LEARN-IN-DEPTH

Student Information Management System

Mostafa Ahmed Hesham Hammad Ahmed

9-27-2023

Contents

Introduction

Background

The Student Information Management System (SIMS) is a software project developed to address the needs of educational institutions, instructors, and administrators for efficient management of student records. In today's digital age, managing large volumes of student data can be a challenging task, and SIMS aims to streamline this process.

Objectives

The primary objectives of the SIMS project are:

- **Efficient Data Management:** Provide a user-friendly platform for storing, organizing, and accessing student information.
- **User Flexibility:** Enable users to add student records manually or import them from an external file.
- **Search and Retrieval:** Facilitate quick and accurate retrieval of student records using various search criteria.
- **Data Integrity:** Ensure the integrity of student data through robust update and delete functionalities.
- **Data Presentation:** Offer a comprehensive view of student information, including the total number of students and detailed profiles.

Scope

The scope of the SIMS project includes:

- Development of a command-line-based application in the C programming language.
- Management of student data, including their first name, last name, roll number, GPA, and enrolled courses.
- Implementation of user-friendly features for adding, searching, updating, and deleting student records.

System Overview

Description

The Student Information Management System (SIMS) is a console-based application designed for efficient management of student records. It features a menu-driven interface that guides users through various operations, ensuring ease of use.

Features

- **Add Student:** Users can add students to the system manually by providing their details or import multiple student records from a file.
- **Find Student:** SIMS offers multiple search options, allowing users to find students by roll number, first name, or course.
- **Update Student Information:** Users can update student information, including names, roll numbers, GPAs, and courses.
- **Delete Student:** The system supports the removal of student records, ensuring data accuracy.
- **Display Total Number of Students:** SIMS calculates and displays the total number of students in the database.
- **View All Students' Information:** Users can view comprehensive profiles of all students, including their personal details and course enrollments.

System Architecture

SIMS is structured as a console application with a menu-driven interface. It relies on a combination of data structures and algorithms to manage student data efficiently. The core components include:

- **Structs:** Student records are defined using structures, allowing for organized storage of information.
- **Arrays:** Arrays are used to store student data and course information. Dynamic arrays are employed to handle varying numbers of courses per student.
- **File Handling:** The system uses file handling mechanisms to import student data from external files.

Implementation

Programming Language and Tools

SIMS is implemented in the C programming language, chosen for its efficiency and portability. The following tools and concepts were used in its development:

- **GCC Compiler:** The GNU Compiler Collection was used for code compilation.
- **File I/O:** File Input/Output functions were utilized for reading student data from external files.

Data Structures

SIMS employs the following data structures:

- **Structs:** Structures are defined to represent student records, containing fields for first name, last name, roll number, GPA, and an array for course enrollments.
- **Arrays:** Arrays are used to store student records and course names. Dynamic arrays are employed to handle varying numbers of courses for each student.

Algorithms

Various algorithms are employed to achieve the system's functionality:

- **Parsing and Tokenization:** To extract data from input lines and break them into individual components.
- **Linear Search:** Used to search for students by roll number, first name, or course.
- **Data Manipulation:** Algorithms are employed to update, delete, and manage student records efficiently.

Functionality

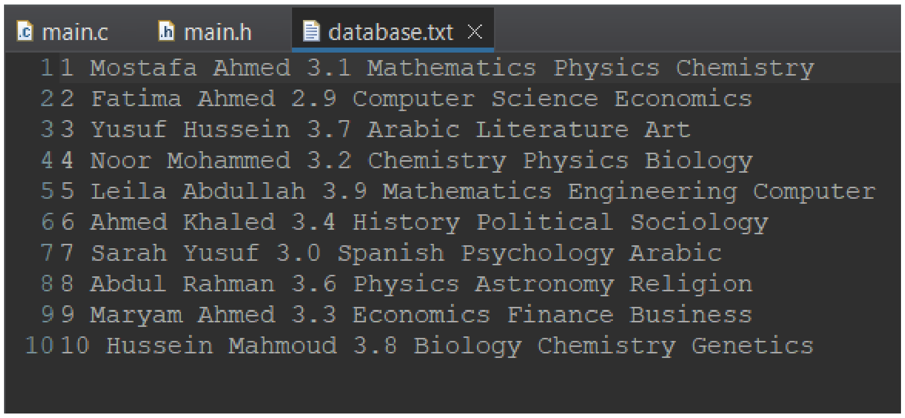
Add Student

Description: Users can add students to the system through two methods: manual entry or importing from an external file.

