Final Project 01: Student Management

Objectives:

Design and implement a comprehensive C application to manage student records, applying key concepts learned throughout the course including:

- Struct.
- Array (Struct Array)
- File processing.

Group Work Guidelines:

- Group size: 3–5 students.
- Duration: 4–6 weeks.
- Deliverables:
  - Source code.
  - Technical report (REPORT.pdf).
  - Presentation slides (SLIDES.pptx).
- Submission via LMS (as directed by the instructor).

Functional Requirements:

1. **Student Management:**
   - Add / edit / delete student records.
   - Student information example:

```
Struct Student {
  char[15] id;
  char[50] name;
  int birthYear;
  char[30] major;
  float gpa;
};
```

2. **Student scores:**
   - Allows entering scores for each student.
   - Uses struct array to manage scores.

- Calculate GPA per student and overall GPA.

3. **Search and Sorting:**
   - Search by student name (part of the name, case insensitive) or id.
   - Sort students by GPA, name, or birth year.
4. **Console-Based User Interface:**
   - Design a simple and intuitive text-based menu system.
5. **File I/O:**
   - Load and save data from/to text or binary files.
6. **Functional Programming:**
   - Divide the program into .c / .h files, these files contain functions to handle the functions: add, delete, edit, sort and search.
7. **Advanced Features:**
   - Optimize add, delete, edit, sort and search operations.

Technical Requirements:

- Organize code into multiple .c and .h files (Modular Programming).
- Prepare a presentation explaining architecture and implementation choices.
- Follow best practices for memory safety and optimization.

-- End --.

Final Project 02: Book Management

Objectives:

Design and implement a comprehensive C application to manage book records, applying key concepts learned throughout the course including:

- Struct.
- Array (Struct Array)
- File processing.

Group Work Guidelines:

- Group size: 3–5 students.
- Duration: 4–6 weeks.
- Deliverables:
  - Source code.
  - Technical report (REPORT.pdf).
  - Presentation slides (SLIDES.pptx).
- Submission via LMS (as directed by the instructor).

Functional Requirements:

1. **Book Management:**
   - Add / edit / delete book records.
   - Book information example:

```
Struct Book {
  char[15] id;
  char[50] title;
  int numberOfPage;
  char[30] author;
  int publishYear;
};
```

2. **Book statistics by author:**
   - Allows entering title for each book.
   - Uses struct array to manage books.
   - Statistics number of books per author.

3. **Search and Sorting:**
   - Search by title (part of the title, case insensitive) or id.
   - Sort books by numberOfPage, title, or publishYear.
4. **Console-Based User Interface:**
   - Design a simple and intuitive text-based menu system.
5. **File I/O:**
   - Load and save data from/to text or binary files.
6. **Functional Programming:**
   - Divide the program into .c / .h files, these files contain functions to handle the functions: add, delete, edit, sort and search.
7. **Advanced Features:**
   - Optimize add, delete, edit, sort and search operations.

Technical Requirements:

- Organize code into multiple .c and .h files (Modular Programming).
- Prepare a presentation explaining architecture and implementation choices.
- Follow best practices for memory safety and optimization.

-- End --.

Final Project 03: Product Management

Objectives:

Design and implement a comprehensive C application to manage product records, applying key concepts learned throughout the course including:

- Struct.
- Array (Struct Array)
- File processing.

Group Work Guidelines:

- Group size: 3–5 students.
- Duration: 4–6 weeks.
- Deliverables:
  - Source code.
  - Technical report (REPORT.pdf).
  - Presentation slides (SLIDES.pptx).
- Submission via LMS (as directed by the instructor).

Functional Requirements:

1. **Product Management:**
   - Add / edit / delete product records.
   - Product information example:

```
Struct Product {
  char[15] id;
  char[50] name;
  int quantity;
  char[30] stockName;
  float unitPrice;
};
```

2. **Product statistics by stock name:**
   - Allows entering title for each product.
   - Uses struct array to manage products.
   - Statistics number of products per stock name.

3. **Search and Sorting:**
   - Search by product name (part of the name, case insensitive) or id.
   - Sort product by name, quantity, or unit Price.
4. **Console-Based User Interface:**
   - Design a simple and intuitive text-based menu system.
5. **File I/O:**
   - Load and save data from/to text or binary files.
6. **Functional Programming:**
   - Divide the program into .c / .h files, these files contain functions to handle the functions: add, delete, edit, sort and search.
7. **Advanced Features:**
   - Optimize add, delete, edit, sort and search operations.

Technical Requirements:

- Organize code into multiple .c and .h files (Modular Programming).
- Prepare a presentation explaining architecture and implementation choices.
- Follow best practices for memory safety and optimization.

-- End --.