Project Report: Student Report Card Management System

1. Title Page

- **Title:** A basic Student Report Card System (Built with C Structures)
- **Submitted By:** Dhruv (SAP ID: 590024139)
- **Submission Date:** 24 November 2025
- **Course/Subject:** Programming in C (Course Code: CSEG1041)

2. Abstract

This report explains how we built a **Student Report Card Management System** using the **C programming language**. The system uses a special data structure called a 'struct' (struct studInfo) and an array to keep track of student details. For each student, it stores their ID, name, five subject scores, the calculated average score, and the final letter grade. The program gives the user a simple menu to do four main things: **add new students (Create)**, **view everyone (Read)**, **change information (Update)**, and **remove students (Delete)**. The main goal of this project was to show that we understand how to use C's main tools, like data structures, arrays, and functions, to manage data while the program is running.

3. Problem Definition

Managing student scores, calculating averages, and assigning grades by hand is slow and easy to mess up.

- **Goal:** To create a simple, working program in C that can easily store and manage student records digitally.
- **Key Requirements:**
    - Store important information for each student (ID, name, and 5 scores).
    - The program must **automatically** calculate the average score and give out the correct letter grade.
    - Give users options to **add**, **show**, **search for**, **change**, and **delete** student data using a basic menu.
    - Be able to hold up to 200 records (MAX_RECORDS = 200).

4. System Design (How it's Built)

The program is set up with a main menu that lets the user choose what to do. All the student data is stored in one large array while the program is running.

# Main Program Flow

The program runs in a loop, always showing the menu until the user decides to quit.

# Data Structure Design

We use a C struct to group all the related pieces of information for one student:

```
struct studInfo {
  int idNumber;
  char studentName[50];
  float scores[5];
  float avgPercent;
  char grade;
};
```

All the student records are held in a global array called records[MAX_RECORDS].

Grade Calculation Rule

The letter grade is set based on the student's average score, using simple 'if-else' rules:

# Average Percentage	Grade
>= 90	**A**
>= 80 (but < 90)	**B**
>= 70 (but < 80)	**C**
>= 60 (but < 70)	**D**
<60	**F**

# Deleting a Record

The remove_student() function handles deletion. When a student is deleted from the middle of the array, we have to make sure there are no empty spots. We do this by **shifting** all the records that came after the deleted one forward by one position.

5. Implementation Details (Code Pieces)

The system is broken down into separate functions for each job (like adding, showing, and deleting).

# Main Program Control

The main() function keeps the program running and calls the correct function based on the user's choice.

```
// Snippet from main()
int choice_of_user;
do {
  // ... menu display ...
  scanf("%d", &choice_of_user);

  switch(choice_of_user) {
    case 1: insertNEWstudent(); break;
    case 2: showALLstudents(); break;
    // ...
  }
} while(choice_of_user != 6);
```

# Adding a New Student (insertNEWstudent())

This function takes input, calculates the average, sets the grade, and puts the new data into the next empty spot in the records array.

```
// Score Calculation and Storage Snippet

// ... score input loop ...

newStudent.avgPercent = sumOfScores / 5.0;


// ... grade assignment logic ...


records[student_count++] = newStudent;
```

# Deleting a Student (remove_student())

This part of the code shows the shifting process:

```
// Deletion/Shifting Snippet
for(int i = 0; i < student_count; i++) {
  if(records[i].idNumber == delId) {
    // Shift records left to cover the deleted one
    for(int k = i; k < student_count - 1; k++) {
      records[k] = records[k + 1];
    }
    student_count--;
    printf("Student record removed.\n");
    break;
  }
}
```

6. Testing & Results

We tested the program by trying out all the menu options and making sure the grade calculations were correct, especially near the pass/fail lines.

Test Case	What We Did	Test Data	What We Expected	Result
1	Add Student	Scores that average exactly 80.0	Grade 'B' (since 80 is the boundary)	**It Worked**

2	Search Student	An ID that wasn't in the list (e.g., 999)	The message "No student found with this ID."	**It Worked**
3	Update Student	Changed the scores for an existing student.	The record updated, and the average/grade was fixed correctly.	**It Worked**
4	Delete	Removed the first student record.	All other records moved up in the array, and the count dropped.	**It Worked**

