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

//defining the maximum number of courses , student, faculty , grades
#define MAX_COURSES 100
#define MAX_STUDENTS 500
#define MAX_FACULTY 100
#define MAX_GRADES 100

//allocating a structure for course name , id , credits
typedef struct
{
   char name[50];
   char course_id[20];
   int credits;
} Course;

//allocating structure for student name , courses enrolled , and grades
typedef struct
{
   char name[50];
   char student_id[20];
   Course courses_enrolled[MAX_COURSES];
   int num_courses;
   float grades[MAX_COURSES]; // Store grades for courses
} Student;

// allocating structure for faculty
typedef struct
{
```

```
    char name[50];
    char faculty_id[20];
    Course courses_taught[MAX_COURSES];
    int num_courses;
} Faculty;

//for admin
typedef struct
{
    char name[50];
    char admin_id[20];
} Admin;

// Function Prototypes
void display_courses(Course courses[], int num_courses);
void enroll_course(Student *student, Course course);
void drop_course(Student *student, char course_id[]);
void view_schedule(Student student);
void manage_courses(Faculty *faculty);
void handle_grades(Faculty faculty);
void handle_attendance(Faculty faculty);
void admin_manage_students();
void admin_manage_faculty();
void admin_manage_courses();
void view_faculties_and_students(Student students[], int num_students, Faculty faculty[], int
num_faculty);

// Global Data
Course courses[MAX_COURSES] = {
    {"Math 101", "Math101", 3},
    {"Physics 202", "Physics202", 4},
```

```
    {"Chemistry 303", "Chemistry303", 3}};
int num_courses = 3;

Student students[MAX_STUDENTS];
int num_students = 0;

Faculty faculty[MAX_FACULTY];
int num_faculty = 0;

Admin admins[MAX_FACULTY];
int num_admins = 0;

int main()
{
    // Initialize test data
    strcpy(students[0].name, "Alice");
    strcpy(students[0].student_id, "S001");
    students[0].num_courses = 0;
    num_students++;

    strcpy(faculty[0].name, "Dr. Smith");
    strcpy(faculty[0].faculty_id, "F001");
    faculty[0].num_courses = 0;
    num_faculty++;

    strcpy(admins[0].name, "AdminUser ");
    strcpy(admins[0].admin_id, "A001");
    num_admins++;

    int choice;
```

```
while (1)
{
    printf("\n--- Course Registration System ---\n");
    printf("1. Student Login\n");
    printf("2. Faculty Login\n");
    printf("3. Admin Login\n");
    printf("4. Exit\n");
    printf("Enter Your Choice: ");
    scanf("%d", &choice);

    switch (choice)
    {
    case 1:
    {
        char student_id[20];
        printf("Enter Student ID: ");
        scanf("%s", student_id);

        // Find student
        int student_index = -1;
        for (int i = 0; i < num_students; i++)
        {
            if (strcmp(students[i].student_id, student_id) == 0)
            {
                student_index = i;
                break;
            }
        }

        if (student_index != -1)
        {
```

```
// Student Menu
while (1)
{
    printf("\n--- Student Menu ---\n");
    printf("1. View Schedule\n");
    printf("2. Enroll in Course\n");
    printf("3. Drop Course\n");
    printf("4. Logout\n");
    printf("Enter Your Choice: ");
    scanf("%d", &choice);

    switch (choice)
    {
    case 1:
        view_schedule(students[student_index]);
        break;
    case 2:
    {
        int course_index;
        display_courses(courses, num_courses);
        printf("Enter Course Index to Enroll: ");
        scanf("%d", &course_index);
        course_index--; // Adjust for 0-based index

        if (course_index >= 0 && course_index < num_courses)
        {
            enroll_course(&students[student_index], courses[course_index]);
        }
        else
        {
            printf("Invalid Course Index!\n");
```