Manual Entry:

- Users are prompted to input the student's first name, last name, roll number, GPA, and up to three courses.
- Students are added to the database, and the system validates that the maximum limit of 55 students is not exceeded.
    - ➢ Choose the "Add Student Manually" option from the main menu.
    - ➢ Follow the prompts to enter the student's details.

File Import:



```
1 Mostafa Ahmed 3.1 Mathematics Physics Chemistry
2 Fatima Ahmed 2.9 Computer Science Economics
3 Yusuf Hussein 3.7 Arabic Literature Art
4 Noor Mohammed 3.2 Chemistry Physics Biology
5 Leila Abdullah 3.9 Mathematics Engineering Computer
6 Ahmed Khaled 3.4 History Political Sociology
7 Sarah Yusuf 3.0 Spanish Psychology Arabic
8 Abdul Rahman 3.6 Physics Astronomy Religion
9 Maryam Ahmed 3.3 Economics Finance Business
10 Hussein Mahmoud 3.8 Biology Chemistry Genetics
```

- Users can import student records from a file named "database.txt."
- Each line in the file represents a student record, with data separated by spaces.
- Imported students are added to the database.
    - ➢ Choose the "Add Students from File" option from the main menu.
    - ➢ Ensure that a file named "database.txt" exists with student records.
    - ➢ The system will import the data from the file.

Find Student

Description: Users can search for students in the system based on three criteria: roll number, first name, or course.

- **Roll Number:** Users input a roll number, and the system displays the student's details if found.
- **First Name:** Users input a first name, and the system displays details of all students with matching first names.
- **Course:** Users input a course ID, and the system displays details of all students enrolled in that course.
    - ➢ Choose the "Find Student" option from the main menu.
    - ➢ Select the search criterion (roll number, first name, or course).
    - ➢ Follow the prompts to provide the relevant information.
    - ➢ View the displayed student information.

Update Student Information

Description: Users can update a student's information by providing the student's roll number.

- Users are prompted to input updated information, including first name, last name, roll number, GPA, and courses.
- The system ensures data integrity by replacing the existing record with the updated one.
  - ➢ Choose the "Update Student Information" option from the main menu.
  - ➢ Enter the student's roll number to update.
  - ➢ Follow the prompts to input the updated information.

Delete Student

Description: Users can delete a student record by specifying the student's roll number.

- The system identifies the student by roll number and removes their record from the database.
- Data integrity is maintained by shifting records after the deleted student to the left.
  - ➢ Choose the "Delete Student" option from the main menu.
  - ➢ Enter the roll number of the student to be deleted.

Display Total Number of Students

Description: The system calculates and displays the total number of students in the database.

  - ➢ Select the "Total Number of Students" option from the main menu.

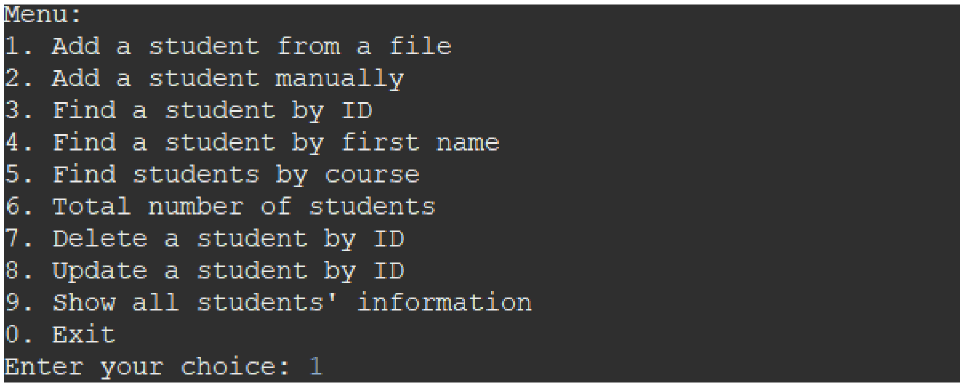
View All Students' Information

Description: Users can view detailed information about all students in the system.

