Week 2

Q1

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

#define MAX_STUDENTS 100
#define MAX_NAME_LENGTH 50
#define MAX_COURSES 4
#define FILENAME "students.txt"

typedef struct {
    char rollno[15];
    char name[MAX_NAME_LENGTH];
    char dept[10];
    char courses[MAX_COURSES][10];
    int credits[MAX_COURSES];
    int grades[MAX_COURSES];
    float gpa;
} Student;

Student students[MAX_STUDENTS];
int student_count = 0;

void load_students_from_file() {
    FILE *file = fopen(FILENAME, "r");
    if (file == NULL) {
        printf("File not found. Creating a new file...\n");
        file = fopen(FILENAME, "w");
        if (file == NULL) {
            perror("Error creating file");
            exit(EXIT_FAILURE);
        }
        fclose(file);
        return;
    }

    while (fscanf(file, "%s %s %s", students[student_count].rollno, students[student_count].name, students[student_count].de
        for (int i = 0; i < MAX_COURSES; i++) {
            fscanf(file, "%s %d %d", students[student_count].courses[i], &students[student_count].credits[i], &students[stud
        }
        fscanf(file, "%f", &students[student_count].gpa);
        student_count++;
    }
    fclose(file);
}

void save_students_to_file() {
    FILE *file = fopen(FILENAME, "w");
    if (file == NULL) {
        perror("Error opening file for writing");
        exit(EXIT_FAILURE);
    }
    for (int i = 0; i < student_count; i++) {
        fprintf(file, "%s %s %s ", students[i].rollno, students[i].name, students[i].dept);
        for (int j = 0; j < MAX_COURSES; j++) {
            fprintf(file, "%s %d %d ", students[i].courses[j], students[i].credits[j], students[i].grades[j]);
        }
        fprintf(file, "%.2f\n", students[i].gpa);
    }
    fclose(file);
}

void insert_student() {
    if (student_count >= MAX_STUDENTS) {
        printf("Max student limit reached.\n");
        return;
    }

    Student s;
    printf("Enter roll number: ");
    scanf("%s", s.rollno);
```

```
    printf("Enter name: ");
    scanf("%s", s.name);
    printf("Enter department: ");
    scanf("%s", s.dept);

    for (int i = 0; i < MAX_COURSES; i++) {
        printf("Enter course %d name (or 'none' to skip): ", i + 1);
        scanf("%s", s.courses[i]);
        if (strcmp(s.courses[i], "none") == 0) {
            s.courses[i][0] = '\0';
            s.credits[i] = 0;
            s.grades[i] = 0;
        } else {
            printf("Enter course %d credits: ", i + 1);
            scanf("%d", &s.credits[i]);
            printf("Enter course %d grade: ", i + 1);
            scanf("%d", &s.grades[i]);
        }
    }

    students[student_count++] = s;
    save_students_to_file();
}

void create_gpa_column() {
    for (int i = 0; i < student_count; i++) {
        float total_points = 0;
        int total_credits = 0;
        for (int j = 0; j < MAX_COURSES && students[i].courses[j][0] != '\0'; j++) {
            total_points += students[i].grades[j] * students[i].credits[j];
            total_credits += students[i].credits[j];
        }
        students[i].gpa = total_points / total_credits;
    }
    save_students_to_file();
}

void delete_course(char *rollno, char *course_name) {
    for (int i = 0; i < student_count; i++) {
        if (strcmp(students[i].rollno, rollno) == 0) {
            for (int j = 0; j < MAX_COURSES; j++) {
                if (strcmp(students[i].courses[j], course_name) == 0) {
                    students[i].courses[j][0] = '\0';
                    students[i].credits[j] = 0;
                    students[i].grades[j] = 0;
                    break;
                }
            }
        }
    }
    save_students_to_file();
}

void insert_course(char *rollno, char *course_name, int credits, int grade) {
    for (int i = 0; i < student_count; i++) {
        if (strcmp(students[i].rollno, rollno) == 0) {
            for (int j = 0; j < MAX_COURSES; j++) {
                if (students[i].courses[j][0] == '\0') {
                    strcpy(students[i].courses[j], course_name);
                    students[i].credits[j] = credits;
                    students[i].grades[j] = grade;
                    break;
                }
            }
        }
    }
    save_students_to_file();
}

void update_course_name(char *rollno, char *old_name, char *new_name) {
    for (int i = 0; i < student_count; i++) {
        if (strcmp(students[i].rollno, rollno) == 0) {
            for (int j = 0; j < MAX_COURSES; j++) {
                if (strcmp(students[i].courses[j], old_name) == 0) {
                    strcpy(students[i].courses[j], new_name);
                    break;
                }
```