```
            }
            break;
        }
        case 3:
        {
            char course_id[20];
            printf("Enter Course ID to Drop: ");
            scanf("%s", course_id);
            drop_course(&students[student_index], course_id);
            break;
        }
        case 4:
            goto exit_student;
        default:
            printf("Invalid Choice!\n");
        }
      }
   }
   else
   {
      printf("Invalid Student ID!\n");
   }
exit_student:;
   break;
}
case 2:
{
   char faculty_id[20];
   printf("Enter Faculty ID: ");
   scanf("%s", faculty_id);
```

```
// Find faculty
int faculty_index = -1;
for (int i = 0; i < num_faculty; i++)
{
   if (strcmp(faculty[i].faculty_id, faculty_id) == 0)
   {
      faculty_index = i;
      break;
   }
}

if (faculty_index != -1)
{
   // Faculty Menu
   while (1)
   {
      printf("\n--- Faculty Menu ---\n");
      printf("1. Manage Courses\n");
      printf("2. Handle Grades\n");
      printf("3. Handle Attendance\n");
      printf("4. Logout\n");
      printf("Enter Your Choice: ");
      scanf("%d", &choice);

      switch (choice)
      {
      case 1:
         manage_courses(&faculty[faculty_index]);
         break;
      case 2:
         handle_grades(faculty[faculty_index]);
```

```
            break;
        case 3:
            handle_attendance(faculty[faculty_index]);
            break;
        case 4:
            goto exit_faculty;
        default:
            printf("Invalid Choice!\n");
        }
      }
   }
   else
   {
      printf("Invalid Faculty ID!\n");
   }
exit_faculty:;
   break;
}
case 3:
{
   char admin_id[20];
   printf("Enter Admin ID: ");
   scanf("%s", admin_id);

   // Find admin
   int admin_index = -1;
   for (int i = 0; i < num_admins; i++)
   {
      if (strcmp(admins[i].admin_id, admin_id) == 0)
      {
         admin_index = i;
```

```
        break;
    }
}

if (admin_index != -1)
{
    while (1)
    {
        printf("\n--- Admin Menu ---\n");
        printf("1. Manage Students\n");
        printf("2. Manage Faculty\n");
        printf("3. Manage Courses\n");
        printf("4. View all faculties and students\n");
        printf("5. Logout\n");
        printf("Enter Your Choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            admin_manage_students();
            break;
        case 2:
            admin_manage_faculty();
            break;
        case 3:
            admin_manage_courses();
            break;
        case 4:
            view_faculties_and_students(students, num_students, faculty, num_faculty);
        case 5:
```

```
                goto exit_admin;
            default:
                printf("Invalid Choice!\n");
            }
        }
    }
    else
    {
        printf("Invalid Admin ID!\n");
    }
exit_admin:;
    break;
}
case 4:
    exit(0);
default:
    printf("Invalid Choice! Please try again.\n");
}
}
}

// Function Definitions
void display_courses(Course courses[], int num_courses)
{
    printf("\n--- Available Courses ---\n");
    for (int i = 0; i < num_courses; i++)
    {
        printf("%d. %s (%s) - %d credits\n", i + 1, courses[i].name, courses[i].course_id,
courses[i].credits);
    }
}
```

```
void enroll_course(Student *student, Course course)
{
    if (student->num_courses < MAX_COURSES)
    {
        student->courses_enrolled[student->num_courses++] = course;
        printf("Enrolled in %s successfully!\n", course.name);
    }
    else
    {
        printf("Cannot enroll, maximum courses reached!\n");
    }
}

void drop_course(Student *student, char course_id[])
{
    for (int i = 0; i < student->num_courses; i++)
    {
        if (strcmp(student->courses_enrolled[i].course_id, course_id) == 0)
        {
            for (int j = i; j < student->num_courses - 1; j++)
            {
                student->courses_enrolled[j] = student->courses_enrolled[j + 1];
            }
            student->num_courses--;
            printf("Dropped course %s successfully!\n", course_id);
            return;
        }
    }
    printf("Course not found in enrolled list!\n");
}
```

