1. #include <stdio.h>

```
#include <stdlib.h>
#include <string.h>
#define MAX_STUDENTS 100
#define MAX_COURSES 4
typedef struct {
int rollno;
char name[50];
char dept[50];
char courses[MAX_COURSES][50];
int credits[MAX_COURSES];
int grades[MAX_COURSES];
int num_courses;
float gpa;
} Student;
Student students[MAX_STUDENTS];
int student_count = 0;
int grade_to_points(char grade) {
switch (grade) {
case 'S': return 10;
case 'A': return 9;
case 'B': return 8;
case 'C': return 7;
case 'D': return 6;
case 'E': return 5;
case 'F': return 0;
default: return -1;
}
}
```

```
void calculate_gpa(Student *s) {
int total_credits = 0;
int total_points = 0;
for (int i = 0; i < s->num_courses; i++) {
total_credits += s->credits[i];
total_points += s->credits[i] * s->grades[i];
}
if (total_credits != 0)
s->gpa = (float)total_points / total_credits;
else
s->gpa = 0;
}
void insert_student() {
if (student_count >= MAX_STUDENTS) {
printf("Student limit reached.\n");
return;
}
Student s;
printf("Enter roll no: ");
scanf("%d", &s.rollno);
printf("Enter name: ");
scanf("%s", s.name);
printf("Enter department: ");
scanf("%s", s.dept);
printf("Enter number of courses (3 or 4): ");
scanf("%d", &s.num_courses);
for (int i = 0; i < s.num_courses; i++) {
printf("Enter course %d name: ", i + 1);
scanf("%s", s.courses[i]);
```

```
printf("Enter course %d credits: ", i + 1);
scanf("%d", &s.credits[i]);
printf("Enter course %d grade (S/A/B/C/D/E/F): ", i + 1);
char grade;
scanf(" %c", &grade);
s.grades[i] = grade_to_points(grade);
}
calculate_gpa(&s);
students[student_count++] = s;
}
void create_gpa_column() {
for (int i = 0; i < student_count; i++) {
calculate_gpa(&students[i]);
}
printf("GPA column created for all students.\n");
}
void delete_course(int rollno) {
for (int i = 0; i < student_count; i++) {
if (students[i].rollno == rollno) {
if (students[i].num_courses == 4) {
students[i].num_courses--;
calculate_gpa(&students[i]);
printf("One course deregistered for student %d.\n", rollno);
} else {
printf("Student does not have 4 courses.\n");
}
return;
}
}
```

```
printf("Student not found.\n");
}
void insert_course(int rollno) {
for (int i = 0; i < student_count; i++) {
if (students[i].rollno == rollno) {
if (students[i].num_courses == 3) {
printf("Enter new course name: ");
scanf("%s", students[i].courses[students[i].num_courses]);
printf("Enter new course credits: ");
scanf("%d", &students[i].credits[students[i].num_courses]);
printf("Enter new course grade (S/A/B/C/D/E/F): ");
char grade;
scanf(" %c", &grade);
students[i].grades[students[i].num_courses] =
grade_to_points(grade);
students[i].num_courses++;
calculate_gpa(&students[i]);
printf("New course added for student %d.\n", rollno);
} else {
printf("Student does not have 3 courses.\n");
}
return;
}
}
printf("Student not found.\n");
}
void update_course_name(int rollno, int course_index, const char
*new_course_name) {
for (int i = 0; i < student_count; i++) {
```

```
if (students[i].rollno == rollno) {
if (course_index < students[i].num_courses) {
strcpy(students[i].courses[course_index], new_course_name);
printf("Course name updated for student %d.\n", rollno);
} else {
printf("Invalid course index.\n");
}
return;
}
}
printf("Student not found.\n");
}
void upgrade_grade_point(int rollno) {
for (int i = 0; i < student_count; i++) {
if (students[i].rollno == rollno) {
for (int j = 0; j < students[i].num_courses; j++) {
if (students[i].grades[j] == 7) {
students[i].grades[j] = 8;
}
}
calculate_gpa(&students[i]);
printf("Grades updated for student %d.\n", rollno);
return;
}
}
printf("Student not found.\n");
}