- The system presents a comprehensive list of all students, including their first name, last name, roll number, GPA, and enrolled courses.
  - ➢ Choose the "Show All Students' Information" option from the main menu.

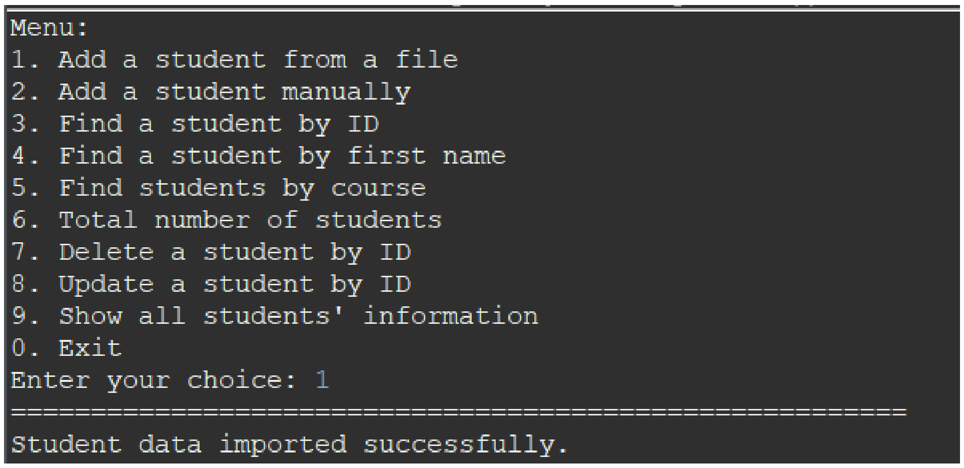
Testing

Menu



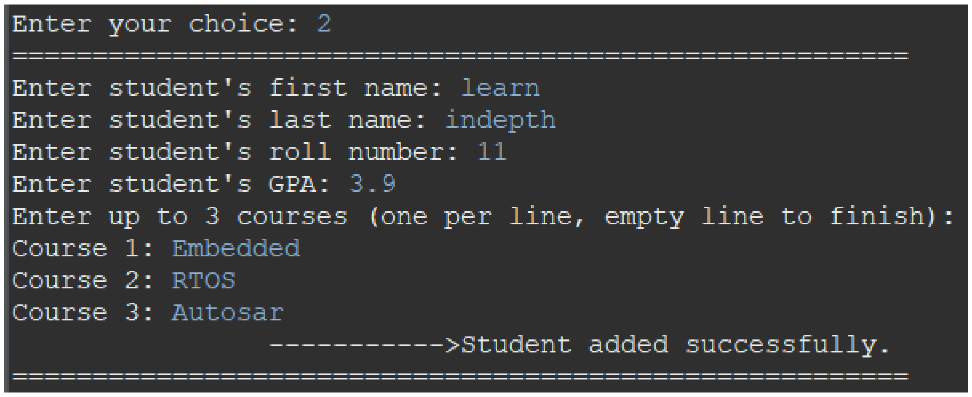
```
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 1
```

Add a student from a file.



