C PROGRAMMING PROJECT

COURSE CODE: CSEG1032

PROJECT TITLE: *Student Management System*

NAME: Prakhar Shukla

BATCH-19

SAP ID- 590022600

ABSTRACT-

The project is a simple in-memory Student Information Management System developed in C. It serves as a console-based application to manage student academic records, specifically marks and attendance, and provides two distinct user interfaces: a Student Portal for viewing personal records and a Teacher Portal for managing student enrollment, updating marks, and recording attendance. The system utilizes basic C concepts like structures, arrays, functions, and string manipulation, along with ANSI color codes for enhanced user experience in the terminal.

Problem Definition-

The core problem this project addresses is the need for a simplified, non-persistent digital system to manage and track the academic data of students within a limited scope.

Manual Data Handling: Traditional record-keeping can be cumbersome and prone to error.

Access Control: Students require a secure way to view *only* their own data, while teachers need administrative access to view and modify *all* relevant student data.

Data Structure: A clear, consistent structure is needed to hold student details (Name, SAP ID, Password, Marks, Attendance for multiple subjects).

System Constraints: The system needs to operate entirely in a C console environment without external file storage or a database, thus using global arrays for data storage.

The project solves this by creating a menu-driven, role-based access system for viewing and modifying structured student records.

Flow Chart Algorithm-

The overall program flow is driven by a main menu (home_menu).

Main Program Flow

Start: main() function execution begins.

Initial Setup: create_initial_data() is called.

Input: User specifies the number of initial teachers and students.

Creation: Loops prompt for and store initial Teacher (Username, Password) and Student (SAP ID, Name, Password) data into global arrays.

Initialize: Student academic data (Marks, Attendance) are set to 0.

Home Menu Loop (home_menu)

Display: Show options: Student Login, Teacher Login, Create New Student ID, Create New Teacher ID, Exit.

Choice: Read user choice.

Case 1 (Student Login): Call student_login().

If login succeeds, enter student_portal().

Case 2 (Teacher Login): Call teacher_login().

If login succeeds, enter teacher_portal().

Case 3/4 (Create ID): Call respective creation functions.

Case 0 (Exit): Terminate program.

Repeat until 'Exit' is chosen.

Teacher Portal Flow (Simplified)

Teacher Login $\rightarrow$ Success $\rightarrow$ Enter teacher_portal().

Teacher Menu: Display options (Manage Enrollment, Edit Data, Logout).

Case 1 (Manage Enrollment): Enter teacher_manage_students().

Sub-Menu: Add Student, Remove Student, View All, Back.

Add Student: If capacity allows, prompt for new SAP ID (must be unique and 9 digits), Name, and Password. Add to array.

Remove Student: Prompt for SAP ID. Find index. Shift array elements to remove. Decrease count.

Case 2 (Edit Data): Enter teacher_edit_student_data().

Input: Prompt for Student SAP ID.

Search: Find the student's index.

Edit Loop: Allow teacher to choose to update Marks (Maths/Physics/Coding) or Attendance, or View Data. Repeat until 'Finish Editing'.

Implementation-

Data Storage

Data is stored in two global arrays of C structures:

Student students[MAX_STUDENTS]: Holds records for up to 100 students. Each record includes SAP ID, Password, Name, 3 Marks fields, and 3 Attendance fields.

Teacher teachers[MAX_TEACHERS]: Holds records for up to 10 teachers, each with a Username and Password.

Key Functions

find_student_index(const char* sap_id): A crucial utility function that linearly searches the students array to find the index of a student given their SAP ID. Returns the index or -1 if not found.

clear_input_buffer(): Ensures proper input handling by consuming any leftover characters (like the newline $\backslash n$) in the input stream, preventing unexpected behavior in subsequent scanf or fgets calls.

student_login() / teacher_login(): Implement the core authentication logic by comparing user-provided credentials (SAP ID/Username and Password) against the data stored in the global arrays using strcmp.

teacher_manage_students(): Handles array manipulation for Adding and Removing student records. The removal uses a standard array shift to maintain contiguity: students[i] = students[i+1];.

teacher_edit_student_data(): Demonstrates the use of a pointer to an integer (int* mark_ptr) to dynamically select and update one of the student's six academic fields (marks or attendance for a specific subject) based on the teacher's menu choice.

Features

Role-Based Access: Separate logins and menus for students and teachers.

Student View: Students can only view their own pre-formatted academic dashboard (student_portal).

Teacher Management: Teachers can add new students, remove existing students, and update their marks and attendance for three subjects (Maths, Physics, Coding).

Input Validation: Checks for SAP ID length, range constraints (0-100) for marks/attendance, and maximum capacity checks (MAX_STUDENTS, MAX_TEACHERS).

Basic Theming: Use of ANSI Escape Codes (C_RED, C_BLUE, etc.) for colored and bold text output in the terminal.

C Programming Concepts Used-

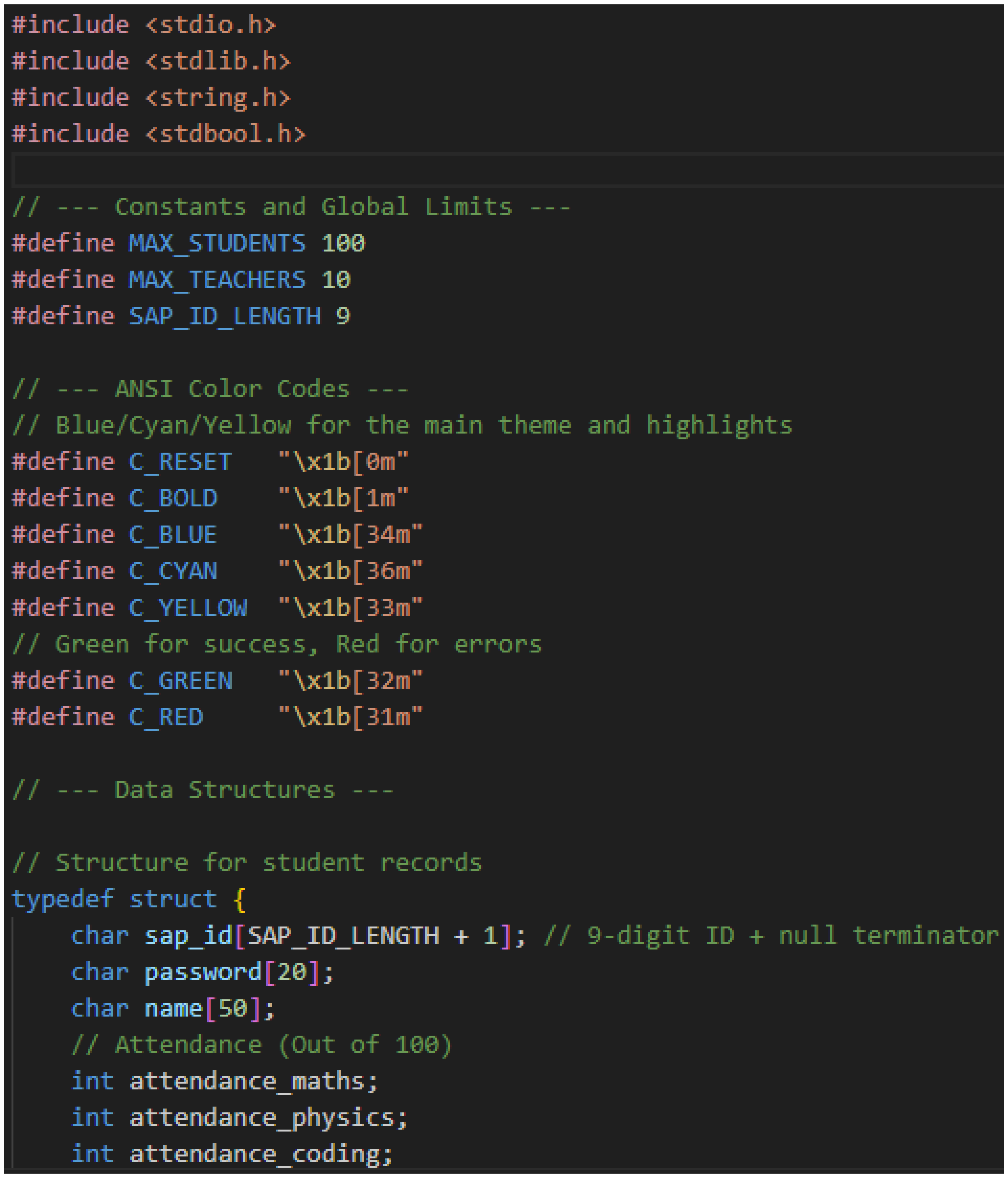
This project utilizes a wide range of fundamental C programming concepts:

C Concept	Usage in Project
Preprocessor Directives	#include <stdio.h>, #define MAX_STUDENTS, and #define C_BLUE for constants and libraries.
main() Function	The program's entry point, which calls the setup and main menu loops.
Functions (Voids)	Extensive use of custom functions (home_menu, clear_input_buffer, teacher_portal, etc.) to organize the code and implement modularity.
Data Types	char, int, and bool (from stdbool.h).
Arrays	Used for global data storage (students[MAX_STUDENTS], teachers[MAX_TEACHERS]) and for character arrays (strings) like sap_id, name, and password.
Structures (struct)	Used to create custom, composite data types: Student and Teacher, grouping related data fields.

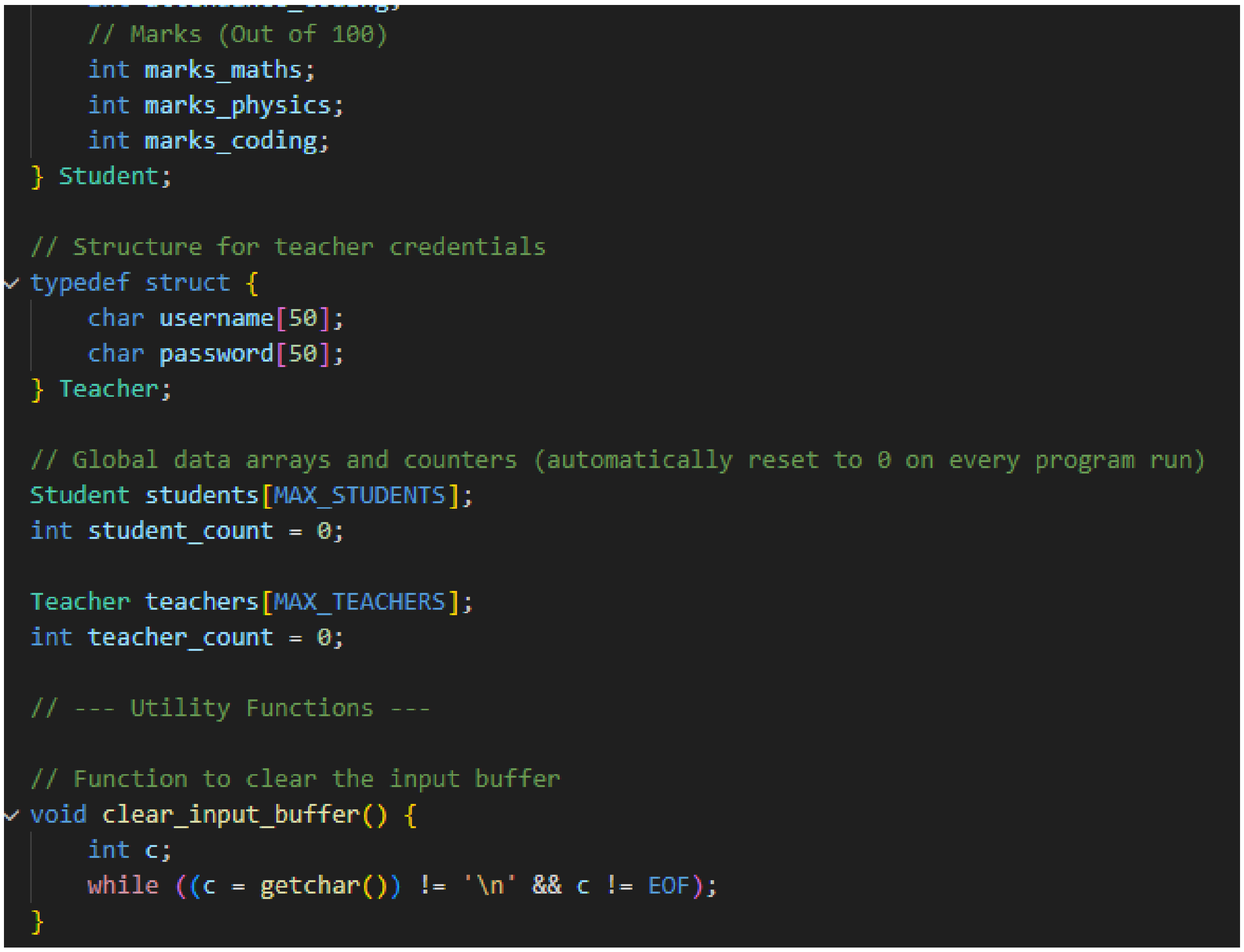
C Concept	Usage in Project
Loops	for loops (e.g., in find_student_index and array iteration) and do-while loops (e.g., in home_menu and teacher_portal) for continuous execution until a condition is met.
Conditional Logic	if-else if-else statements (e.g., in login checks and input validation) and switch statements (e.g., in all menu functions).
String Handling	Functions from <string.h>: strcmp for comparing passwords and usernames, strlen for checking SAP ID length, and strcpy for data assignment.
Input/Output	printf for displaying output, scanf for reading simple integer/string input, and fgets for reading a student name that may contain spaces.
Pointers	Used in teacher_edit_student_data (e.g., int* mark_ptr) to efficiently update different variables based on user selection, and implicitly in passing arrays to functions.