7. Conclusion & Future Work

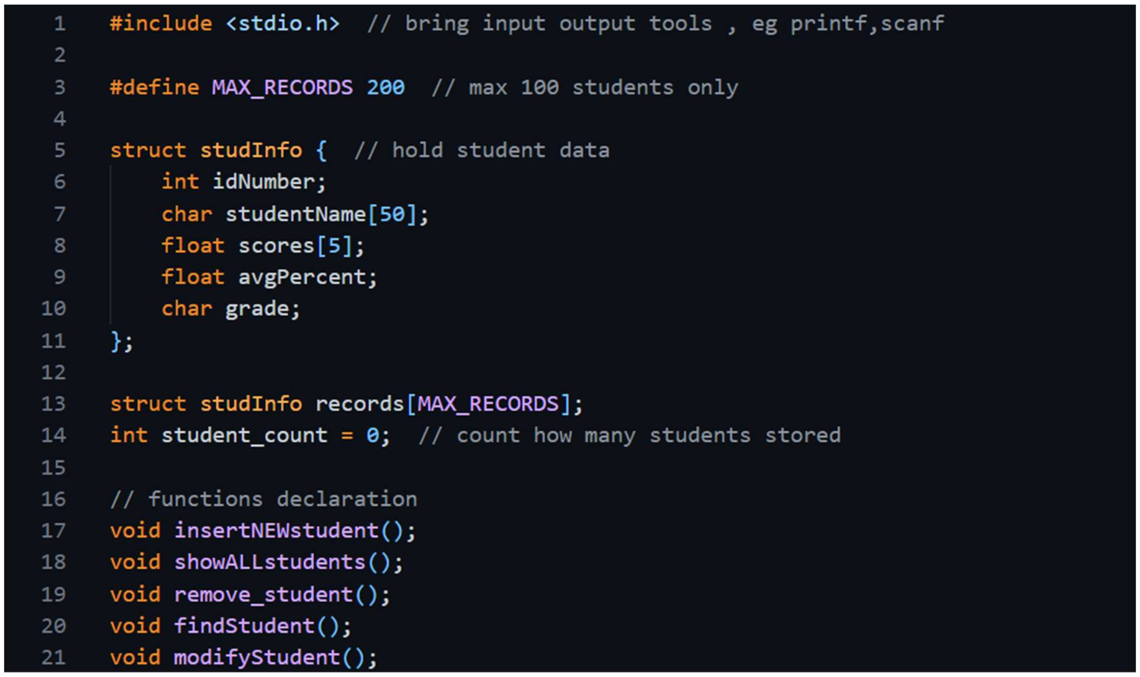
Conclusion

The Student Report Card Management System works well. It successfully shows that we can use the core parts of C programming—like defining structs, working with arrays, and splitting the program into different functions—to manage data effectively.

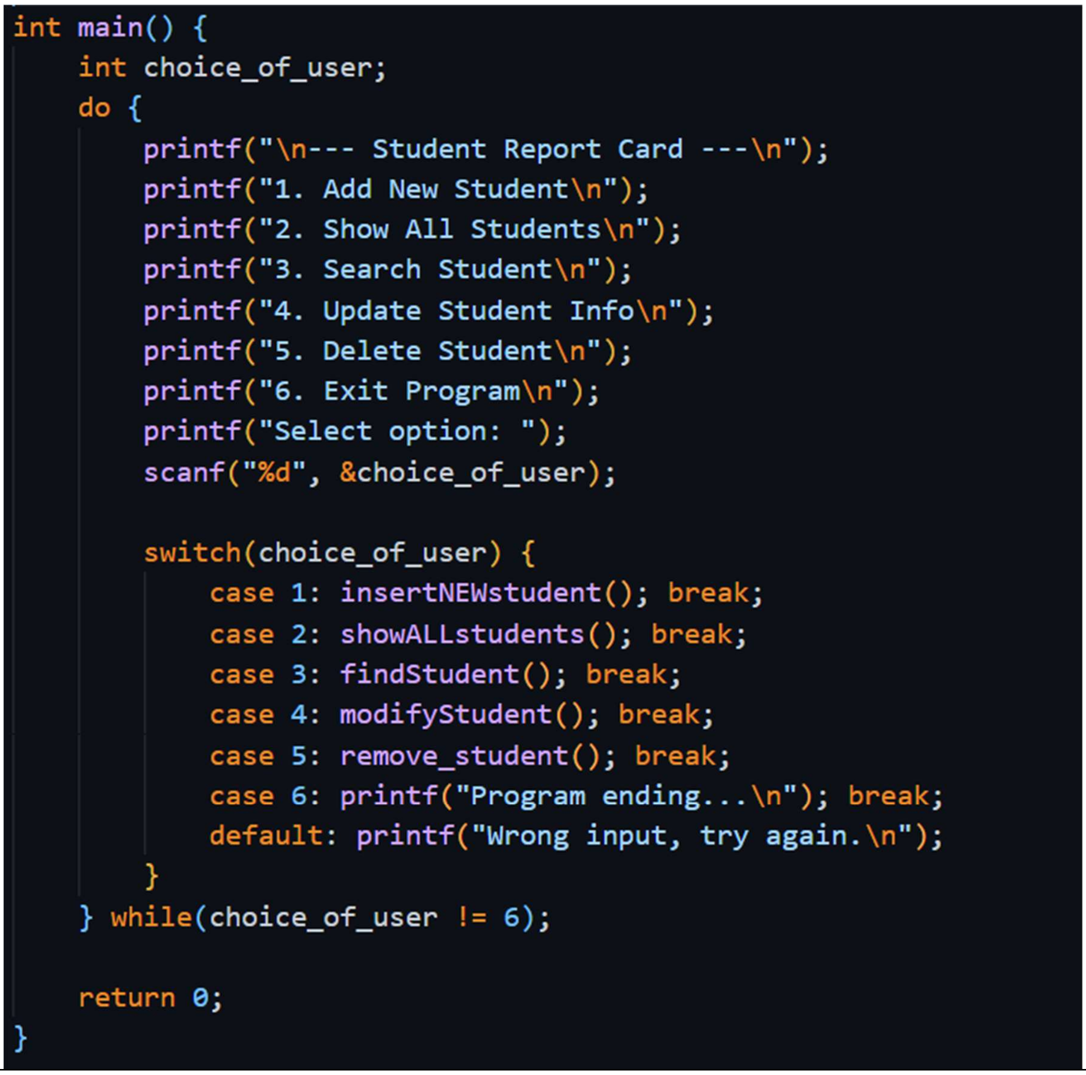
Future Work (What can be added later)