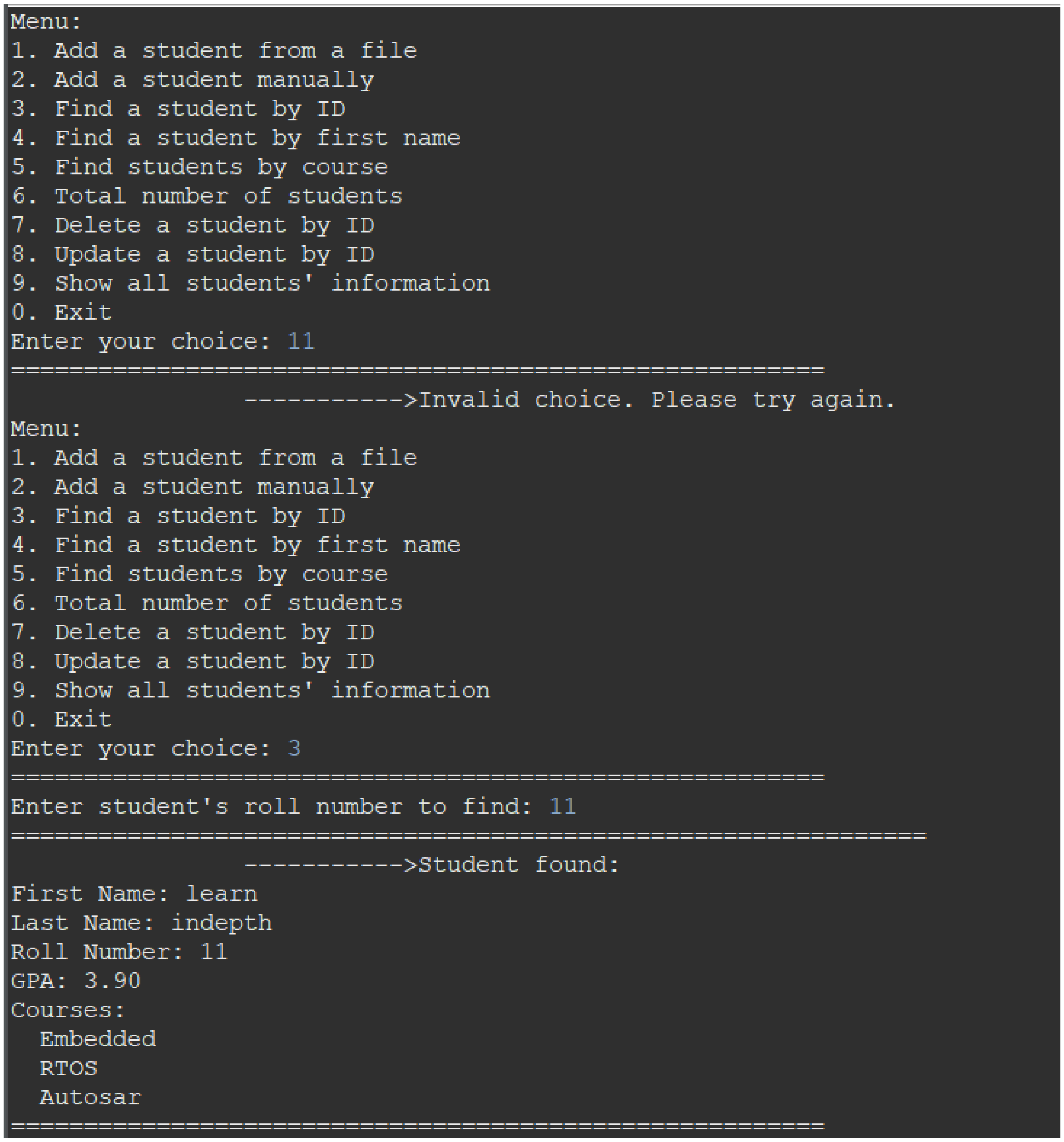
```
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 1
========================================================
Student data imported successfully.
```

Add a student manually.



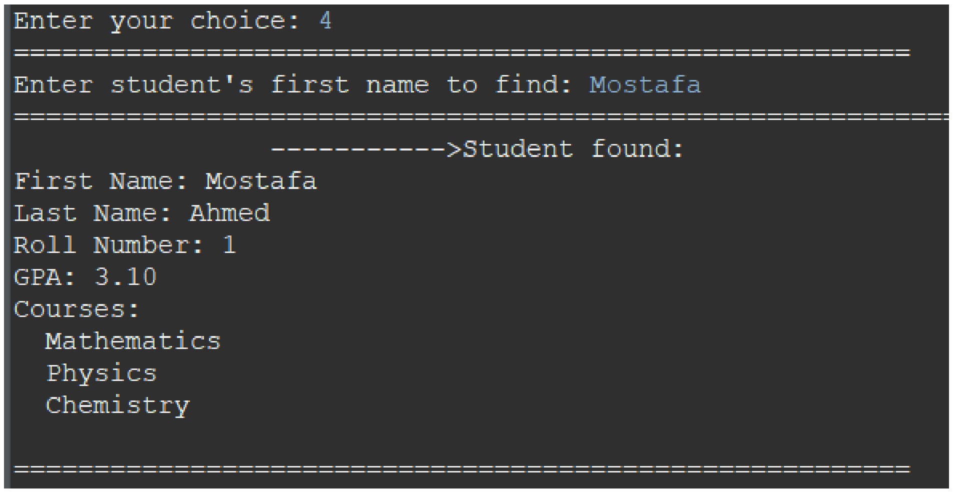
```
Enter your choice: 2
========================================================
Enter student's first name: learn
Enter student's last name: indepth
Enter student's roll number: 11
Enter student's GPA: 3.9
Enter up to 3 courses (one per line, empty line to finish):
Course 1: Embedded
Course 2: RTOS
Course 3: Autosar
                ----------->Student added successfully.
========================================================
```

Find a student by ID.