C Concept	Usage in Project
Enumerations/Constants	Using #define to create constants for limits (MAX_STUDENTS) and ANSI color codes (C_BLUE).

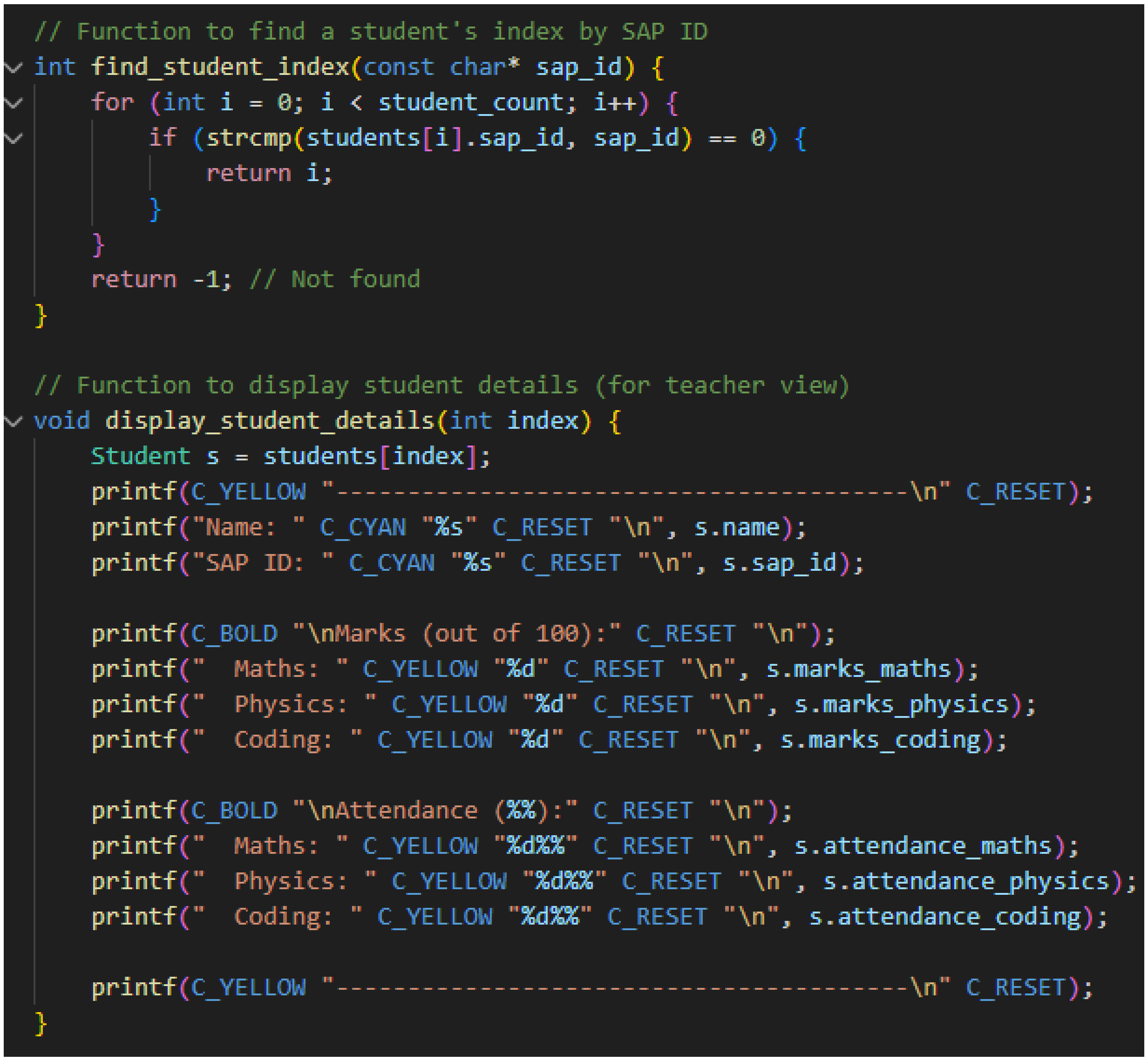
C PROGRAMMING CODE:-



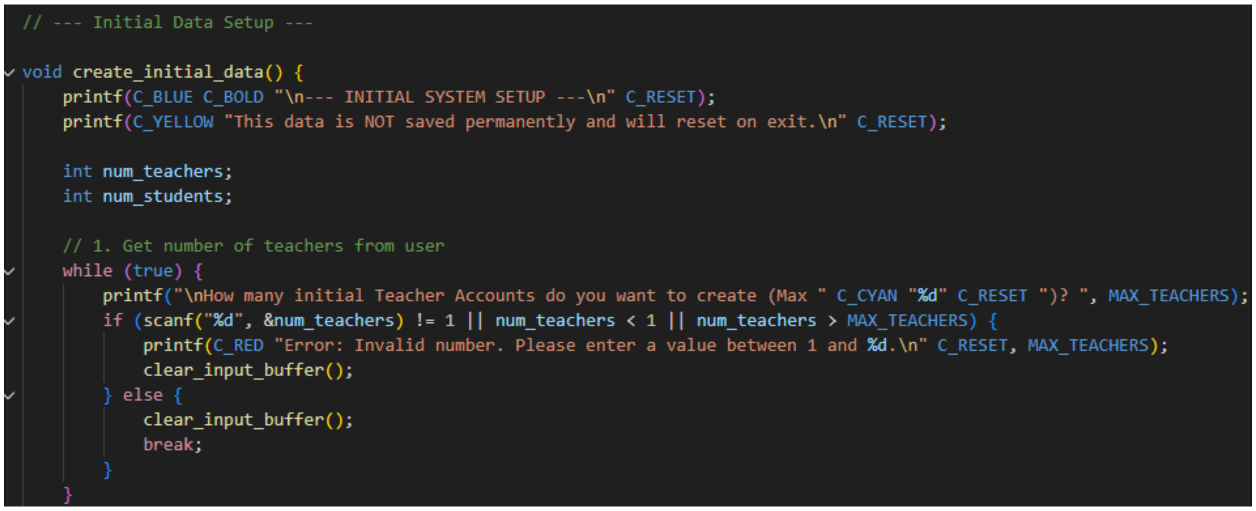
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

// --- Constants and Global Limits ---
#define MAX_STUDENTS 100
#define MAX_TEACHERS 10
#define SAP_ID_LENGTH 9

// --- ANSI Color Codes ---
// Blue/Cyan/Yellow for the main theme and highlights
#define C_RESET   "\x1b[0m"
#define C_BOLD    "\x1b[1m"
#define C_BLUE    "\x1b[34m"
#define C_CYAN    "\x1b[36m"
#define C_YELLOW  "\x1b[33m"
// Green for success, Red for errors
#define C_GREEN   "\x1b[32m"
#define C_RED     "\x1b[31m"

// --- Data Structures ---

// Structure for student records
typedef struct {
    char sap_id[SAP_ID_LENGTH + 1]; // 9-digit ID + null terminator
    char password[20];
    char name[50];
    // Attendance (Out of 100)
    int attendance_maths;
    int attendance_physics;
    int attendance_coding;
```



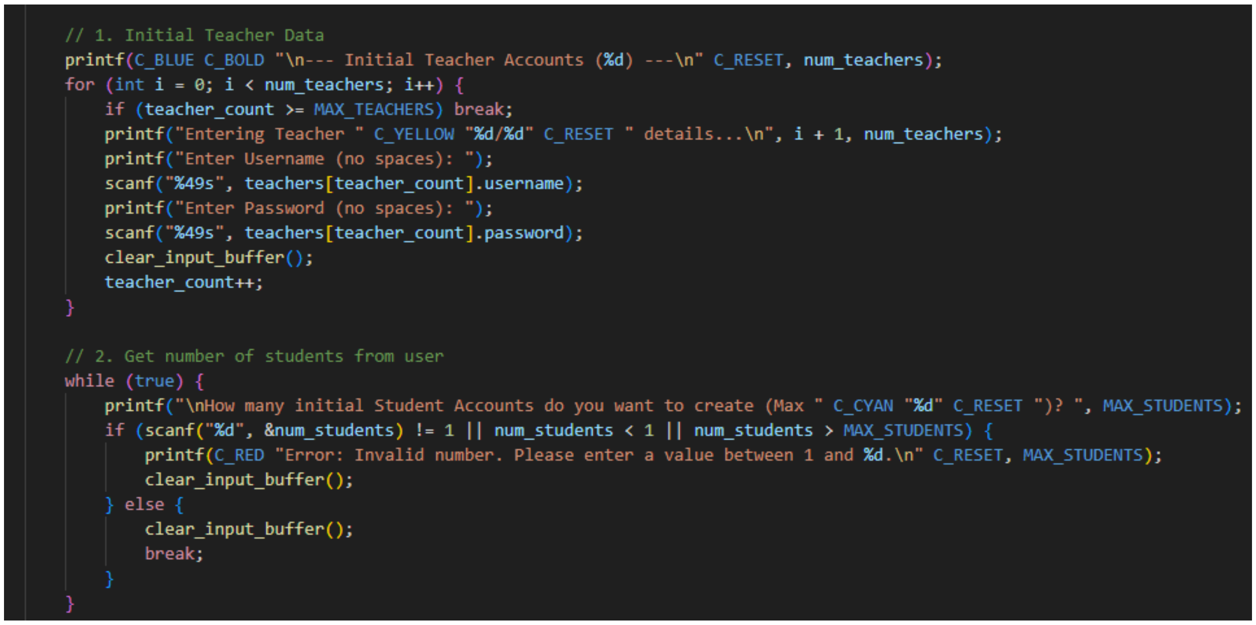
```c
    // Marks (Out of 100)
    int marks_maths;
    int marks_physics;
    int marks_coding;
} Student;

// Structure for teacher credentials
typedef struct {
    char username[50];
    char password[50];
} Teacher;

// Global data arrays and counters (automatically reset to 0 on every program run)
Student students[MAX_STUDENTS];
int student_count = 0;

Teacher teachers[MAX_TEACHERS];
int teacher_count = 0;

// --- Utility Functions ---

// Function to clear the input buffer
void clear_input_buffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}
```



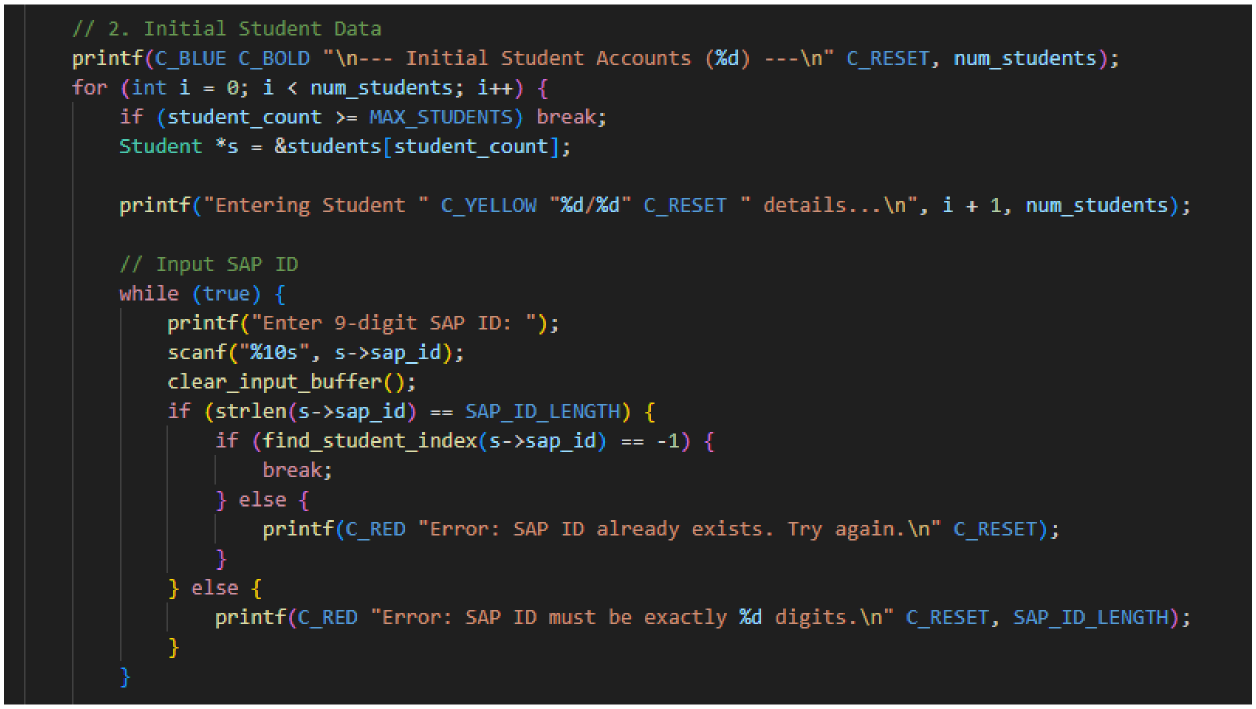
```c
// Function to find a student's index by SAP ID
int find_student_index(const char* sap_id) {
    for (int i = 0; i < student_count; i++) {
        if (strcmp(students[i].sap_id, sap_id) == 0) {
            return i;
        }
    }
    return -1; // Not found
}