1. **Saving Data Permanently (File I/O):** Right now, the data is lost when the program closes. We need to add code to save the records array to a file (like records.dat) and load it back when the program starts.
2. **Making Input Safer:** We should add checks to make sure people enter valid numbers for scores (e.g., between 0 and 100) and prevent them from entering the same ID number twice.
3. **Better Search and View:** It would be helpful to allow users to search by **student name** and to have an option to view the list sorted by **average score**.

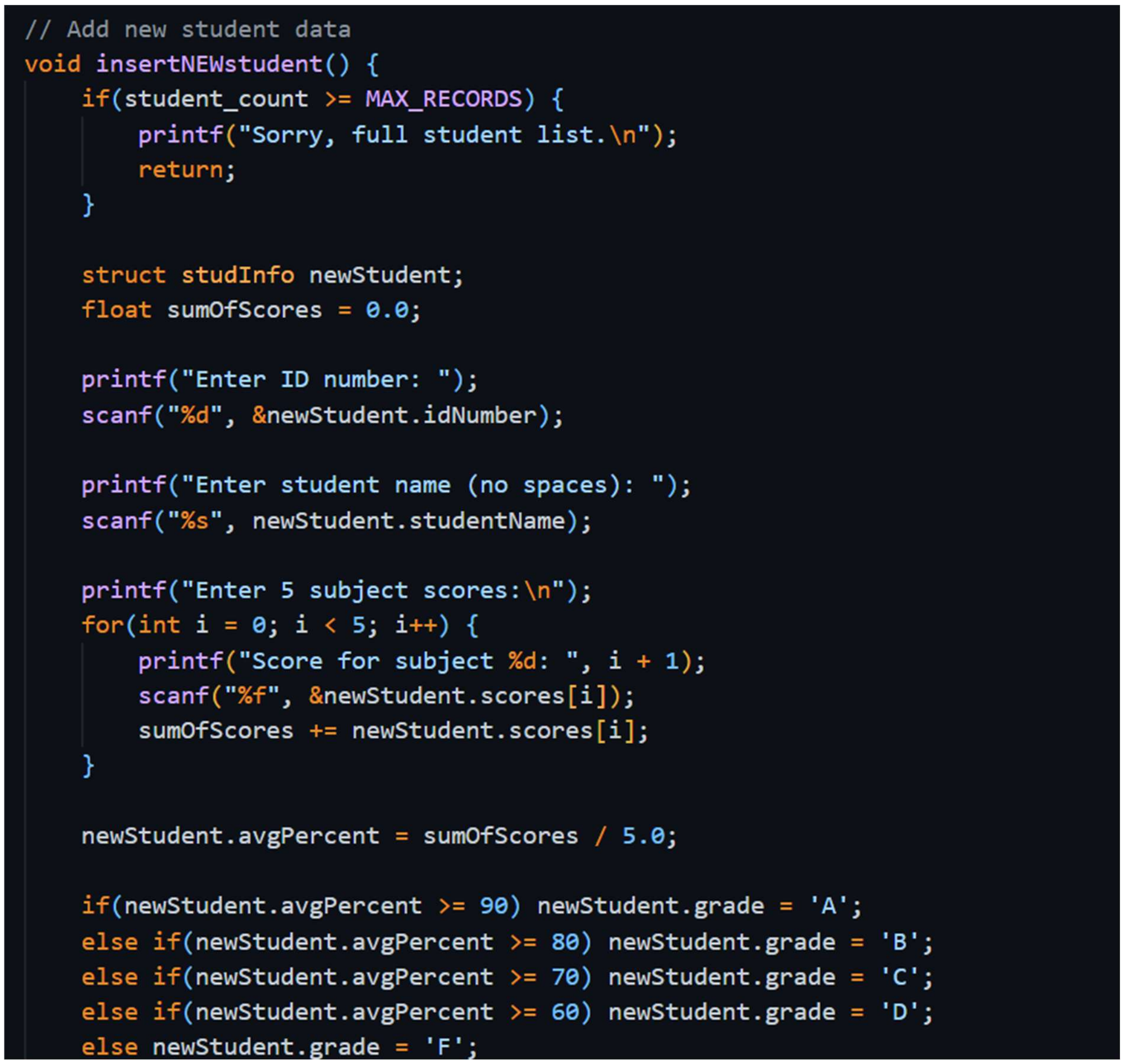
Appendix(Full Source Code)