```
void view_schedule(Student student)
{
   printf("\n--- Your Schedule ---\n");
   for (int i = 0; i < student.num_courses; i++)
   {
      printf("%s (%s) - %d credits\n", student.courses_enrolled[i].name,
student.courses_enrolled[i].course_id, student.courses_enrolled[i].credits);
   }
}

void manage_courses(Faculty *faculty)
{
   printf("\n--- Manage Courses ---\n");
   int choice;
   printf("1. Add a Course\n2. Remove a Course\nEnter Your Choice: ");
   scanf("%d", &choice);

   if (choice == 1)
   {
      if (num_courses < MAX_COURSES)
      {
         char name[50], course_id[20];
         int credits;
         printf("Enter Course Name: ");
         scanf("%s", name);
         printf("Enter Course ID: ");
         scanf("%s", course_id);
         printf("Enter Credits: ");
         scanf("%d", &credits);
```

```
        strcpy(courses[num_courses].name, name);
        strcpy(courses[num_courses].course_id, course_id);
        courses[num_courses].credits = credits;
        num_courses++;

        printf("Course added successfully!\n");
    }
    else
    {
        printf("Cannot add more courses, limit reached!\n");
    }
}
else if (choice == 2)
{
    char course_id[20];
    printf("Enter Course ID to Remove: ");
    scanf("%s", course_id);

    int course_index = -1;
    for (int i = 0; i < num_courses; i++)
    {
        if (strcmp(courses[i].course_id, course_id) == 0)
        {
            course_index = i;
            break;
        }
    }

    if (course_index != -1)
    {
        for (int i = course_index; i < num_courses - 1; i++)
```

```
            {
                courses[i] = courses[i + 1];
            }
            num_courses--;
            printf("Course removed successfully!\n");
        }
        else
        {
            printf("Course not found!\n");
        }
    }
    else
    {
        printf("Invalid Choice!\n");
    }
}

void handle_grades(Faculty faculty)
{
    char student_id[20], course_id[20];
    float grade;

    printf("Enter Student ID: ");
    scanf("%s", student_id);
    printf("Enter Course ID: ");
    scanf("%s", course_id);
    printf("Enter Grade: ");
    scanf("%f", &grade);

    // Find student
    for (int i = 0; i < num_students; i++)
```

```
    {
        if (strcmp(students[i].student_id, student_id) == 0)
        {
            // Find course in student's schedule
            for (int j = 0; j < students[i].num_courses; j++)
            {
                if (strcmp(students[i].courses_enrolled[j].course_id, course_id) == 0)
                {
                    students[i].grades[j] = grade;
                    printf("Grade updated successfully!\n");
                    return;
                }
            }
        }
    }
    printf("Invalid Student ID or Course ID!\n");
}

void handle_attendance(Faculty faculty)
{
    printf("\n--- Handle Attendance ---\n");
    char student_id[20];
    printf("Enter Student ID to mark attendance: ");
    scanf("%s", student_id);

    int student_index = -1;
    for (int i = 0; i < num_students; i++)
    {
        if (strcmp(students[i].student_id, student_id) == 0)
        {
            student_index = i;
```

```
            break;
        }
    }

    if (student_index != -1)
    {
        printf("Attendance marked for %s.\n", students[student_index].name);
    }
    else
    {
        printf("Student not found!\n");
    }
}

void admin_manage_students()
{
    printf("\n--- Manage Students ---\n");
    int choice;
    printf("1. Add Student\n2. Remove Student\nEnter Your Choice: ");
    scanf("%d", &choice);

    if (choice == 1)
    {
        if (num_students < MAX_STUDENTS)
        {
            Student new_student;
            printf("Enter Student Name: ");
            scanf("%s", new_student.name);
            printf("Enter Student ID: ");
            scanf("%s", new_student.student_id);
            new_student.num_courses = 0;
```

```
            students[num_students++] = new_student;
            printf("Student added successfully!\n");
        }
        else
        {
            printf("Cannot add more students, maximum limit reached!\n");
        }
    }
    else if (choice == 2)
    {
        char student_id[20];
        printf("Enter Student ID to Remove: ");
        scanf("%s", student_id);

        int student_index = -1;
        for (int i = 0; i < num_students; i++)
        {
            if (strcmp(students[i].student_id, student_id) == 0)
            {
                student_index = i;
                break;
            }
        }

        if (student_index != -1)
        {
            for (int j = student_index; j < num_students - 1; j++)
            {
                students[j] = students[j + 1];
            }
            num_students--;
```