// Function to display student details (for teacher view)
void display_student_details(int index) {
    Student s = students[index];
    printf(C_YELLOW "----------------------------------------\n" C_RESET);
    printf("Name: " C_CYAN "%s" C_RESET "\n", s.name);
    printf("SAP ID: " C_CYAN "%s" C_RESET "\n", s.sap_id);

    printf(C_BOLD "\nMarks (out of 100):" C_RESET "\n");
    printf("  Maths: " C_YELLOW "%d" C_RESET "\n", s.marks_maths);
    printf("  Physics: " C_YELLOW "%d" C_RESET "\n", s.marks_physics);
    printf("  Coding: " C_YELLOW "%d" C_RESET "\n", s.marks_coding);

    printf(C_BOLD "\nAttendance (%%):" C_RESET "\n");
    printf("  Maths: " C_YELLOW "%d%%" C_RESET "\n", s.attendance_maths);
    printf("  Physics: " C_YELLOW "%d%%" C_RESET "\n", s.attendance_physics);
    printf("  Coding: " C_YELLOW "%d%%" C_RESET "\n", s.attendance_coding);

    printf(C_YELLOW "----------------------------------------\n" C_RESET);
}
```



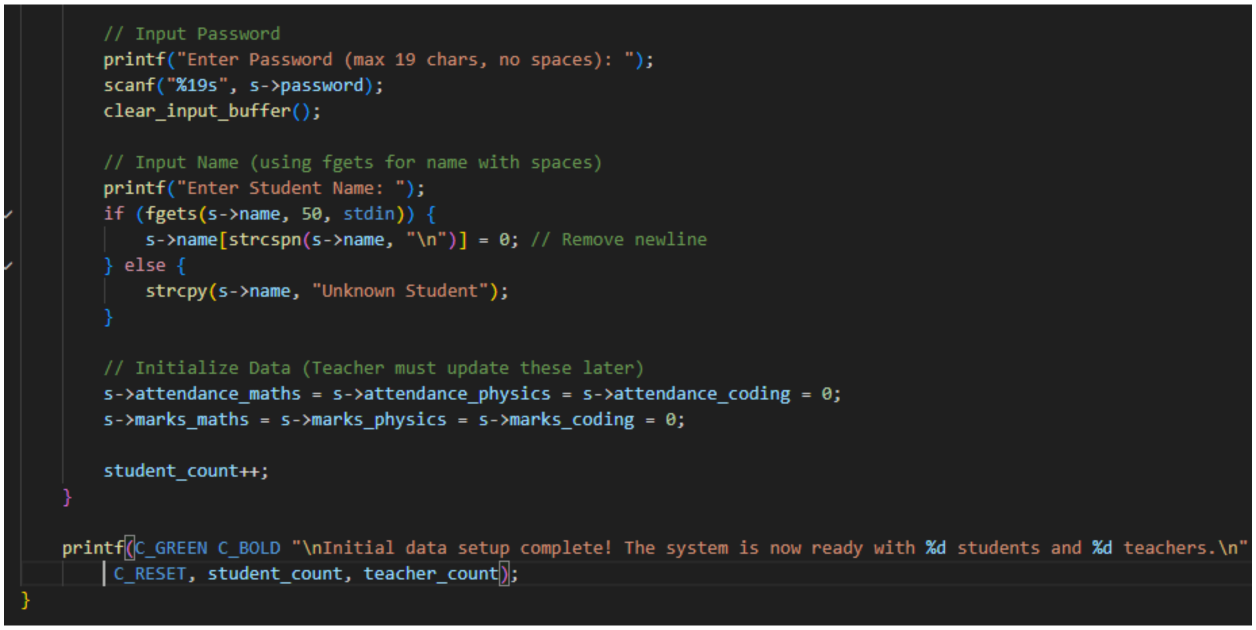
```c
// --- Initial Data Setup ---

void create_initial_data() {
    printf(C_BLUE C_BOLD "\n--- INITIAL SYSTEM SETUP ---\n" C_RESET);
    printf(C_YELLOW "This data is NOT saved permanently and will reset on exit.\n" C_RESET);

    int num_teachers;
    int num_students;

    // 1. Get number of teachers from user
    while (true) {
        printf("\nHow many initial Teacher Accounts do you want to create (Max " C_CYAN "%d" C_RESET ")? ", MAX_TEACHERS);
        if (scanf("%d", &num_teachers) != 1 || num_teachers < 1 || num_teachers > MAX_TEACHERS) {
            printf(C_RED "Error: Invalid number. Please enter a value between 1 and %d.\n" C_RESET, MAX_TEACHERS);
            clear_input_buffer();
        } else {
            clear_input_buffer();
            break;
        }
    }
```



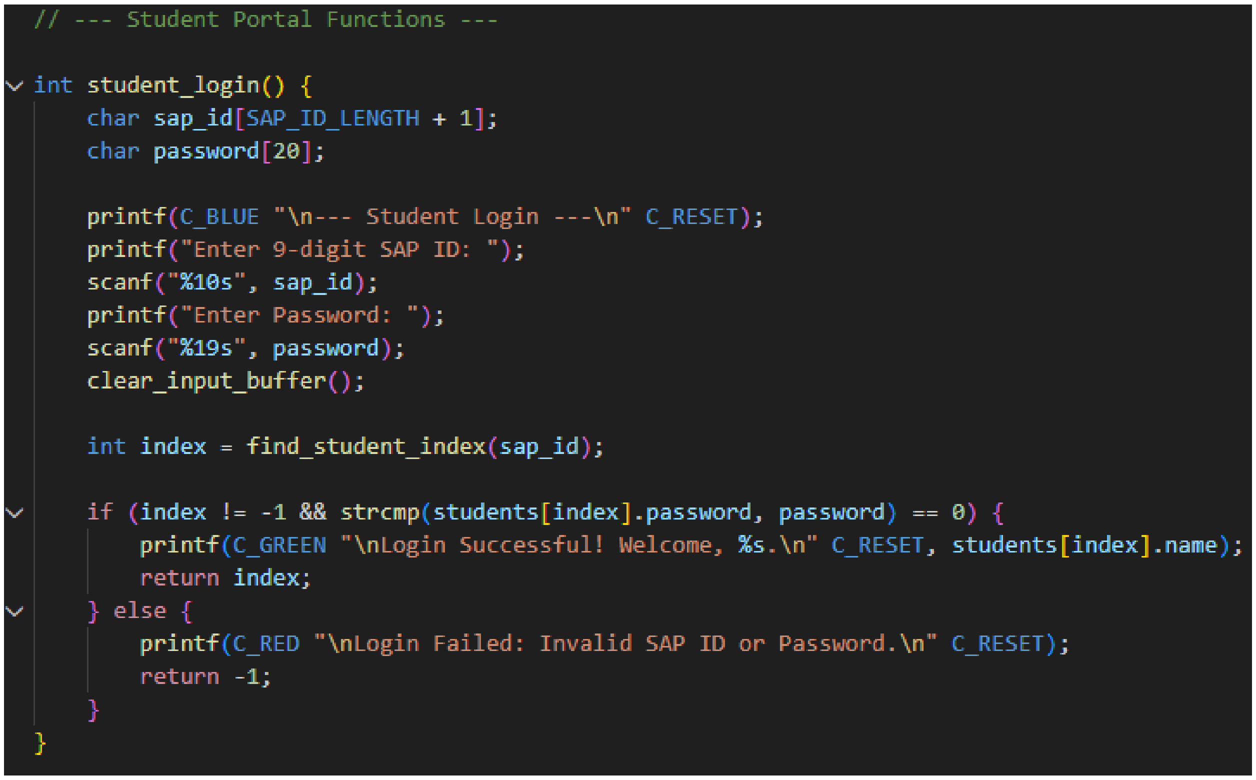
```c
// 1. Initial Teacher Data
printf(C_BLUE C_BOLD "\n--- Initial Teacher Accounts (%d) ---\n" C_RESET, num_teachers);
for (int i = 0; i < num_teachers; i++) {
    if (teacher_count >= MAX_TEACHERS) break;
    printf("Entering Teacher " C_YELLOW "%d/%d" C_RESET " details...\n", i + 1, num_teachers);
    printf("Enter Username (no spaces): ");
    scanf("%49s", teachers[teacher_count].username);
    printf("Enter Password (no spaces): ");
    scanf("%49s", teachers[teacher_count].password);
    clear_input_buffer();
    teacher_count++;
}

// 2. Get number of students from user
while (true) {
    printf("\nHow many initial Student Accounts do you want to create (Max " C_CYAN "%d" C_RESET ")? ", MAX_STUDENTS);
    if (scanf("%d", &num_students) != 1 || num_students < 1 || num_students > MAX_STUDENTS) {
        printf(C_RED "Error: Invalid number. Please enter a value between 1 and %d.\n" C_RESET, MAX_STUDENTS);
        clear_input_buffer();
    } else {
        clear_input_buffer();
        break;
    }
}
```



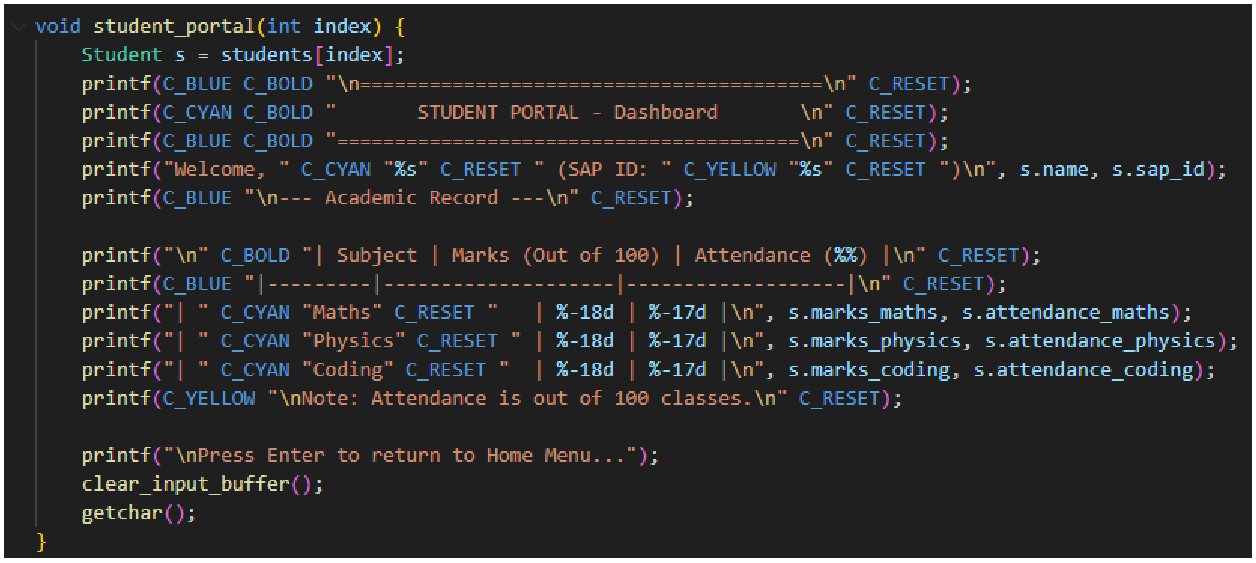
```c
// 2. Initial Student Data
printf(C_BLUE C_BOLD "\n--- Initial Student Accounts (%d) ---\n" C_RESET, num_students);
for (int i = 0; i < num_students; i++) {
    if (student_count >= MAX_STUDENTS) break;
    Student *s = &students[student_count];

    printf("Entering Student " C_YELLOW "%d/%d" C_RESET " details...\n", i + 1, num_students);