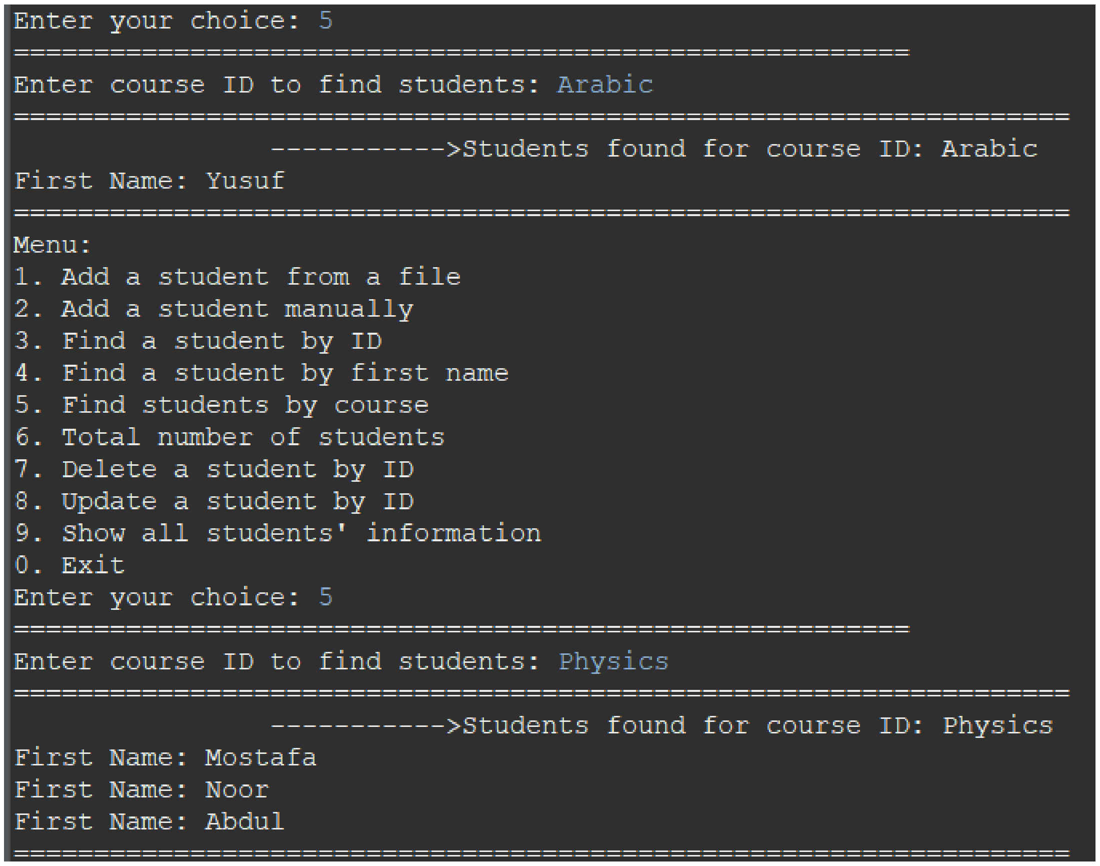
```
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 11
========================================================
                ----------->Invalid choice. Please try again.
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 3
========================================================
Enter student's roll number to find: 11
===============================================================
                ----------->Student found:
First Name: learn
Last Name: indepth
Roll Number: 11
GPA: 3.90
Courses:
  Embedded
  RTOS
  Autosar
========================================================
```

Find a student by first name.