```
            printf("Student removed successfully!\n");
        }
        else
        {
            printf("Student not found!\n");
        }
    }
    else
    {
        printf("Invalid Choice!\n");
    }
}

void admin_manage_faculty()
{
    printf("\n--- Manage Faculty ---\n");
    int choice;
    printf("1. Add Faculty\n2. Remove Faculty\nEnter Your Choice: ");
    scanf("%d", &choice);

    if (choice == 1)
    {
        if (num_faculty < MAX_FACULTY)
        {
            char name[50], faculty_id[20];
            printf("Enter Faculty Name: ");
            scanf("%s", name);
            printf("Enter Faculty ID: ");
            scanf("%s", faculty_id);

            strcpy(faculty[num_faculty].name, name);
```

```
            strcpy(faculty[num_faculty].faculty_id, faculty_id);
            faculty[num_faculty].num_courses = 0;
            num_faculty++;

            printf("Faculty added successfully!\n");
        }
        else
        {
            printf("Cannot add more faculty, limit reached!\n");
        }
    }
    else if (choice == 2)
    {
        char faculty_id[20];
        printf("Enter Faculty ID to Remove: ");
        scanf("%s", faculty_id);

        int faculty_index = -1;
        for (int i = 0; i < num_faculty; i++)
        {
            if (strcmp(faculty[i].faculty_id, faculty_id) == 0)
            {
                faculty_index = i;
                break;
            }
        }

        if (faculty_index != -1)
        {
            for (int i = faculty_index; i < num_faculty - 1; i++)
            {
```

```
            faculty[i] = faculty[i + 1];
        }
        num_faculty--;
        printf("Faculty removed successfully!\n");
    }
    else
    {
        printf("Faculty not found!\n");
    }
  }
  else
  {
    printf("Invalid Choice!\n");
  }
}

void admin_manage_courses()
{
  printf("\n--- Manage Courses ---\n");
  int choice;
  printf("1. Add Course\n2. Remove Course\nEnter Your Choice: ");
  scanf("%d", &choice);

  if (choice == 1)
  {
    if (num_courses < MAX_COURSES)
    {
        Course new_course;
        printf("Enter Course Name: ");
        scanf("%s", new_course.name);
        printf("Enter Course ID: ");
```

```
        scanf("%s", new_course.course_id);
        printf("Enter Course Credits: ");
        scanf("%d", &new_course.credits);
        courses[num_courses++] = new_course;
        printf("Course added successfully!\n");
    }
    else
    {
        printf("Cannot add more courses, maximum limit reached!\n");
    }
}
else if (choice == 2)
{
    char course_id[20];
    printf("Enter Course ID to Remove: ");
    scanf("%s", course_id);

    int course_index = -1;
    for (int i = 0; i < num_courses; i++)
    {
        if (strcmp(courses[i].course_id, course_id) == 0)
        {
            course_index = i;
            break;
        }
    }

    if (course_index != -1)
    {
        for (int j = course_index; j < num_courses - 1; j++)
        {
```

```
            courses[j] = courses[j + 1];
        }
        num_courses--;
        printf("Course removed successfully!\n");
    }
    else
    {
        printf("Course not found!\n");
    }
  }
  else
  {
    printf("Invalid Choice!\n");
  }
}

void view_faculties_and_students(Student students[], int num_students, Faculty faculty[], int num_faculty)
{
  printf("\n--- List of Faculties ---\n");
  for (int i = 0; i < num_faculty; i++)
  {
    printf("Faculty Name: %s, Faculty ID: %s\n", faculty[i].name, faculty[i].faculty_id);
  }

  printf("\n--- List of Students ---\n");
  for (int i = 0; i < num_students; i++)
  {
    printf("Student Name: %s, Student ID: %s\n", students[i].name, students[i].student_id);
  }
```