    // Input SAP ID
    while (true) {
        printf("Enter 9-digit SAP ID: ");
        scanf("%10s", s->sap_id);
        clear_input_buffer();
        if (strlen(s->sap_id) == SAP_ID_LENGTH) {
            if (find_student_index(s->sap_id) == -1) {
                break;
            } else {
                printf(C_RED "Error: SAP ID already exists. Try again.\n" C_RESET);
            }
        } else {
            printf(C_RED "Error: SAP ID must be exactly %d digits.\n" C_RESET, SAP_ID_LENGTH);
        }
    }
```



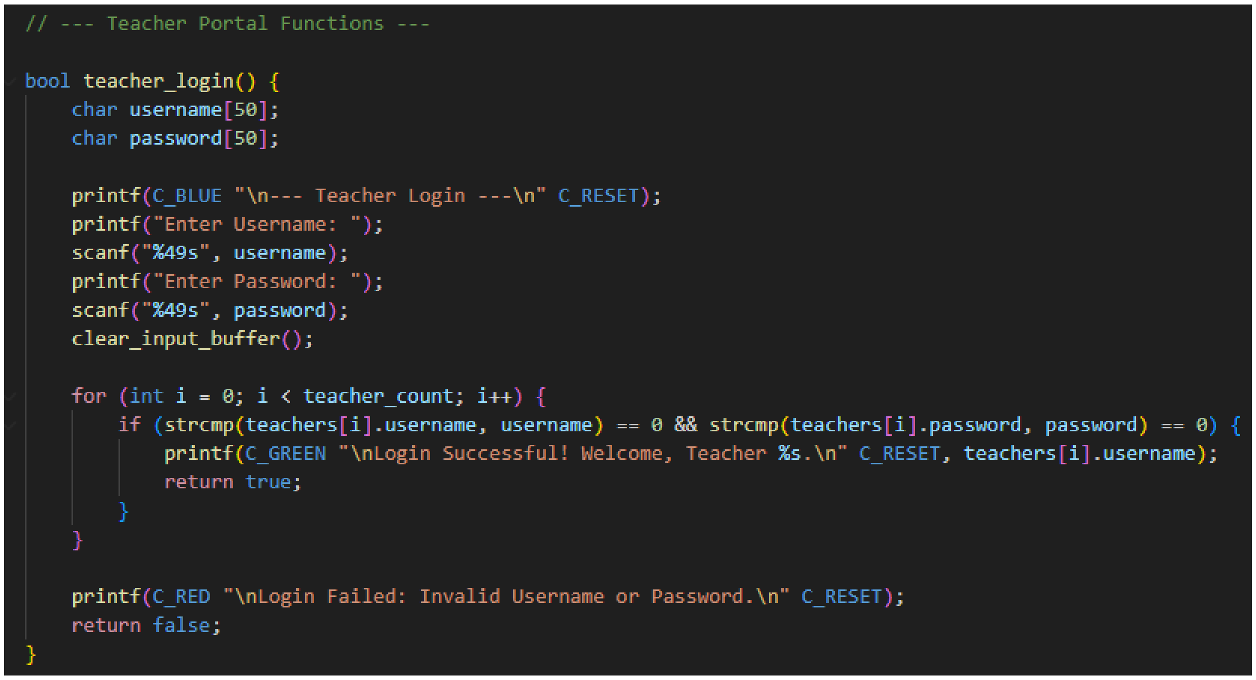
```c
        // Input Password
        printf("Enter Password (max 19 chars, no spaces): ");
        scanf("%19s", s->password);
        clear_input_buffer();

        // Input Name (using fgets for name with spaces)
        printf("Enter Student Name: ");
        if (fgets(s->name, 50, stdin)) {
            s->name[strcspn(s->name, "\n")] = 0; // Remove newline
        } else {
            strcpy(s->name, "Unknown Student");
        }

        // Initialize Data (Teacher must update these later)
        s->attendance_maths = s->attendance_physics = s->attendance_coding = 0;
        s->marks_maths = s->marks_physics = s->marks_coding = 0;

        student_count++;
    }

    printf(C_GREEN C_BOLD "\nInitial data setup complete! The system is now ready with %d students and %d teachers.\n"
        C_RESET, student_count, teacher_count);
}
```



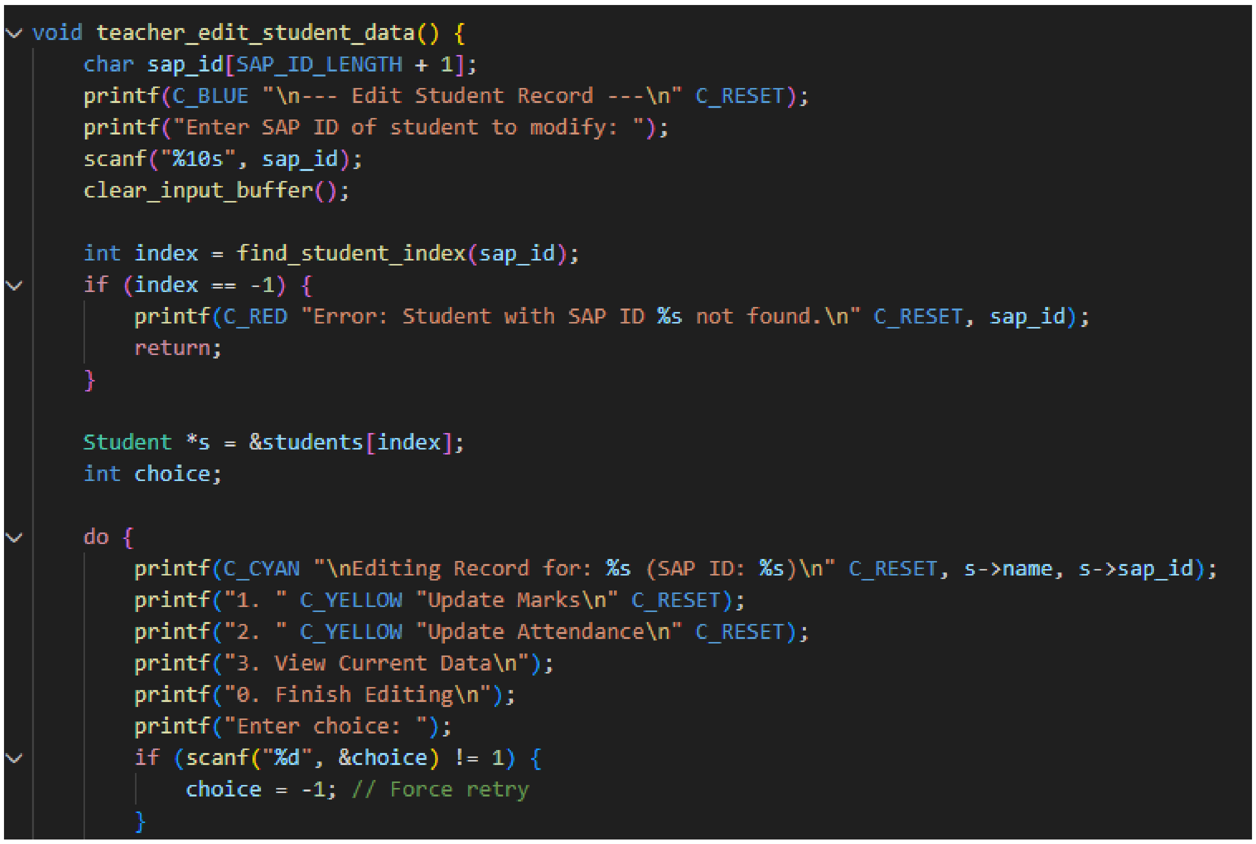
```c
// --- Student Portal Functions ---

int student_login() {
    char sap_id[SAP_ID_LENGTH + 1];
    char password[20];

    printf(C_BLUE "\n--- Student Login ---\n" C_RESET);
    printf("Enter 9-digit SAP ID: ");
    scanf("%10s", sap_id);
    printf("Enter Password: ");
    scanf("%19s", password);
    clear_input_buffer();

    int index = find_student_index(sap_id);

    if (index != -1 && strcmp(students[index].password, password) == 0) {
        printf(C_GREEN "\nLogin Successful! Welcome, %s.\n" C_RESET, students[index].name);
        return index;
    } else {
        printf(C_RED "\nLogin Failed: Invalid SAP ID or Password.\n" C_RESET);
        return -1;
    }
}
```



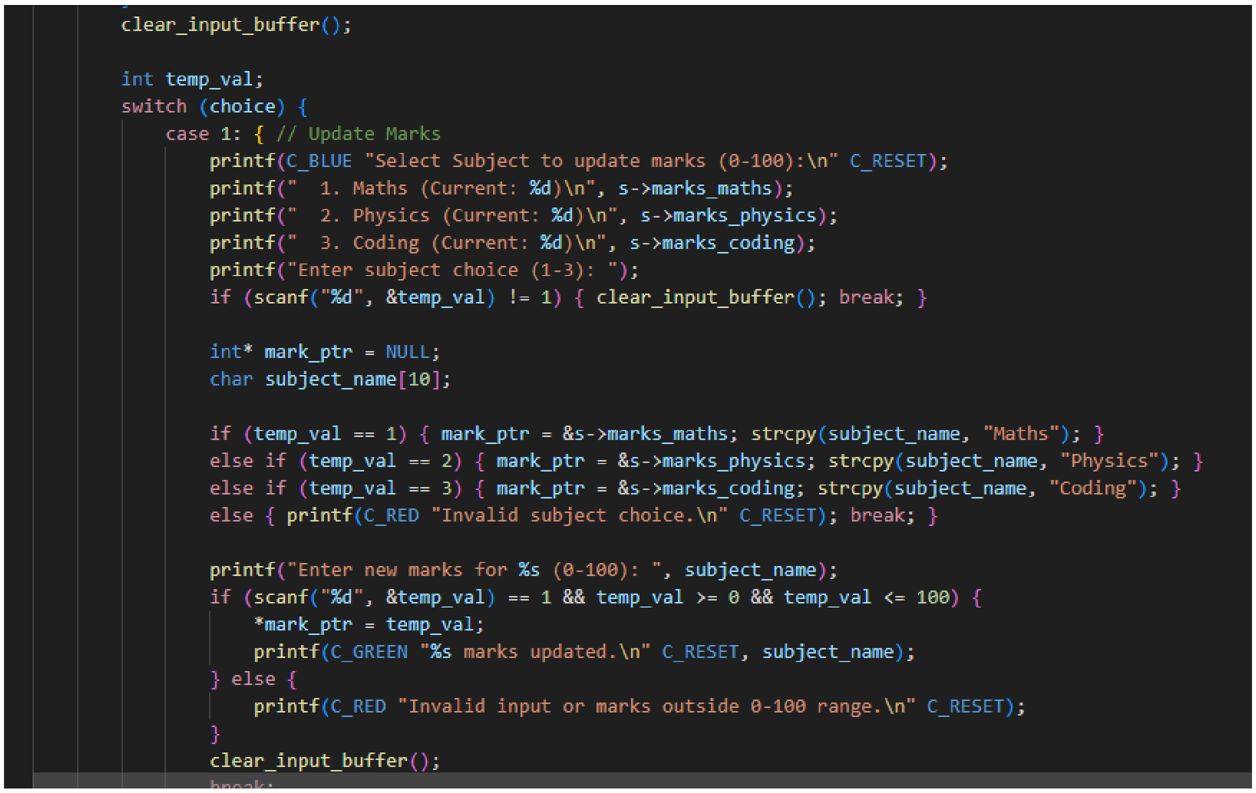
```c
void student_portal(int index) {
    Student s = students[index];
    printf(C_BLUE C_BOLD "\n========================================\n" C_RESET);
    printf(C_CYAN C_BOLD "       STUDENT PORTAL - Dashboard       \n" C_RESET);
    printf(C_BLUE C_BOLD "========================================\n" C_RESET);
    printf("Welcome, " C_CYAN "%s" C_RESET " (SAP ID: " C_YELLOW "%s" C_RESET ")\n", s.name, s.sap_id);
    printf(C_BLUE "\n--- Academic Record ---\n" C_RESET);

    printf("\n" C_BOLD "| Subject | Marks (Out of 100) | Attendance (%%) |\n" C_RESET);
    printf(C_BLUE "|---------|--------------------|-------------------|\n" C_RESET);
    printf("| " C_CYAN "Maths" C_RESET "   | %-18d | %-17d |\n", s.marks_maths, s.attendance_maths);
    printf("| " C_CYAN "Physics" C_RESET " | %-18d | %-17d |\n", s.marks_physics, s.attendance_physics);
    printf("| " C_CYAN "Coding" C_RESET "  | %-18d | %-17d |\n", s.marks_coding, s.attendance_coding);
    printf(C_YELLOW "\nNote: Attendance is out of 100 classes.\n" C_RESET);