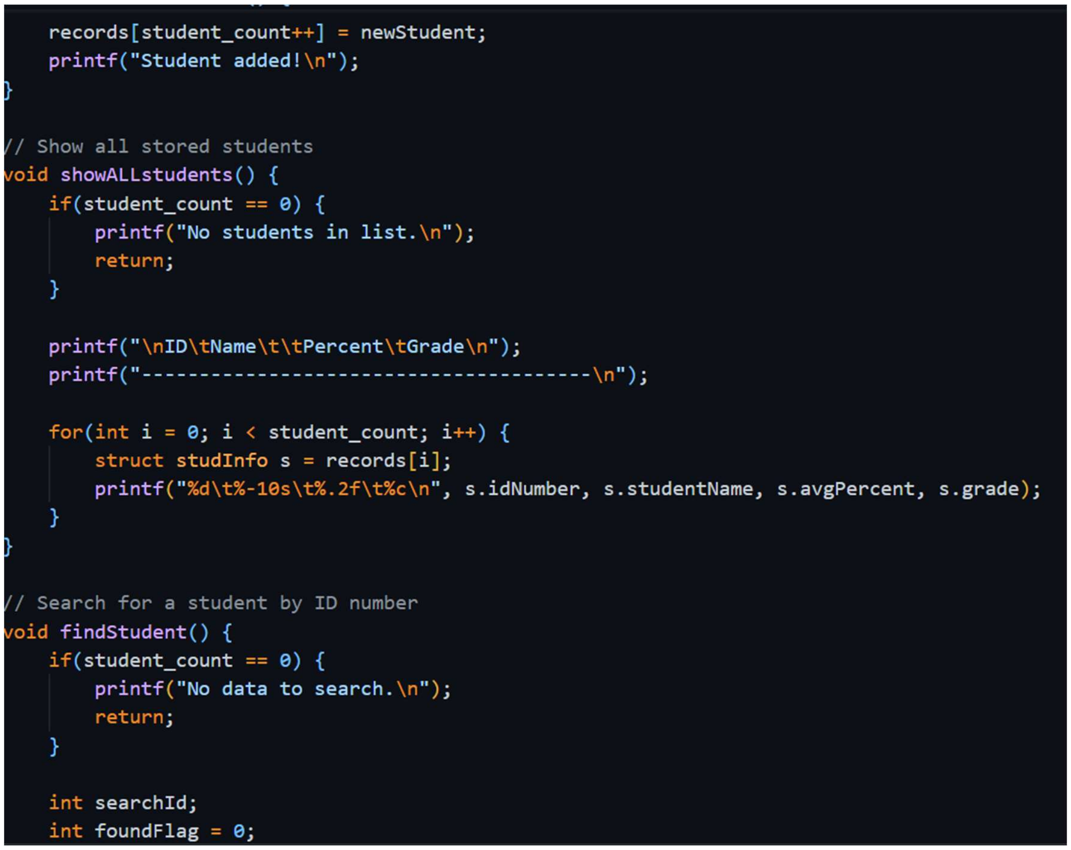
```c
#include <stdio.h>  // bring input output tools , eg printf,scanf

#define MAX_RECORDS 200  // max 100 students only

struct studInfo {  // hold student data
    int idNumber;
    char studentName[50];
    float scores[5];
    float avgPercent;
    char grade;
};

struct studInfo records[MAX_RECORDS];
int student_count = 0;  // count how many students stored

// functions declaration
void insertNEWstudent();
void showALLstudents();
void remove_student();
void findStudent();
void modifyStudent();
```