```
            }
        }
    }
    save_students_to_file();
}

void upgrade_grade(char *rollno, char *course_name) {
    for (int i = 0; i < student_count; i++) {
        if (strcmp(students[i].rollno, rollno) == 0) {
            for (int j = 0; j < MAX_COURSES; j++) {
                if (strcmp(students[i].courses[j], course_name) == 0 && students[i].grades[j] == 7) {
                    students[i].grades[j] = 8;
                    break;
                }
            }
        }
    }
    save_students_to_file();
}

void print_grade_report(char *rollno) {
    for (int i = 0; i < student_count; i++) {
        if (strcmp(students[i].rollno, rollno) == 0) {
            printf("Roll No: %s\n", students[i].rollno);
            printf("Name: %s\n", students[i].name);
            printf("Department: %s\n", students[i].dept);
            printf("Courses:\n");
            for (int j = 0; j < MAX_COURSES; j++) {
                if (students[i].courses[j][0] != '\0') {
                    printf("%s (Credits: %d, Grade: %d)\n", students[i].courses[j], students[i].credits[j], students[i].grad
                }
            }
            printf("GPA: %.2f\n", students[i].gpa);
            break;
        }
    }
}

int main() {
    load_students_from_file();

    while (1) {
        int choice;
        printf("1. Insert Student\n2. Create GPA Column\n3. Delete Course\n4. Insert Course\n5. Update Course Name\n6.
        Upgrade Grade\n7. Print Grade Report\n8. Exit\nEnter choice: ");
        scanf("%d", &choice);

        char rollno[15], course_name[10], new_name[10];
        int credits, grade;

        switch (choice) {
            case 1:
                insert_student();
                break;
            case 2:
                create_gpa_column();
                break;
            case 3:
                printf("Enter roll number: ");
                scanf("%s", rollno);
                printf("Enter course name: ");
                scanf("%s", course_name);
                delete_course(rollno, course_name);
                break;
            case 4:
                printf("Enter roll number: ");
                scanf("%s", rollno);
                printf("Enter course name: ");
                scanf("%s", course_name);
                printf("Enter credits: ");
                scanf("%d", &credits);
                printf("Enter grade: ");
                scanf("%d", &grade);
                insert_course(rollno, course_name, credits, grade);
                break;
            case 5:
                printf("Enter roll number: ");
```

```
                scanf("%s", rollno);
                printf("Enter old course name: ");
                scanf("%s", course_name);
                printf("Enter new course name: ");
                scanf("%s", new_name);
                update_course_name(rollno, course_name, new_name);
                break;
            case 6:
                printf("Enter roll number: ");
                scanf("%s", rollno);
                printf("Enter course name: ");
                scanf("%s", course_name);
                upgrade_grade(rollno, course_name);
                break;
            case 7:
                printf("Enter roll number: ");
                scanf("%s", rollno);
                print_grade_report(rollno);
                break;
            case 8:
                exit(0);
            default:
                printf("Invalid choice.\n");
        }
    }

    return 0;
}
```

Q2

```
-- Create Database
CREATE DATABASE StudentDB;

-- Use the newly created database
USE StudentDB;

-- Create the Student table with the specified schema
CREATE TABLE Student (
    Std_rollno VARCHAR(15) PRIMARY KEY,
    Std_name VARCHAR(50),
    Dept VARCHAR(10),
    Course1 VARCHAR(10),
    Course2 VARCHAR(10),
    Course3 VARCHAR(10),
    Course4 VARCHAR(10)
);

-- Insert at least 5 student records into the Student table
INSERT INTO Student (Std_rollno, Std_name, Dept, Course1, Course2, Course3, Course4) VALUES
('1', 'Alice', 'CSE', 'DBMS', 'OS', 'DS', NULL),
('2', 'Bob', 'ECE', 'DBMS', 'OS', 'DS', 'COA'),
('3', 'Charlie', 'EEE', 'DBMS', 'OS', NULL, NULL),
('4', 'David', 'CSE', 'DBMS', 'OS', 'DS', NULL),
('5', 'Eve', 'CSE', 'DBMS', 'OS', 'DS', 'COA');

-- Delete Course2 and Course3 attributes from the Student table
ALTER TABLE Student DROP COLUMN Course2;
ALTER TABLE Student DROP COLUMN Course3;

-- Insert two new columns DoB and email into the Student table
ALTER TABLE Student ADD DoB DATE NOT NULL;
ALTER TABLE Student ADD email VARCHAR(50) NOT NULL CHECK (email LIKE '%@nitt.edu');

-- Change Course1 datatype to VARCHAR(10)
ALTER TABLE Student MODIFY Course1 VARCHAR(10);

-- Update the column name ‘Std_rollno’ to ‘Std_rno’
ALTER TABLE Student RENAME COLUMN Std_rollno TO Std_rno;

-- Update all student records who pursue a course named “DBMS” to “OS”
UPDATE Student SET Course1 = 'OS' WHERE Course1 = 'DBMS';

-- Delete a student record with student name starting with letter ‘S’
DELETE FROM Student WHERE Std_name LIKE 'S%';
```

```
-- Display all records in which a student has born after the year 2005
SELECT * FROM Student WHERE YEAR(DoB) > 2005;

-- Simulate DROP and TRUNCATE commands
DROP TABLE Student;
TRUNCATE TABLE Student;
```