    printf("\nPress Enter to return to Home Menu...");
    clear_input_buffer();
    getchar();
}
```



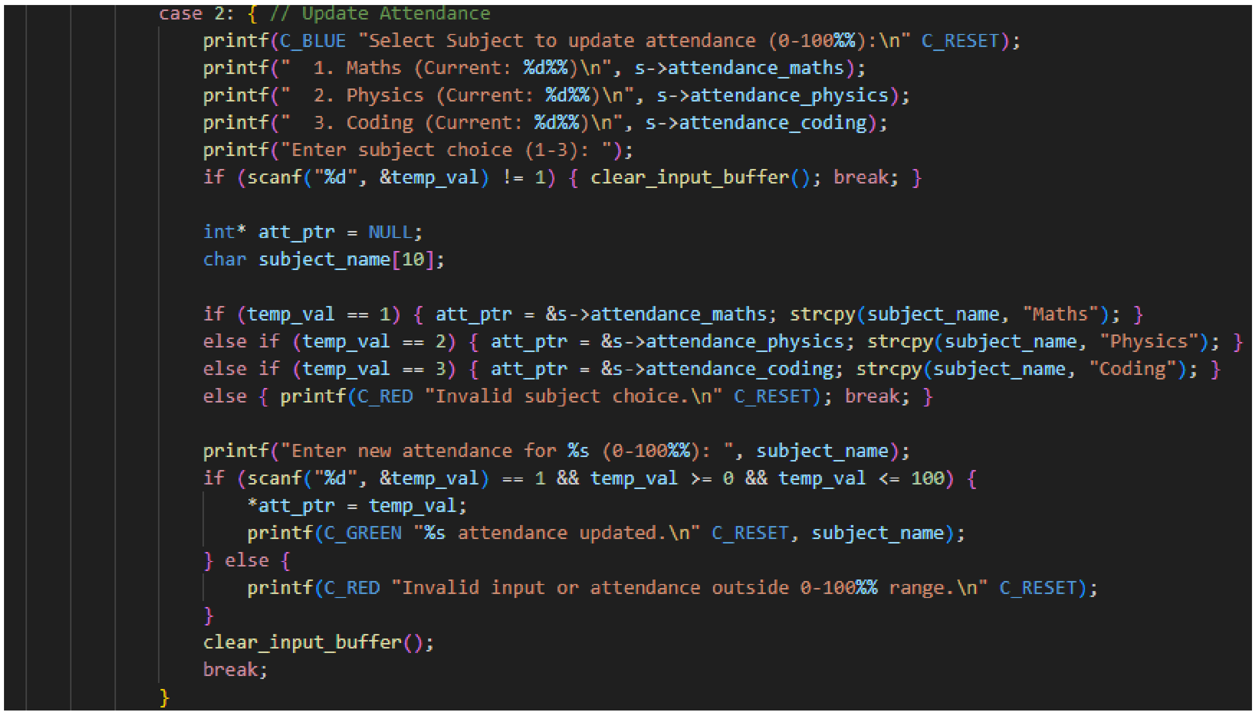
```c
// --- Teacher Portal Functions ---

bool teacher_login() {
    char username[50];
    char password[50];

    printf(C_BLUE "\n--- Teacher Login ---\n" C_RESET);
    printf("Enter Username: ");
    scanf("%49s", username);
    printf("Enter Password: ");
    scanf("%49s", password);
    clear_input_buffer();

    for (int i = 0; i < teacher_count; i++) {
        if (strcmp(teachers[i].username, username) == 0 && strcmp(teachers[i].password, password) == 0) {
            printf(C_GREEN "\nLogin Successful! Welcome, Teacher %s.\n" C_RESET, teachers[i].username);
            return true;
        }
    }

    printf(C_RED "\nLogin Failed: Invalid Username or Password.\n" C_RESET);
    return false;
}
```



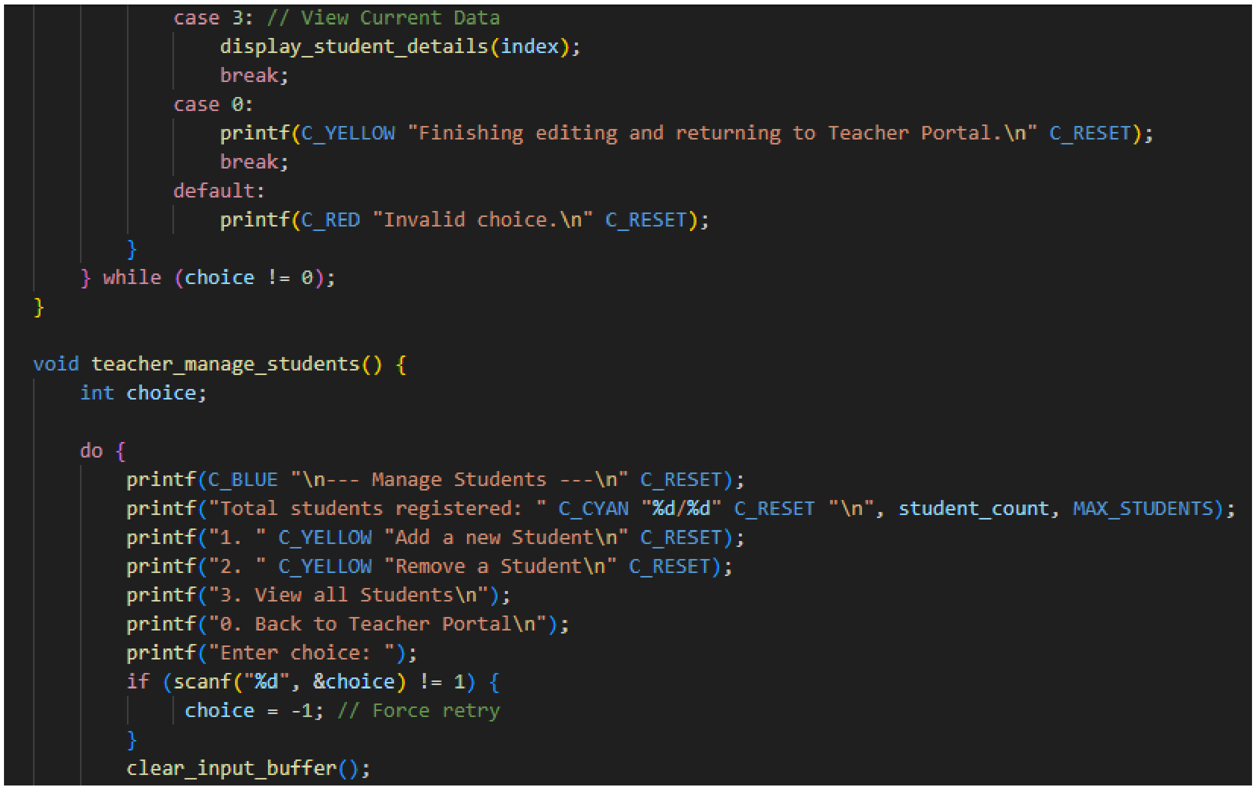
```c
void teacher_edit_student_data() {
    char sap_id[SAP_ID_LENGTH + 1];
    printf(C_BLUE "\n--- Edit Student Record ---\n" C_RESET);
    printf("Enter SAP ID of student to modify: ");
    scanf("%10s", sap_id);
    clear_input_buffer();

    int index = find_student_index(sap_id);
    if (index == -1) {
        printf(C_RED "Error: Student with SAP ID %s not found.\n" C_RESET, sap_id);
        return;
    }

    Student *s = &students[index];
    int choice;

    do {
        printf(C_CYAN "\nEditing Record for: %s (SAP ID: %s)\n" C_RESET, s->name, s->sap_id);
        printf("1. " C_YELLOW "Update Marks\n" C_RESET);
        printf("2. " C_YELLOW "Update Attendance\n" C_RESET);
        printf("3. View Current Data\n");
        printf("0. Finish Editing\n");
        printf("Enter choice: ");
        if (scanf("%d", &choice) != 1) {
            choice = -1; // Force retry
        }
```



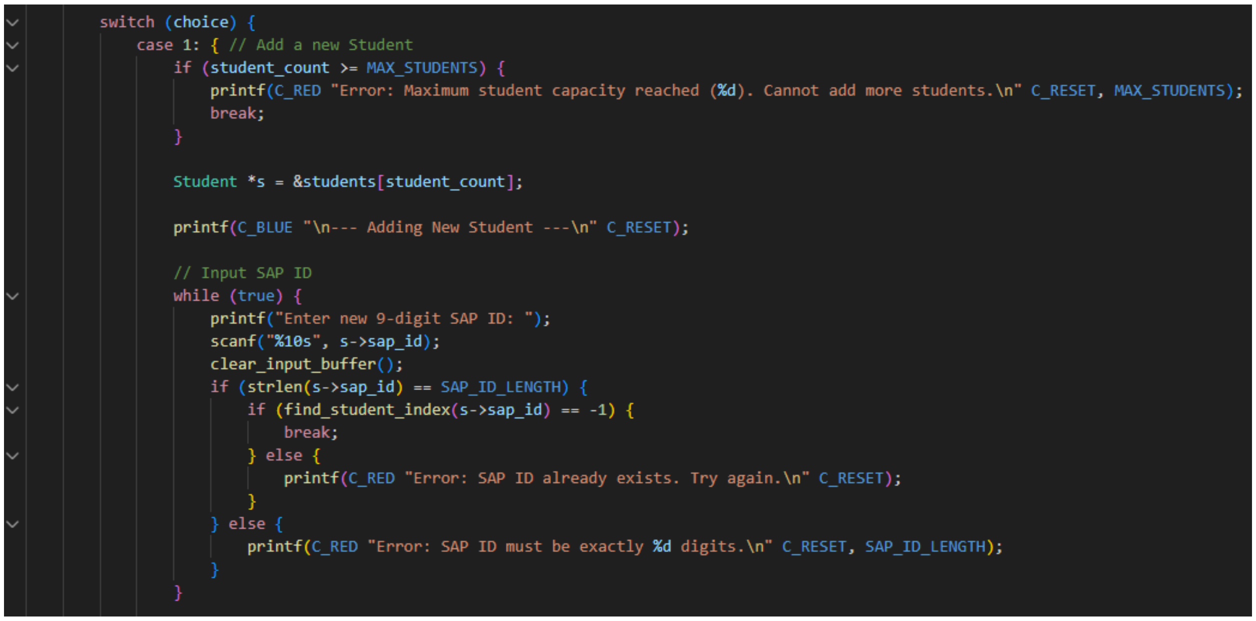
```c
        clear_input_buffer();

        int temp_val;
        switch (choice) {
            case 1: { // Update Marks
                printf(C_BLUE "Select Subject to update marks (0-100):\n" C_RESET);
                printf("  1. Maths (Current: %d)\n", s->marks_maths);
                printf("  2. Physics (Current: %d)\n", s->marks_physics);
                printf("  3. Coding (Current: %d)\n", s->marks_coding);
                printf("Enter subject choice (1-3): ");
                if (scanf("%d", &temp_val) != 1) { clear_input_buffer(); break; }

                int* mark_ptr = NULL;
                char subject_name[10];

                if (temp_val == 1) { mark_ptr = &s->marks_maths; strcpy(subject_name, "Maths"); }
                else if (temp_val == 2) { mark_ptr = &s->marks_physics; strcpy(subject_name, "Physics"); }
                else if (temp_val == 3) { mark_ptr = &s->marks_coding; strcpy(subject_name, "Coding"); }
                else { printf(C_RED "Invalid subject choice.\n" C_RESET); break; }

                printf("Enter new marks for %s (0-100): ", subject_name);
                if (scanf("%d", &temp_val) == 1 && temp_val >= 0 && temp_val <= 100) {
                    *mark_ptr = temp_val;
                    printf(C_GREEN "%s marks updated.\n" C_RESET, subject_name);
                } else {
                    printf(C_RED "Invalid input or marks outside 0-100 range.\n" C_RESET);
                }
                clear_input_buffer();
                break;
```



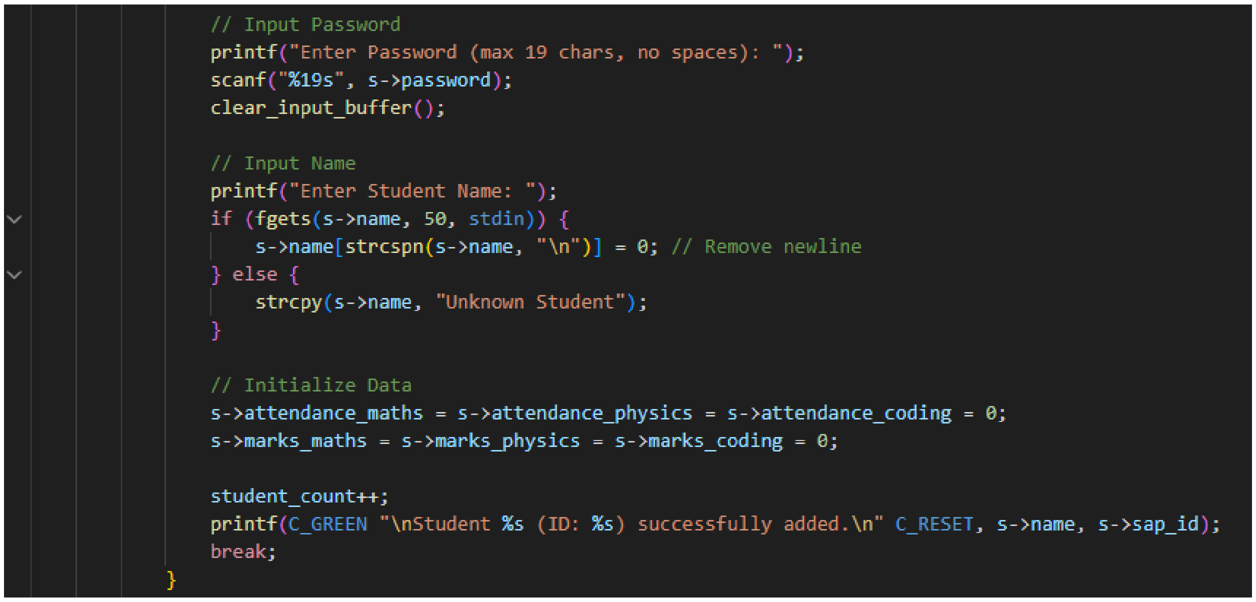
```c
            case 2: { // Update Attendance
                printf(C_BLUE "Select Subject to update attendance (0-100%%):\n" C_RESET);
                printf("  1. Maths (Current: %d%%)\n", s->attendance_maths);
                printf("  2. Physics (Current: %d%%)\n", s->attendance_physics);
                printf("  3. Coding (Current: %d%%)\n", s->attendance_coding);
                printf("Enter subject choice (1-3): ");
                if (scanf("%d", &temp_val) != 1) { clear_input_buffer(); break; }