void print_grade_report(int rollno) {
```

```
for (int i = 0; i < student_count; i++) {
if (students[i].rollno == rollno) {
printf("Grade report for %s (Roll No: %d):\n",
students[i].name, students[i].rollno);
for (int j = 0; j < students[i].num_courses; j++) {
printf("Course: %s, Credits: %d, Grade: %d\n",
students[i].courses[j], students[i].credits[j], students[i].grades[j]);
}
printf("GPA: %.2f\n", students[i].gpa);
return;
}
}
printf("Student not found.\n");
}
void menu() {
int choice, rollno, course_index;
char new_course_name[50];
while (1) {
printf("\nMenu:\n");
printf("1. Insert student record\n");
printf("2. Create GPA column for all students\n");
printf("3. Deregister a course for a student\n");
printf("4. Register a new course for a student\n");
printf("5. Update course name for a student\n");
printf("6. Upgrade grade point for a student\n");
printf("7. Print grade report for a student\n");
printf("8. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
```

```
switch (choice) {
case 1:
insert_student();
break;
case 2:
create_gpa_column();
break;
case 3:
printf("Enter roll no of the student: ");
scanf("%d", &rollno);
delete_course(rollno);
break;
case 4:
printf("Enter roll no of the student: ");
scanf("%d", &rollno);
insert_course(rollno);
break;
case 5:
printf("Enter roll no of the student: ");
scanf("%d", &rollno);
printf("Enter course index (0 to 3): ");
scanf("%d", &course_index);
printf("Enter new course name: ");
scanf("%s", new_course_name);
update_course_name(rollno, course_index, new_course_name);
break;
case 6:
printf("Enter roll no of the student: ");
```

```
scanf("%d", &rollno);
upgrade_grade_point(rollno);
break;
case 7:
printf("Enter roll no of the student: ");
scanf("%d", &rollno);
print_grade_report(rollno);
break;
case 8:
exit(0);
default:
printf("Invalid choice.\n");
}
}
}
int main() {
menu();
return 0;
}
```

```
2. -- Create Student table
CREATE TABLE Student (
Std_rollno INT PRIMARY KEY,
Std_name VARCHAR(50),
Dept VARCHAR(50),
Course1 CHAR(50),
Course2 CHAR(50),
Course3 CHAR(50),
Course4 CHAR(50)
);

-- Insert 5 student records
INSERT INTO Student (Std_rollno, Std_name, Dept, Course1, Course2, Course3,
Course4) VALUES
(1, 'John Doe', 'CSE', 'DBMS', 'OS', 'DS', 'Maths'),
(2, 'Jane Smith', 'ECE', 'Signals', 'Networks', 'Electronics', 'Maths'),
(3, 'Alice Johnson', 'MECH', 'Thermo', 'Fluid', 'Mechanics', 'Maths'),
(4, 'Bob Brown', 'CIVIL', 'Structures', 'Geo', 'Hydraulics', 'Maths'),
(5, 'Charlie Davis', 'EEE', 'Circuits', 'EMF', 'Power', 'Maths');

-- Drop Course2 and Course3 columns
ALTER TABLE Student
DROP COLUMN Course2;

ALTER TABLE Student
DROP COLUMN Course3;

-- Add DoB and email columns
ALTER TABLE Student
```

```
ADD DoB DATE NOT NULL,
ADD email VARCHAR(50) CONSTRAINT email_format CHECK (email LIKE '%@nitt.edu
');

-- Change Course1 datatype to VARCHAR2
ALTER TABLE Student
MODIFY Course1 VARCHAR2(50);

-- Rename Std_rollno to Std_rno
ALTER TABLE Student
RENAME COLUMN Std_rollno TO Std_rno;

-- Update Course1 from 'DBMS' to 'OS'
UPDATE Student
SET Course1 = 'OS'
WHERE Course1 = 'DBMS';

-- Delete students with name starting with 'S'
DELETE FROM Student
WHERE Std_name LIKE 'S%';

-- Select students born after 2005
SELECT * FROM Student
WHERE DoB > '2005-12-31';

-- Simulate DROP TABLE
DROP TABLE Student;

-- Recreate the table to simulate TRUNCATE
```

```
CREATE TABLE Student (
Std_rno INT PRIMARY KEY,
Std_name VARCHAR(50),
Dept VARCHAR(50),
Course1 VARCHAR2(50),
Course4 CHAR(50),
DoB DATE NOT NULL,
email VARCHAR(50) CONSTRAINT email_format CHECK (email LIKE '%@nitt.edu
')
);

-- Simulate TRUNCATE
TRUNCATE TABLE Student;
```