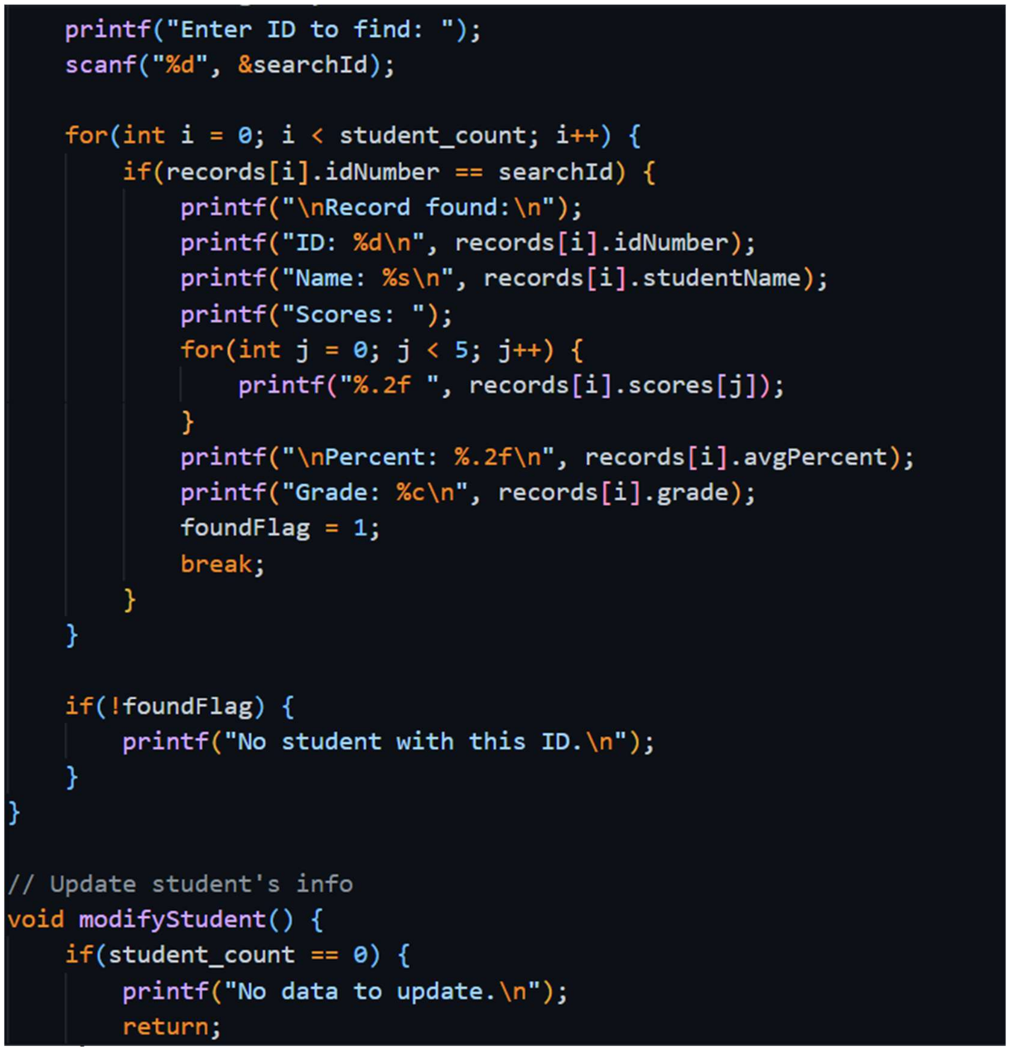
```c
int main() {
    int choice_of_user;
    do {
        printf("\n--- Student Report Card ---\n");
        printf("1. Add New Student\n");
        printf("2. Show All Students\n");
        printf("3. Search Student\n");
        printf("4. Update Student Info\n");
        printf("5. Delete Student\n");
        printf("6. Exit Program\n");
        printf("Select option: ");
        scanf("%d", &choice_of_user);

        switch(choice_of_user) {
            case 1: insertNEWstudent(); break;
            case 2: showALLstudents(); break;
            case 3: findStudent(); break;
            case 4: modifyStudent(); break;
            case 5: remove_student(); break;
            case 6: printf("Program ending...\n"); break;
            default: printf("Wrong input, try again.\n");
        }
    } while(choice_of_user != 6);

    return 0;
}
```