                int* att_ptr = NULL;
                char subject_name[10];

                if (temp_val == 1) { att_ptr = &s->attendance_maths; strcpy(subject_name, "Maths"); }
                else if (temp_val == 2) { att_ptr = &s->attendance_physics; strcpy(subject_name, "Physics"); }
                else if (temp_val == 3) { att_ptr = &s->attendance_coding; strcpy(subject_name, "Coding"); }
                else { printf(C_RED "Invalid subject choice.\n" C_RESET); break; }

                printf("Enter new attendance for %s (0-100%%): ", subject_name);
                if (scanf("%d", &temp_val) == 1 && temp_val >= 0 && temp_val <= 100) {
                    *att_ptr = temp_val;
                    printf(C_GREEN "%s attendance updated.\n" C_RESET, subject_name);
                } else {
                    printf(C_RED "Invalid input or attendance outside 0-100%% range.\n" C_RESET);
                }
                clear_input_buffer();
                break;
            }
```



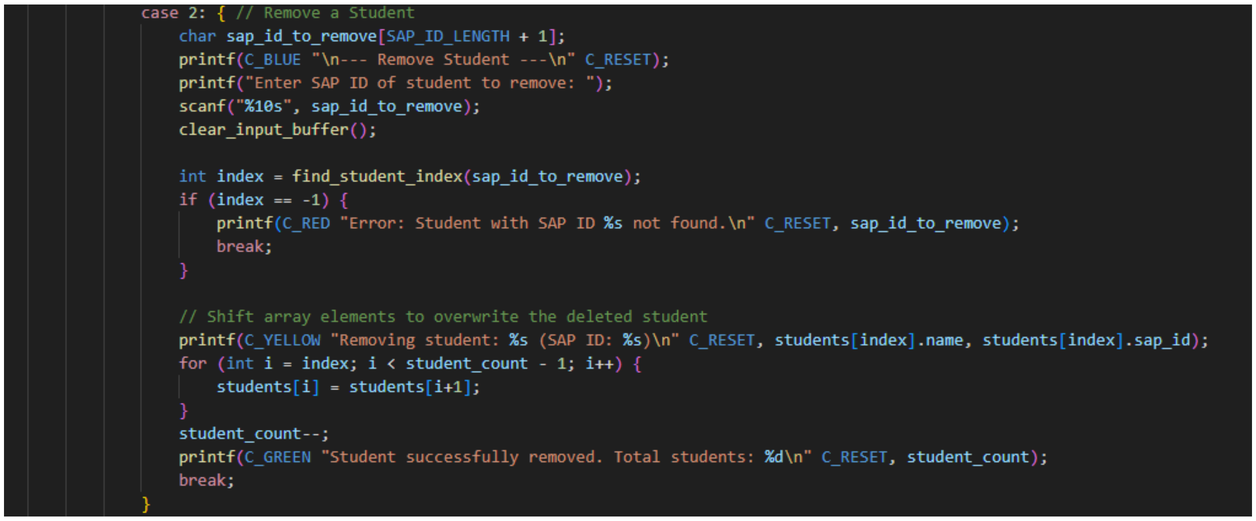
```c
            case 3: // View Current Data
                display_student_details(index);
                break;
            case 0:
                printf(C_YELLOW "Finishing editing and returning to Teacher Portal.\n" C_RESET);
                break;
            default:
                printf(C_RED "Invalid choice.\n" C_RESET);
        }
    } while (choice != 0);
}

void teacher_manage_students() {
    int choice;

    do {
        printf(C_BLUE "\n--- Manage Students ---\n" C_RESET);
        printf("Total students registered: " C_CYAN "%d/%d" C_RESET "\n", student_count, MAX_STUDENTS);
        printf("1. " C_YELLOW "Add a new Student\n" C_RESET);
        printf("2. " C_YELLOW "Remove a Student\n" C_RESET);
        printf("3. View all Students\n");
        printf("0. Back to Teacher Portal\n");
        printf("Enter choice: ");
        if (scanf("%d", &choice) != 1) {
            choice = -1; // Force retry
        }
        clear_input_buffer();
```



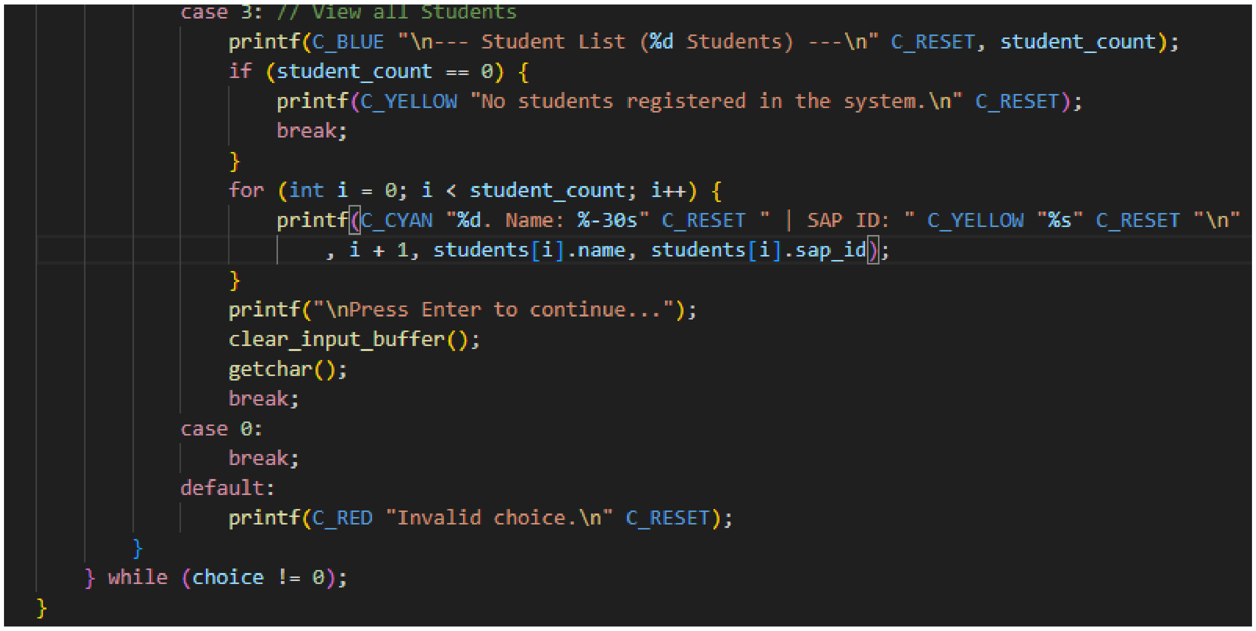
```c
        switch (choice) {
            case 1: { // Add a new Student
                if (student_count >= MAX_STUDENTS) {
                    printf(C_RED "Error: Maximum student capacity reached (%d). Cannot add more students.\n" C_RESET, MAX_STUDENTS);
                    break;
                }

                Student *s = &students[student_count];

                printf(C_BLUE "\n--- Adding New Student ---\n" C_RESET);

                // Input SAP ID
                while (true) {
                    printf("Enter new 9-digit SAP ID: ");
                    scanf("%10s", s->sap_id);
                    clear_input_buffer();
                    if (strlen(s->sap_id) == SAP_ID_LENGTH) {
                        if (find_student_index(s->sap_id) == -1) {
                            break;
                        } else {
                            printf(C_RED "Error: SAP ID already exists. Try again.\n" C_RESET);
                        }
                    } else {
                        printf(C_RED "Error: SAP ID must be exactly %d digits.\n" C_RESET, SAP_ID_LENGTH);
                    }
                }
```



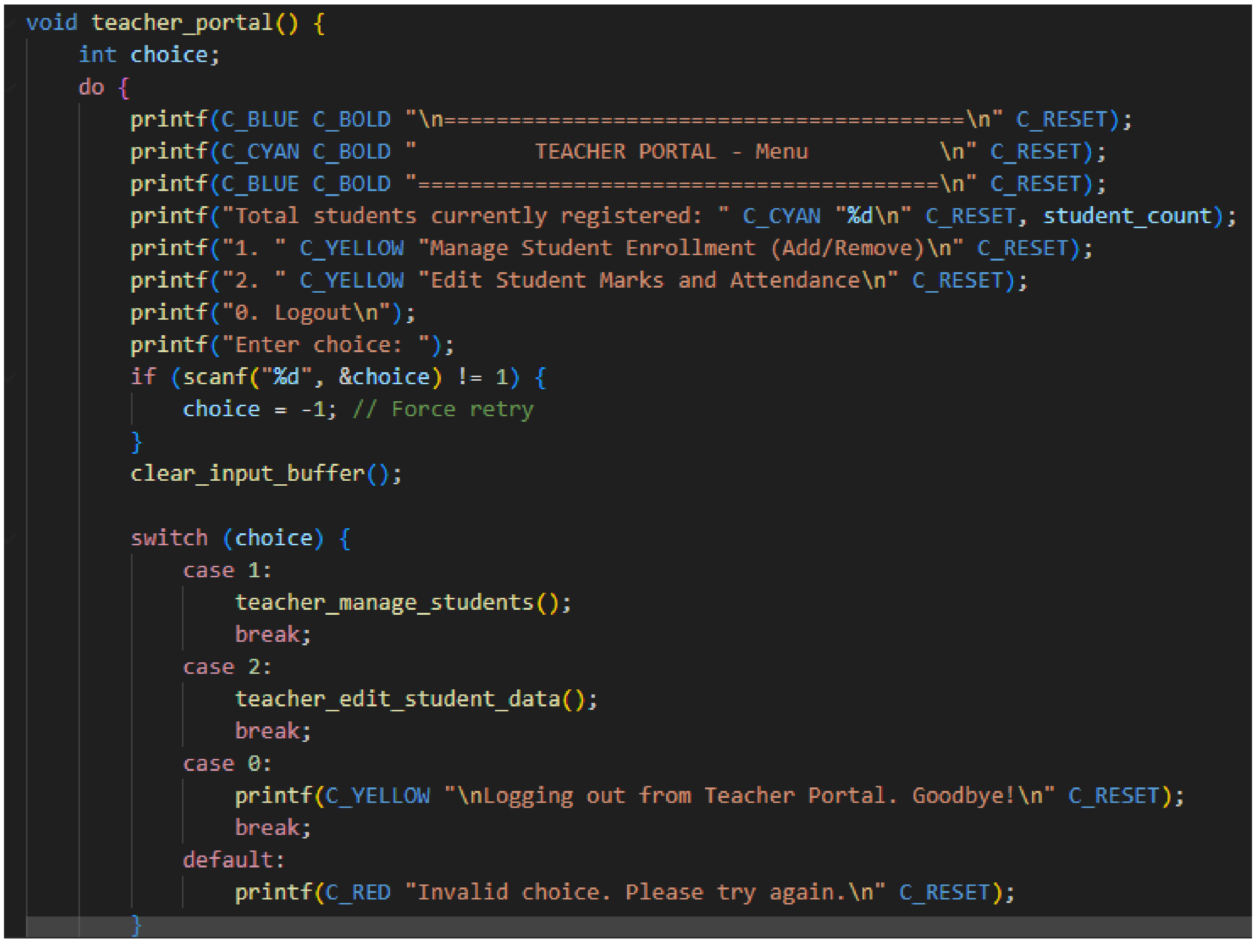
```c
                // Input Password
                printf("Enter Password (max 19 chars, no spaces): ");
                scanf("%19s", s->password);
                clear_input_buffer();

                // Input Name
                printf("Enter Student Name: ");
                if (fgets(s->name, 50, stdin)) {
                    s->name[strcspn(s->name, "\n")] = 0; // Remove newline
                } else {
                    strcpy(s->name, "Unknown Student");
                }