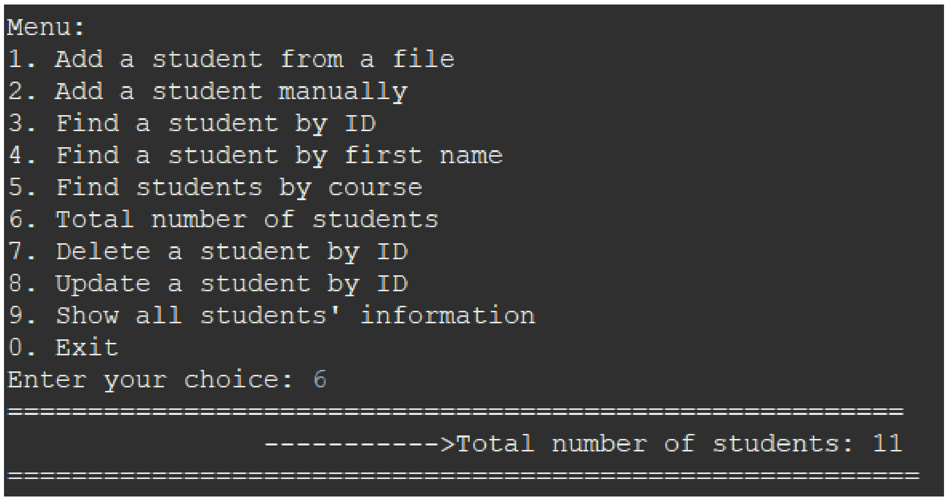
```
Enter your choice: 4
=========================================================
Enter student's first name to find: Mostafa
=============================================================
                ----------->Student found:
First Name: Mostafa
Last Name: Ahmed
Roll Number: 1
GPA: 3.10
Courses:
  Mathematics
  Physics
  Chemistry

=========================================================
```

Find students by course.



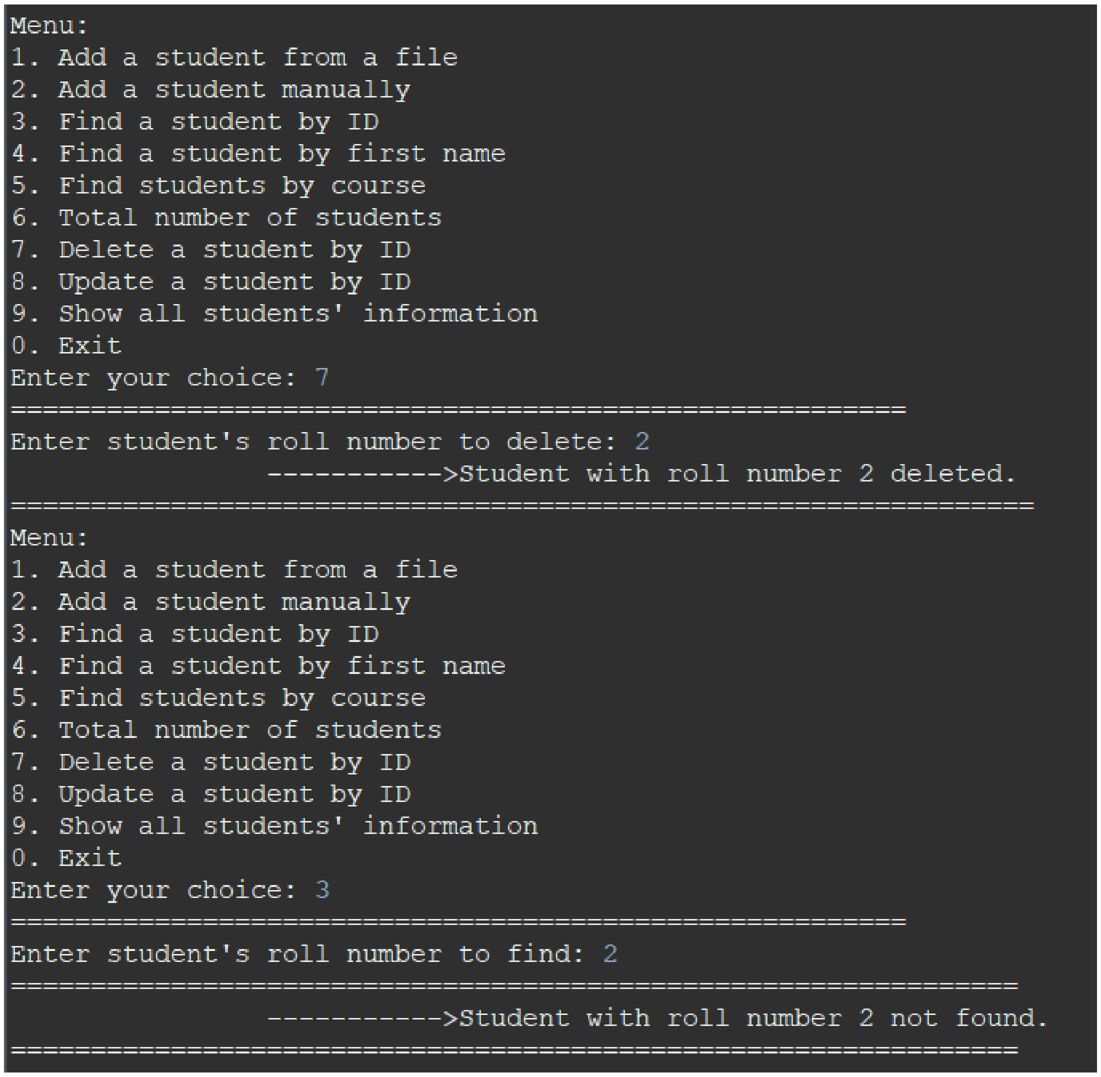
```
Enter your choice: 5
=========================================================
Enter course ID to find students: Arabic
===================================================================
                ----------->Students found for course ID: Arabic
First Name: Yusuf
===================================================================
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 5
=========================================================
Enter course ID to find students: Physics
===================================================================
                ----------->Students found for course ID: Physics
First Name: Mostafa
First Name: Noor
First Name: Abdul
===================================================================
```