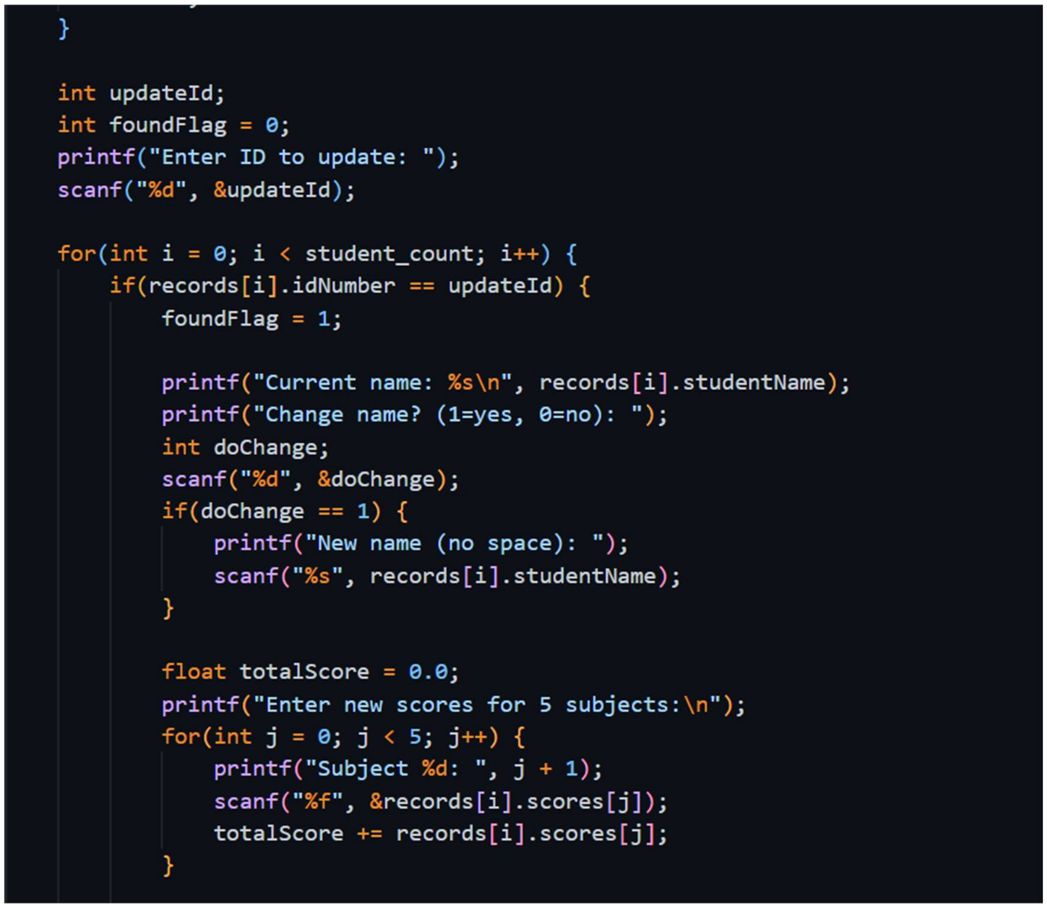
```c
// Add new student data
void insertNEWstudent() {
    if(student_count >= MAX_RECORDS) {
        printf("Sorry, full student list.\n");
        return;
    }

    struct studInfo newStudent;
    float sumOfScores = 0.0;

    printf("Enter ID number: ");
    scanf("%d", &newStudent.idNumber);

    printf("Enter student name (no spaces): ");
    scanf("%s", newStudent.studentName);

    printf("Enter 5 subject scores:\n");
    for(int i = 0; i < 5; i++) {
        printf("Score for subject %d: ", i + 1);
        scanf("%f", &newStudent.scores[i]);
        sumOfScores += newStudent.scores[i];
    }

    newStudent.avgPercent = sumOfScores / 5.0;

    if(newStudent.avgPercent >= 90) newStudent.grade = 'A';
    else if(newStudent.avgPercent >= 80) newStudent.grade = 'B';
    else if(newStudent.avgPercent >= 70) newStudent.grade = 'C';
    else if(newStudent.avgPercent >= 60) newStudent.grade = 'D';
    else newStudent.grade = 'F';
```