                // Initialize Data
                s->attendance_maths = s->attendance_physics = s->attendance_coding = 0;
                s->marks_maths = s->marks_physics = s->marks_coding = 0;

                student_count++;
                printf(C_GREEN "\nStudent %s (ID: %s) successfully added.\n" C_RESET, s->name, s->sap_id);
                break;
            }
```



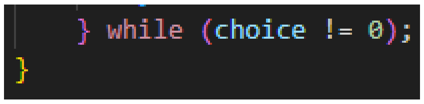
```
            case 2: { // Remove a Student
                char sap_id_to_remove[SAP_ID_LENGTH + 1];
                printf(C_BLUE "\n--- Remove Student ---\n" C_RESET);
                printf("Enter SAP ID of student to remove: ");
                scanf("%10s", sap_id_to_remove);
                clear_input_buffer();

                int index = find_student_index(sap_id_to_remove);
                if (index == -1) {
                    printf(C_RED "Error: Student with SAP ID %s not found.\n" C_RESET, sap_id_to_remove);
                    break;
                }

                // Shift array elements to overwrite the deleted student
                printf(C_YELLOW "Removing student: %s (SAP ID: %s)\n" C_RESET, students[index].name, students[index].sap_id);
                for (int i = index; i < student_count - 1; i++) {
                    students[i] = students[i+1];
                }
                student_count--;
                printf(C_GREEN "Student successfully removed. Total students: %d\n" C_RESET, student_count);
                break;
            }
```



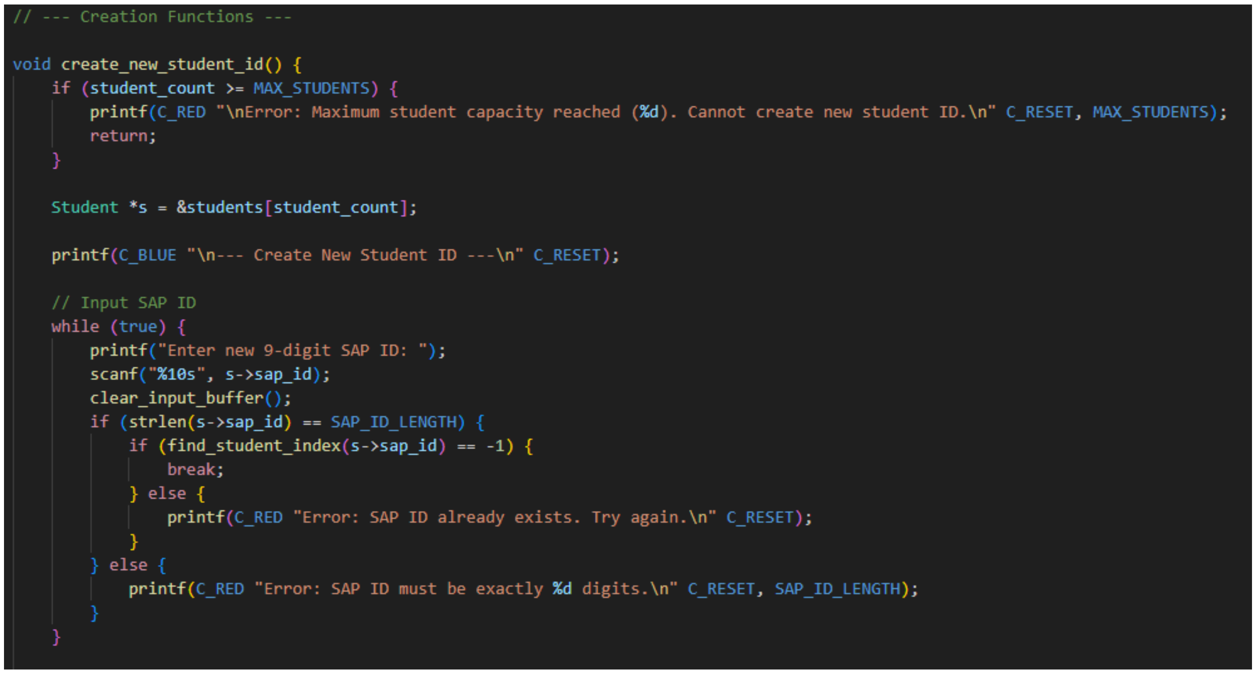
```
            case 3: // View all Students
                printf(C_BLUE "\n--- Student List (%d Students) ---\n" C_RESET, student_count);
                if (student_count == 0) {
                    printf(C_YELLOW "No students registered in the system.\n" C_RESET);
                    break;
                }
                for (int i = 0; i < student_count; i++) {
                    printf(C_CYAN "%d. Name: %-30s" C_RESET " | SAP ID: " C_YELLOW "%s" C_RESET "\n"
                        , i + 1, students[i].name, students[i].sap_id);
                }
                printf("\nPress Enter to continue...");
                clear_input_buffer();
                getchar();
                break;
            case 0:
                break;
            default:
                printf(C_RED "Invalid choice.\n" C_RESET);
        }
    } while (choice != 0);
}
```



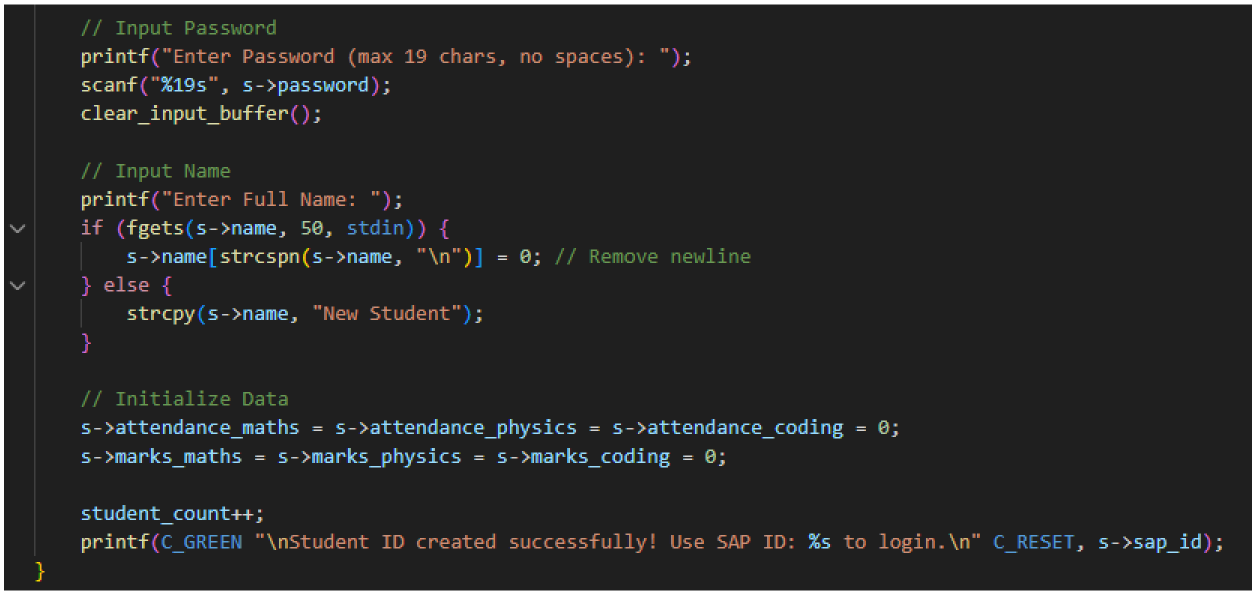
```c
void teacher_portal() {
    int choice;
    do {
        printf(C_BLUE C_BOLD "\n========================================\n" C_RESET);
        printf(C_CYAN C_BOLD "         TEACHER PORTAL - Menu          \n" C_RESET);
        printf(C_BLUE C_BOLD "========================================\n" C_RESET);
        printf("Total students currently registered: " C_CYAN "%d\n" C_RESET, student_count);
        printf("1. " C_YELLOW "Manage Student Enrollment (Add/Remove)\n" C_RESET);
        printf("2. " C_YELLOW "Edit Student Marks and Attendance\n" C_RESET);
        printf("0. Logout\n");
        printf("Enter choice: ");
        if (scanf("%d", &choice) != 1) {
            choice = -1; // Force retry
        }
        clear_input_buffer();

        switch (choice) {
            case 1:
                teacher_manage_students();
                break;
            case 2:
                teacher_edit_student_data();
                break;
            case 0:
                printf(C_YELLOW "\nLogging out from Teacher Portal. Goodbye!\n" C_RESET);
                break;
            default:
                printf(C_RED "Invalid choice. Please try again.\n" C_RESET);
        }
```



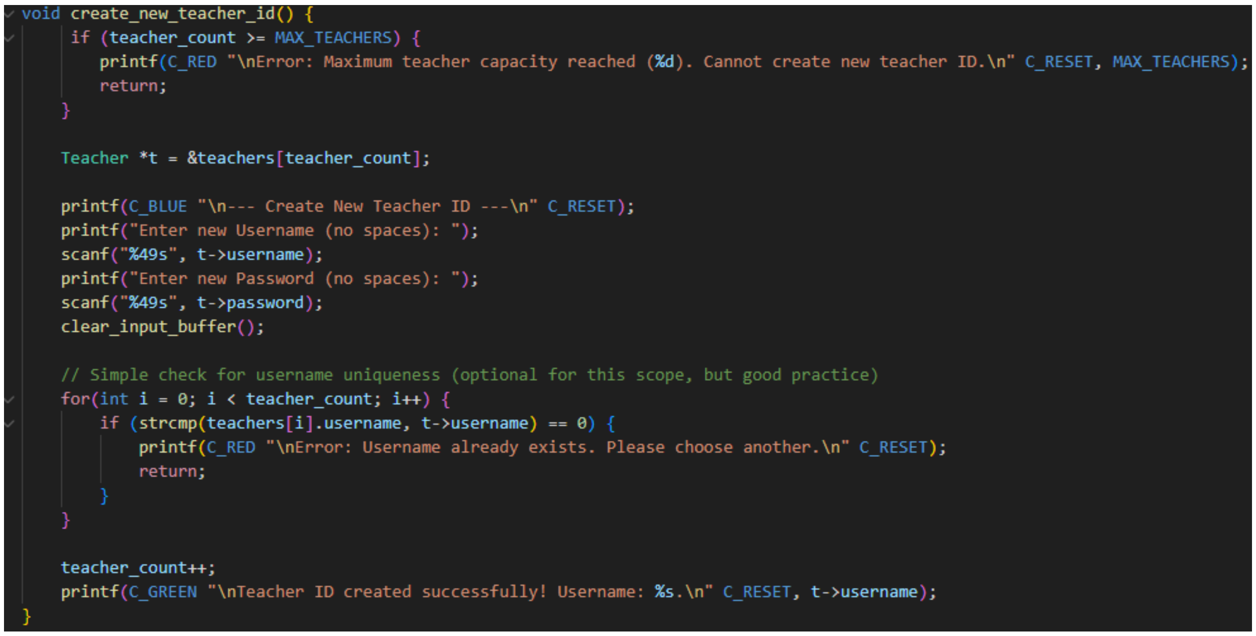
```c
    } while (choice != 0);
}
```



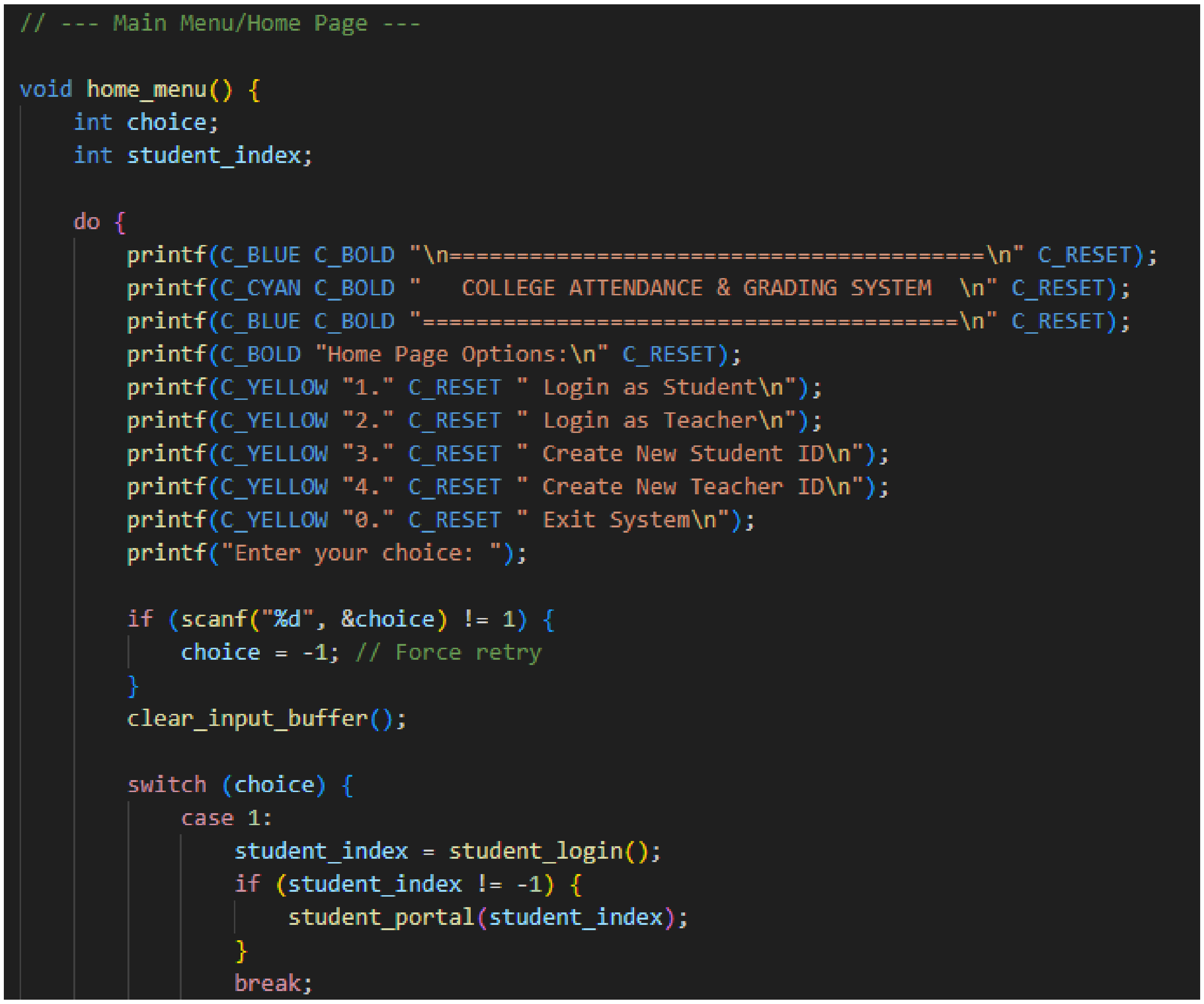
```c
// --- Creation Functions ---

void create_new_student_id() {
    if (student_count >= MAX_STUDENTS) {
        printf(C_RED "\nError: Maximum student capacity reached (%d). Cannot create new student ID.\n" C_RESET, MAX_STUDENTS);
        return;
    }

