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

#define MAX_COURSES 5
#define MAX_STUDENTS 50
#define MAX_FACULTY 5

// Structure to store course details
struct Course {
int course_id;
char course_name[50];
int capacity;
int registered;
int faculty_id; // ID of the faculty who manages this course
};

// Structure to store student details
struct Student {
int student_id;
char student_name[50];
int registered_courses[MAX_COURSES]; // Array of course IDs the student is registered in
};

// Structure to store faculty details
struct Faculty {
int faculty_id;
char faculty_name[50];
};

// Function to display available courses for students
void display_courses(struct Course courses[], int num_courses) {
printf("\nAvailable Courses:\n");
for (int i = 0; i < num_courses; i++) {
printf("Course ID: %d, Course Name: %s, Capacity: %d, Registered: %d\n",
courses[i].course_id, courses[i].course_name, courses[i].capacity, courses[i].registered);
}
}

// Function for students to register for courses
void register_course(struct Student *student, struct Course courses[], int num_courses) {
int course_id;
printf("\nEnter the Course ID you want to register for: ");
scanf("%d", &course_id);

// Validate the course ID
if (course_id < 1 || course_id > num_courses) {
printf("Invalid Course ID.\n");
return;
}

// Check if the student is already registered for the course
```

```
for (int i = 0; i < MAX_COURSES; i++) {
if (student->registered_courses[i] == course_id) {
printf("You are already registered for this course.\n");
return;
}
}

// Check if the course has available spots
if (courses[course_id - 1].registered >= courses[course_id - 1].capacity) {
printf("Sorry, this course is full.\n");
return;
}

// Register the student for the course
for (int i = 0; i < MAX_COURSES; i++) {
if (student->registered_courses[i] == 0) {
student->registered_courses[i] = course_id;
courses[course_id - 1].registered++;
printf("You have successfully registered for the course: %s\n", courses[course_id -
1].course_name);
return;
}
}

printf("You cannot register for more courses.\n");
}

// Function to view the courses a student is registered for
void view_registered_courses(struct Student student, struct Course courses[], int
num_courses) {
printf("\nCourses you are registered for:\n");
for (int i = 0; i < MAX_COURSES; i++) {
if (student.registered_courses[i] != 0) {
printf("Course Name: %s\n", courses[student.registered_courses[i] - 1].course_name);
}
}
}

// Function for faculty to add new courses
void add_course(struct Faculty faculty, struct Course courses[], int *num_courses) {
if (*num_courses >= MAX_COURSES) {
printf("Cannot add more courses. Maximum limit reached.\n");
return;
}

printf("\nEnter Course Name: ");
getchar(); // To clear the newline character
fgets(courses[*num_courses].course_name, 50, stdin);
courses[*num_courses].course_name[strcspn(courses[*num_courses].course_name, "\n")]
= 0; // Remove newline
courses[*num_courses].course_id = *num_courses + 1;
```

```
printf("Enter Capacity for Course: ");
scanf("%d", &courses[*num_courses].capacity);
courses[*num_courses].registered = 0;
courses[*num_courses].faculty_id = faculty.faculty_id;

(*num_courses)++;
printf("Course successfully added.\n");
}

// Function for faculty to view and manage courses
void manage_courses(struct Faculty faculty, struct Course courses[], int num_courses) {
printf("\nCourses managed by you (Faculty: %s):\n", faculty.faculty_name);
for (int i = 0; i < num_courses; i++) {
if (courses[i].faculty_id == faculty.faculty_id) {
printf("Course ID: %d, Name: %s, Capacity: %d, Registered: %d\n",
courses[i].course_id, courses[i].course_name, courses[i].capacity, courses[i].registered);
}
}
}

// Main function
int main() {
// Initialize faculty members
struct Faculty faculty[MAX_FACULTY] = {
{1, "Dr. Smith"},
{2, "Dr. Johnson"},
{3, "Dr. Brown"}
};

// Initialize courses
struct Course courses[MAX_COURSES] = {
{1, "Computer Science", 10, 0, 1},
{2, "Mathematics", 10, 0, 2},
{3, "Physics", 10, 0, 2},
{4, "Chemistry", 10, 0, 3},
{5, "Biology", 10, 0, 3}
};

struct Student students[MAX_STUDENTS] = {
{1, "John Doe", {0}},
{2, "Jane Smith", {0}},
{3, "Alice Johnson", {0}},
{4, "Bob Brown", {0}},
{5, "Charlie Davis", {0}}
};

int student_id, faculty_id, choice;
printf("Enter your role (1 for Student, 2 for Faculty): ");
scanf("%d", &choice);

if (choice == 1) {
```

```
printf("Enter Student ID: ");
scanf("%d", &student_id);
if (student_id < 1 || student_id > MAX_STUDENTS) {
printf("Invalid Student ID.\n");
return 1;
}

struct Student *current_student = &students[student_id - 1];
while (1) {
printf("\nStudent Menu:\n");
printf("1. View Available Courses\n");
printf("2. Register for a Course\n");
printf("3. View Registered Courses\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
case 1:
display_courses(courses, MAX_COURSES);
break;
case 2:
register_course(current_student, courses, MAX_COURSES);
break;
case 3:
view_registered_courses(*current_student, courses, MAX_COURSES);
break;
case 4:
printf("Exiting the system. Goodbye!\n");
return 0;
default:
printf("Invalid choice. Please try again.\n");
}
}
} else if (choice == 2) {
printf("Enter Faculty ID: ");
scanf("%d", &faculty_id);
if (faculty_id < 1 || faculty_id > MAX_FACULTY) {
printf("Invalid Faculty ID.\n");
return 1;
}

struct Faculty *current_faculty = &faculty[faculty_id - 1];
while (1) {
printf("\nFaculty Menu (Faculty: %s):\n", current_faculty->faculty_name);
printf("1. Add a Course\n");
printf("2. View and Manage Courses\n");
printf("3. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
```

```
switch (choice) {
case 1:
add_course(*current_faculty, courses, MAX_COURSES);
break;
case 2:
manage_courses(*current_faculty, courses, MAX_COURSES);
break;
case 3:
printf("Exiting the system. Goodbye!\n");
return 0;
default:
printf("Invalid choice. Please try again.\n");
}
}
} else {
printf("Invalid choice. Exiting...\n");
}

return 0;
}
```