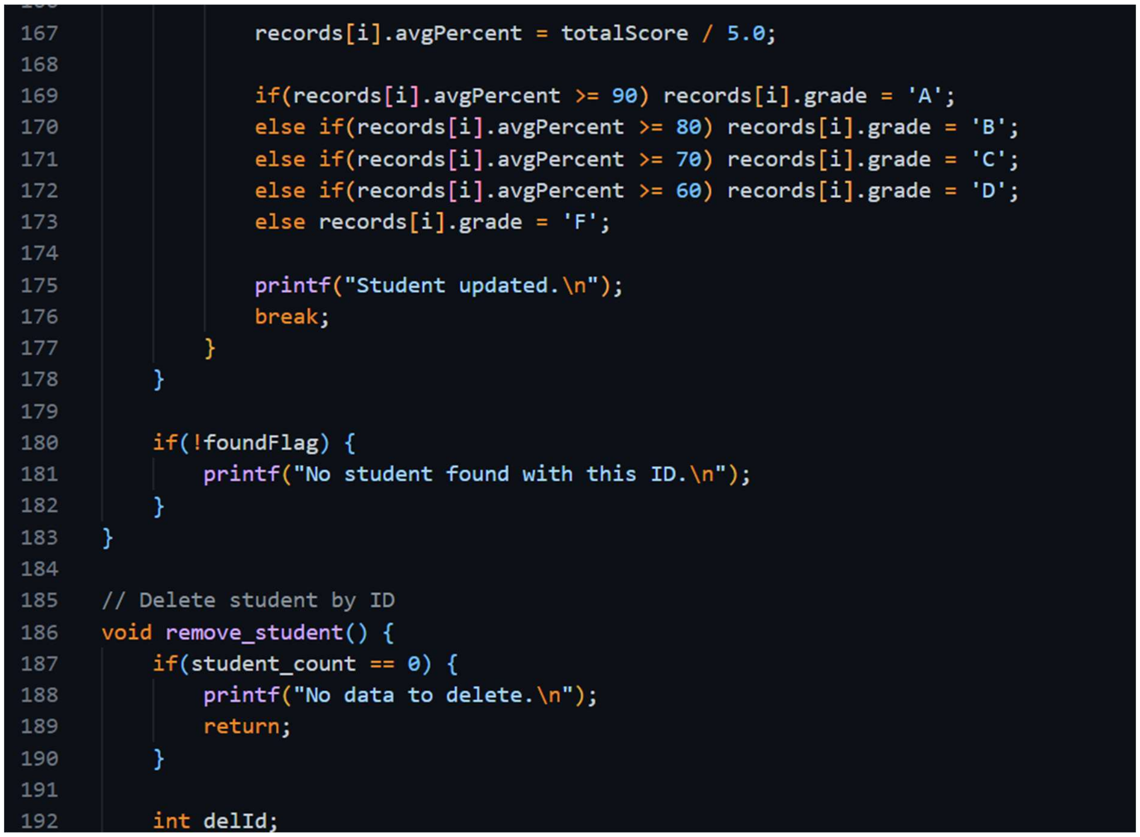
```c
    records[student_count++] = newStudent;
    printf("Student added!\n");
}

// Show all stored students
void showALLstudents() {
    if(student_count == 0) {
        printf("No students in list.\n");
        return;
    }

    printf("\nID\tName\t\tPercent\tGrade\n");
    printf("---------------------------------------\n");

    for(int i = 0; i < student_count; i++) {
        struct studInfo s = records[i];
        printf("%d\t%-10s\t%.2f\t%c\n", s.idNumber, s.studentName, s.avgPercent, s.grade);
    }
}

// Search for a student by ID number
void findStudent() {
    if(student_count == 0) {
        printf("No data to search.\n");
        return;
    }

    int searchId;
    int foundFlag = 0;
```