    Student *s = &students[student_count];

    printf(C_BLUE "\n--- Create New Student ID ---\n" C_RESET);

    // Input SAP ID
    while (true) {
        printf("Enter new 9-digit SAP ID: ");
        scanf("%10s", s->sap_id);
        clear_input_buffer();
        if (strlen(s->sap_id) == SAP_ID_LENGTH) {
            if (find_student_index(s->sap_id) == -1) {
                break;
            } else {
                printf(C_RED "Error: SAP ID already exists. Try again.\n" C_RESET);
            }
        } else {
            printf(C_RED "Error: SAP ID must be exactly %d digits.\n" C_RESET, SAP_ID_LENGTH);
        }
    }
```



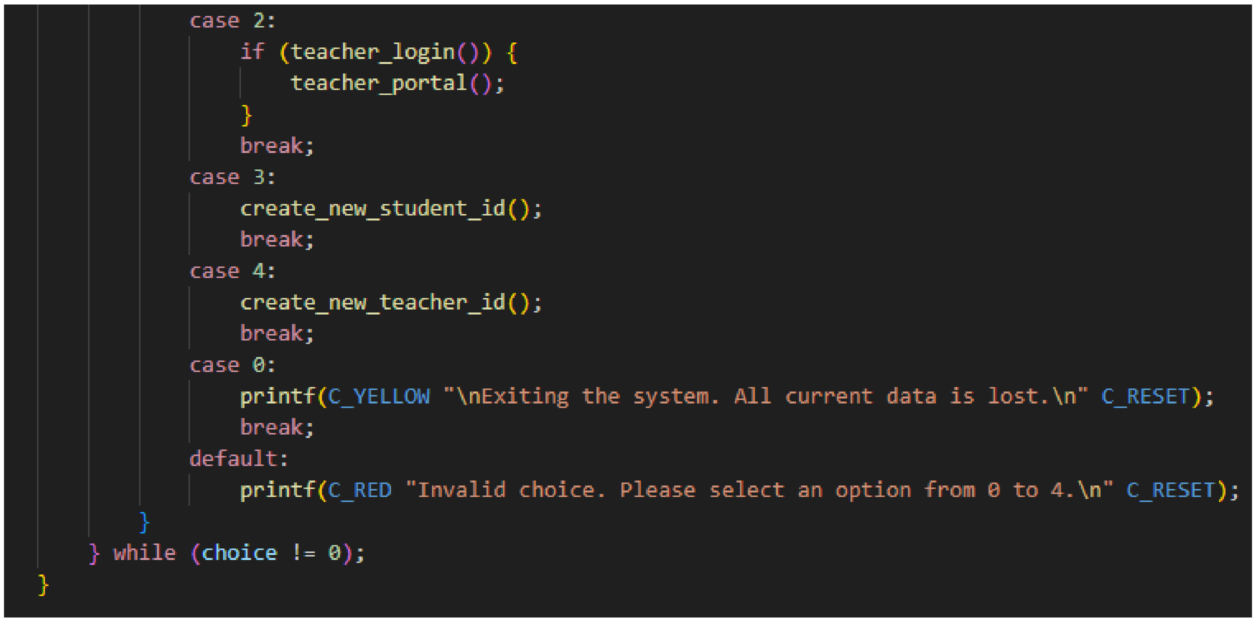
```c
    // Input Password
    printf("Enter Password (max 19 chars, no spaces): ");
    scanf("%19s", s->password);
    clear_input_buffer();

    // Input Name
    printf("Enter Full Name: ");
    if (fgets(s->name, 50, stdin)) {
        s->name[strcspn(s->name, "\n")] = 0; // Remove newline
    } else {
        strcpy(s->name, "New Student");
    }

    // Initialize Data
    s->attendance_maths = s->attendance_physics = s->attendance_coding = 0;
    s->marks_maths = s->marks_physics = s->marks_coding = 0;

    student_count++;
    printf(C_GREEN "\nStudent ID created successfully! Use SAP ID: %s to login.\n" C_RESET, s->sap_id);
}
```



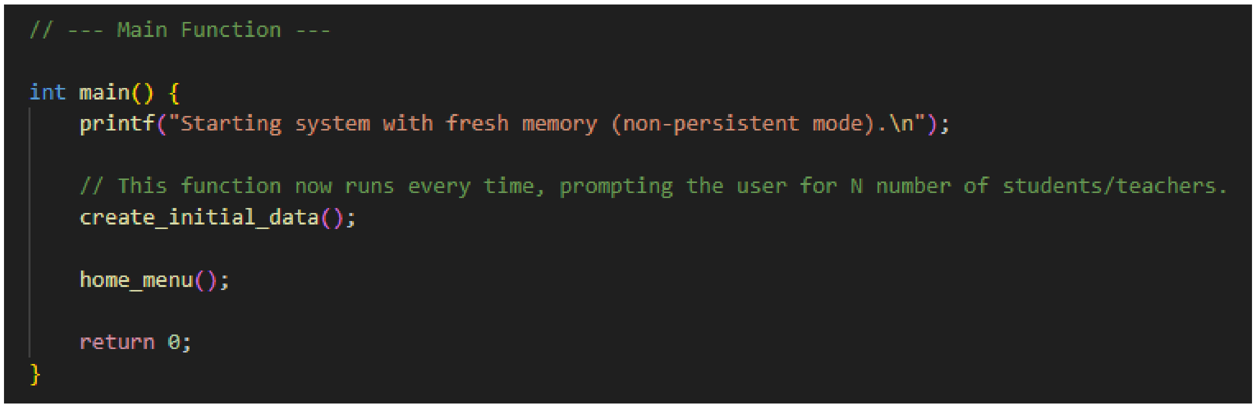
```c
void create_new_teacher_id() {
    if (teacher_count >= MAX_TEACHERS) {
        printf(C_RED "\nError: Maximum teacher capacity reached (%d). Cannot create new teacher ID.\n" C_RESET, MAX_TEACHERS);
        return;
    }

    Teacher *t = &teachers[teacher_count];

    printf(C_BLUE "\n--- Create New Teacher ID ---\n" C_RESET);
    printf("Enter new Username (no spaces): ");
    scanf("%49s", t->username);
    printf("Enter new Password (no spaces): ");
    scanf("%49s", t->password);
    clear_input_buffer();

    // Simple check for username uniqueness (optional for this scope, but good practice)
    for(int i = 0; i < teacher_count; i++) {
        if (strcmp(teachers[i].username, t->username) == 0) {
            printf(C_RED "\nError: Username already exists. Please choose another.\n" C_RESET);
            return;
        }
    }

    teacher_count++;
    printf(C_GREEN "\nTeacher ID created successfully! Username: %s.\n" C_RESET, t->username);
}
```



```c
// --- Main Menu/Home Page ---

void home_menu() {
    int choice;
    int student_index;

    do {
        printf(C_BLUE C_BOLD "\n========================================\n" C_RESET);
        printf(C_CYAN C_BOLD "   COLLEGE ATTENDANCE & GRADING SYSTEM  \n" C_RESET);
        printf(C_BLUE C_BOLD "========================================\n" C_RESET);
        printf(C_BOLD "Home Page Options:\n" C_RESET);
        printf(C_YELLOW "1." C_RESET " Login as Student\n");
        printf(C_YELLOW "2." C_RESET " Login as Teacher\n");
        printf(C_YELLOW "3." C_RESET " Create New Student ID\n");
        printf(C_YELLOW "4." C_RESET " Create New Teacher ID\n");
        printf(C_YELLOW "0." C_RESET " Exit System\n");
        printf("Enter your choice: ");

        if (scanf("%d", &choice) != 1) {
            choice = -1; // Force retry
        }
        clear_input_buffer();

        switch (choice) {
            case 1:
                student_index = student_login();
                if (student_index != -1) {
                    student_portal(student_index);
                }
                break;
```



```c
            case 2:
                if (teacher_login()) {
                    teacher_portal();
                }
                break;
            case 3:
                create_new_student_id();
                break;
            case 4:
                create_new_teacher_id();
                break;
            case 0:
                printf(C_YELLOW "\nExiting the system. All current data is lost.\n" C_RESET);
                break;
            default:
                printf(C_RED "Invalid choice. Please select an option from 0 to 4.\n" C_RESET);
        }
    } while (choice != 0);
}
```



```c
// --- Main Function ---

int main() {
    printf("Starting system with fresh memory (non-persistent mode).\n");

    // This function now runs every time, prompting the user for N number of students/teachers.
    create_initial_data();

    home_menu();

    return 0;
}
```

Header Files-

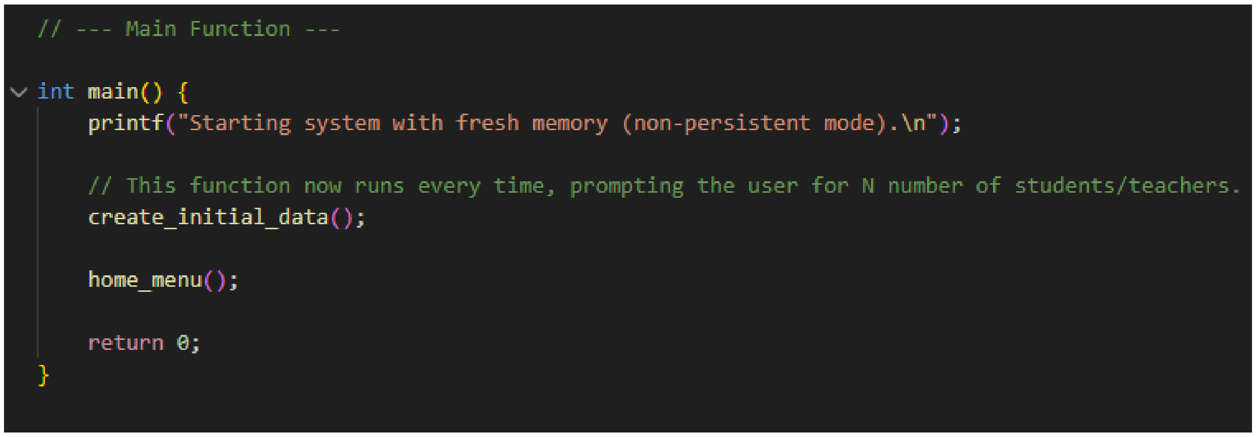
 stdio.h: Standard Input/Output library. Used for functions like printf(), scanf(), getchar(), and fgets().

 stdlib.h: Standard Library. Used for general utility functions (though specific functions like malloc or exit might not be directly visible in the final code, it's often included for general robustness in C programs).

 string.h: String handling library. Used for functions like strcmp() (string comparison), strlen() (string length), strcpy() (string copy), and strcspn() (for finding the length of the initial segment of a string which consists of characters not in a specified set, typically used here to remove the newline character from fgets input).

□ **stdbool.h: Standard Boolean library. Used to define and support the bool data type and the constants true and false.**

Main File-



```c
// --- Main Function ---

int main() {
    printf("Starting system with fresh memory (non-persistent mode).\n");

    // This function now runs every time, prompting the user for N number of students/teachers.
    create_initial_data();

    home_menu();

    return 0;
}
```

THANK YOU