCount = 13; // Đã có 13 sinh viên ban đầu

    printf("\n=== DANH SACH SINH VIEN BAN DAU ===\n");
    printStudentList(students, studentCount);

    // Nhập thêm sinh viên từ bàn phím
    int nAdded = inputStudents(students, studentCount, MAX_STUDENTS);
    studentCount += nAdded;

    printf("\n=== DANH SACH SINH VIEN SAU KHI NHAP THEM ===\n");
    printStudentList(students, studentCount);

    // Lưu vào file
    saveToFile(students, studentCount, FILENAME);

    // Đọc lại từ file và in
    struct Student fromFile[MAX_STUDENTS];
    int nFromFile = readFromFile(fromFile, MAX_STUDENTS, FILENAME);
    printf("\n");
    printListFromFile(fromFile, nFromFile);

    // --- Test Case cho tìm kiếm sinh viên ---
    printf("\n=== TEST TIM KIEM SINH VIEN THEO ID ===\n");
```

```c
// 1. ID tồn tại
int test_id = students[2].id; // ID = 3 (chắc chắn tồn tại)
struct Student* found = findStudentByID(students, studentCount, test_id);
printf("\nTest case 1: Tim ID = %d (ton tai)\n", test_id);
if (found) {
    printf("Tim thay sinh vien: ID=%d, Ten=%s, Tuoi=%d, GPA=%.1f\n",
           found->id, found->name, found->age, found->gpa);
} else {
    printf("Khong tim thay sinh vien co ID = %d\n", test_id);
}

// 2. ID không tồn tại
int not_exist_id = 1000;
found = findStudentByID(students, studentCount, not_exist_id);
printf("\nTest case 2: Tim ID = %d (khong ton tai)\n", not_exist_id);
if (found) {
    printf("Tim thay sinh vien: ID=%d, Ten=%s, Tuoi=%d, GPA=%.1f\n",
           found->id, found->name, found->age, found->gpa);
} else {
    printf("Khong tim thay sinh vien co ID = %d\n", not_exist_id);
}

// 3. ID âm
int neg_id = -5;
found = findStudentByID(students, studentCount, neg_id);
printf("\nTest case 3: Tim ID = %d (so am)\n", neg_id);
if (found) {
    printf("Tim thay sinh vien: ID=%d, Ten=%s, Tuoi=%d, GPA=%.1f\n",
           found->id, found->name, found->age, found->gpa);
} else {
    printf("ID = %d khong hop le (so am hoac khong ton tai)\n", neg_id);
}
```

```c
    // --- Tính GPA trung bình ---
    printf("\n=== TINH DIEM TRUNG BINH GPA ===\n");
    float avgGPA = calculateAverageGPA(students, studentCount);
    if (avgGPA < 0)
        printf("Khong the tinh GPA trung binh (danh sach rong).\n");
    else
        printf("GPA trung binh cua %d sinh vien: %.2f\n", studentCount, avgGPA);

    // Test tình huống danh sách rỗng/tệp tin trống:
    printf("\n=== TEST DANH SACH SINH VIEN RONG ===\n");
    struct Student emptyArr[5];
    int emptyCount = 0;
    float avgEmpty = calculateAverageGPA(emptyArr, emptyCount);
    if (avgEmpty < 0)
        printf("-> Danh sach rong: Khong the tinh GPA trung binh.\n");
    else
        printf("GPA danh sach rong: %.2f\n", avgEmpty);

    struct Student* testFind = findStudentByID(emptyArr, emptyCount, 1);
    if (testFind == NULL)
        printf("-> Tim kiem tren mang rong: Khong tim thay sinh vien.\n");
    else
        printf("Sinh vien bat ky: ID=%d\n", testFind->id);

    // Test file trống
    FILE *ftest = fopen("empty.txt", "w");
    if (ftest) fclose(ftest);
    int nEmptyFile = readFromFile(emptyArr, 5, "empty.txt");
    printf("Doc tu file empty.txt, so luong: %d\n", nEmptyFile);
    printListFromFile(emptyArr, nEmptyFile);

    return 0;
}
```