Total number of students



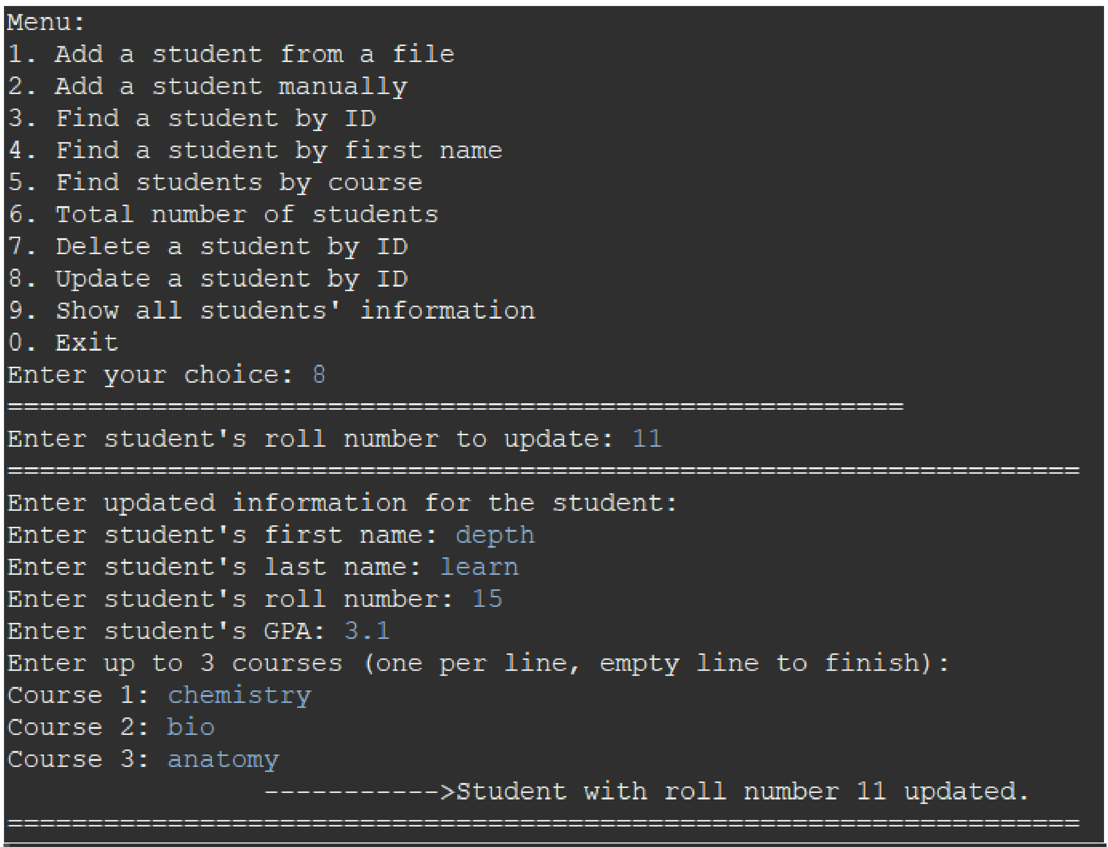
```
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 6
========================================================
                ----------->Total number of students: 11
=========================================================
```

Delete a student by ID.