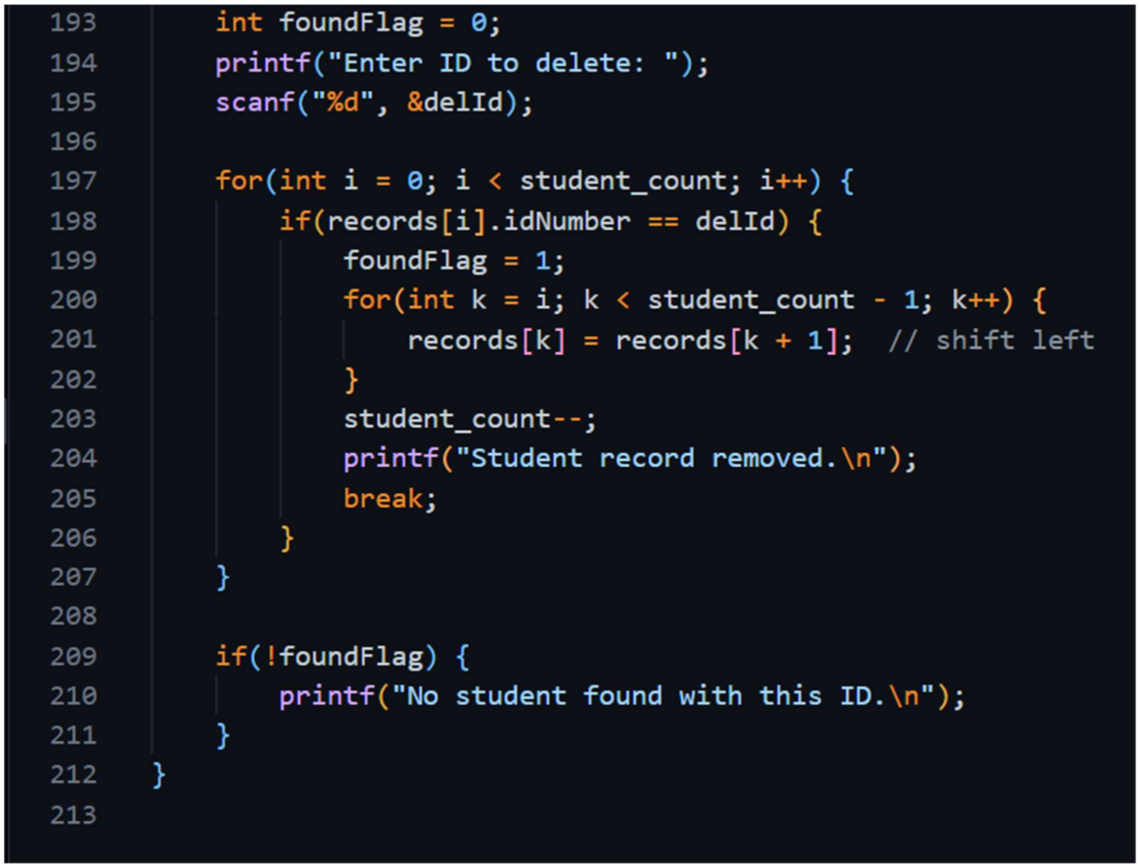
```c
    printf("Enter ID to find: ");
    scanf("%d", &searchId);

    for(int i = 0; i < student_count; i++) {
        if(records[i].idNumber == searchId) {
            printf("\nRecord found:\n");
            printf("ID: %d\n", records[i].idNumber);
            printf("Name: %s\n", records[i].studentName);
            printf("Scores: ");
            for(int j = 0; j < 5; j++) {
                printf("%.2f ", records[i].scores[j]);
            }
            printf("\nPercent: %.2f\n", records[i].avgPercent);
            printf("Grade: %c\n", records[i].grade);
            foundFlag = 1;
            break;
        }
    }

    if(!foundFlag) {
        printf("No student with this ID.\n");
    }
}

// Update student's info
void modifyStudent() {
    if(student_count == 0) {
        printf("No data to update.\n");
        return;
```



```c
    }

    int updateId;
    int foundFlag = 0;
    printf("Enter ID to update: ");
    scanf("%d", &updateId);

    for(int i = 0; i < student_count; i++) {
        if(records[i].idNumber == updateId) {
            foundFlag = 1;

            printf("Current name: %s\n", records[i].studentName);
            printf("Change name? (1=yes, 0=no): ");
            int doChange;
            scanf("%d", &doChange);
            if(doChange == 1) {
                printf("New name (no space): ");
                scanf("%s", records[i].studentName);
            }

            float totalScore = 0.0;
            printf("Enter new scores for 5 subjects:\n");
            for(int j = 0; j < 5; j++) {
                printf("Subject %d: ", j + 1);
                scanf("%f", &records[i].scores[j]);
                totalScore += records[i].scores[j];
            }
```



```c
            records[i].avgPercent = totalScore / 5.0;

            if(records[i].avgPercent >= 90) records[i].grade = 'A';
            else if(records[i].avgPercent >= 80) records[i].grade = 'B';
            else if(records[i].avgPercent >= 70) records[i].grade = 'C';
            else if(records[i].avgPercent >= 60) records[i].grade = 'D';
            else records[i].grade = 'F';

            printf("Student updated.\n");
            break;
        }
    }

    if(!foundFlag) {
        printf("No student found with this ID.\n");
    }
}

// Delete student by ID
void remove_student() {
    if(student_count == 0) {
        printf("No data to delete.\n");
        return;
    }

    int delId;
```



```c
    int foundFlag = 0;
    printf("Enter ID to delete: ");
    scanf("%d", &delId);

    for(int i = 0; i < student_count; i++) {
        if(records[i].idNumber == delId) {
            foundFlag = 1;
            for(int k = i; k < student_count - 1; k++) {
                records[k] = records[k + 1];  // shift left
            }
            student_count--;
            printf("Student record removed.\n");
            break;
        }
    }

    if(!foundFlag) {
        printf("No student found with this ID.\n");
    }
}
```

References

- Kernighan, B. W., & Ritchie, D. M. *The C Programming Language*, 2nd Edition, Prentice Hall.

- Yashavant Kanetkar, *Let Us C*, BPB Publications.

- Online C tutorials and documentation (e.g., cppreference, tutorialspoint, GeeksforGeeks).