Here is a detailed documentation for the provided C program of student management system:

Program Documentation: Student Management System

Header Files

The program includes the following header files:

1. **<stdio.h>** - Provides input/output functionality.
2. **<stdlib.h>** - Used for general utility functions such as exit().
3. **<string.h>** - Used for string manipulation functions such as strcmp() and strcpy().

Preprocessor Directives

- **#define MAX_STUDENTS 100**: Limits the maximum number of students to 100.
- **#define MAX_SUBJECTS 6**: Specifies the number of subjects for grade management.

Structures

1. **struct Student**:
    - **Attributes**: id, name, age, gender, marks, percentage.
    - Stores detailed student information.
2. **struct User**:
    - **Attributes**: username, password.
    - Used for handling user login credentials.

Global Variables

- **struct Student students[MAX_STUDENTS]**: Array to store student data.
- **int student_count**: Tracks the number of students in the system.

Functions

User Authentication

1. **void user_login()**
    - Verifies admin login credentials (admin/adminpass).
    - Terminates the program if login fails.

Main Menu

2. **void menu()**

- Displays available actions to the user.
- Handles user input and calls corresponding functions using a switch statement.

Core Functionalities

3. **void add_student()**
    - Adds a new student to the system by accepting details like name, age, and gender.
    - Validates gender input.
4. **void display_students()**
    - Lists all students with their id, name, age, and gender.
5. **void update_student()**
    - Updates an existing student's details (name, age, gender).
6. **void delete_student()**
    - Deletes a student by their id and shifts the array to maintain order.
7. **void search_student()**
    - Searches for a student by name and displays their detailed report if found.

Academic Management

8. **void input_grades(struct Student* student)**
    - Prompts the user to input grades for six subjects. Validates input to ensure marks are between 0 and 100.
9. **void calculate_percentage(struct Student* student)**
    - Calculates the percentage based on the total marks of six subjects.
10. **void check_pass_status(struct Student* student)**
    - Checks if the student has passed (percentage ≥ 50%).
11. **void detailed_report(struct Student* student)**
    - Displays a detailed report of a student, including subject-wise marks and percentage.

Data Handling

12. **void save_data()**
    - Saves all student records to a file (students_data.txt).
13. **void load_data()**
    - Loads student records from a file into the program.

14. **void backup_data()**
    - Creates a backup of the student data file.
15. **void print_reports()**
    - Prints detailed reports for all students.

How the Program Works

1. The program starts with **admin login** using user_login().
2. The **menu** displays options for managing student records.
3. Based on the user's choice, the respective function is called to:
    - Add, update, delete, or display student information.
    - Input grades, calculate percentages, check pass status, or generate reports.
    - Save, load, or back up data.

Key User-Defined Functions

Function	**Purpose**
add_student()	Adds a new student to the system.
display_students()	Displays a list of all students.
update_student()	Updates details of a specific student.
delete_student()	Removes a student record from the system.
input_grades()	Inputs grades for a student after validating marks.
calculate_percentage()	Calculates the percentage of marks for a student.
check_pass_status()	Determines whether a student has passed based on their percentage.
detailed_report()	Generates a detailed report of a student, including marks and percentage.
save_data()	Saves student records to a file for persistent storage.
load_data()	Loads student records from a file into the system.
backup_data()	Creates a backup of the student data file for safekeeping.

Function	Purpose
print_reports()	Prints reports for all students in the system.
search_student()	Searches and displays information for a specific student by name.

Contributors

Some functions are labeled with **Zeyam Hussain** as their contributor, indicating their development of specific functionalities like input_grades, calculate_percentage, detailed_report, check_pass_status, and backup_data.

This documentation should serve as a comprehensive guide for understanding and maintaining the program.