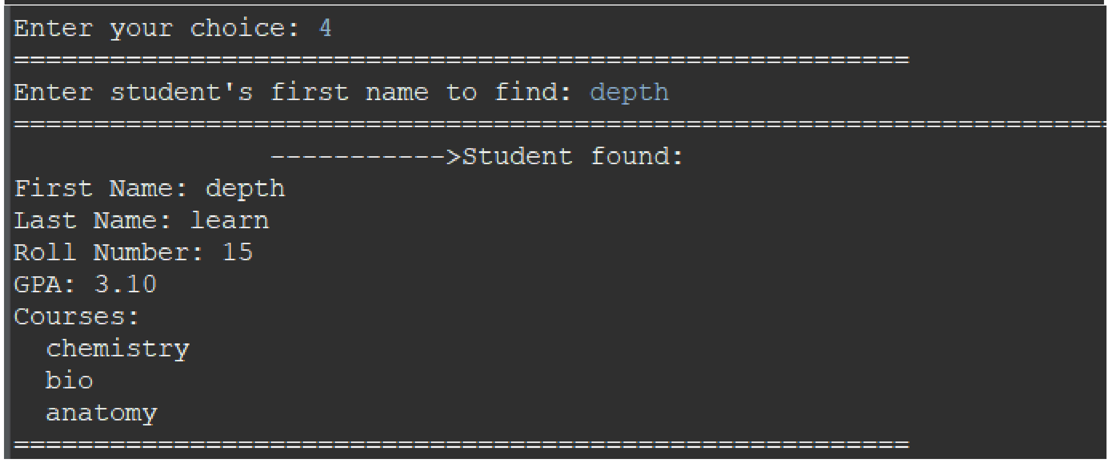


```
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 7
========================================================
Enter student's roll number to delete: 2
                ----------->Student with roll number 2 deleted.
================================================================
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 3
========================================================
Enter student's roll number to find: 2
===============================================================
                ----------->Student with roll number 2 not found.
===============================================================
```

Update a student by ID.



```
Menu:
1. Add a student from a file
2. Add a student manually
3. Find a student by ID
4. Find a student by first name
5. Find students by course
6. Total number of students
7. Delete a student by ID
8. Update a student by ID
9. Show all students' information
0. Exit
Enter your choice: 8
========================================================
Enter student's roll number to update: 11
===================================================================
Enter updated information for the student:
Enter student's first name: depth
Enter student's last name: learn
Enter student's roll number: 15
Enter student's GPA: 3.1
Enter up to 3 courses (one per line, empty line to finish):
Course 1: chemistry
Course 2: bio
Course 3: anatomy
                -----------&gt;Student with roll number 11 updated.
===================================================================
```



```
Enter your choice: 4
========================================================
Enter student's first name to find: depth
=====================================================================
                -----------&gt;Student found:
First Name: depth
Last Name: learn
Roll Number: 15
GPA: 3.10
Courses:
  chemistry
  bio
  anatomy
========================================================
```

Code Appendix

main.h

```
#ifndef MAIN_H_
#define MAIN_H_

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#define clear fflush(stdin); fflush(stdout)

struct Sinfo
{
        char fname[50];
        char lname[50];
        int roll;
        float GPA;
    char courses[3][50]  ;//Courses registered by the student
}st_base[55];


void add_student_file();
void add_student_manually();
void find_by_id();
void find_by_firstName();
void find_by_course();
void total_number_of_students();
void del_student_by_id();
void update_student_by_id();
void show_all_students_information();


#endif /* MAIN_H_ *
```

main.c

```c
#include "main.h"

int main() {
    int choice;

    do {
        printf("Menu:\n");
        printf("1. Add a student from a file\n");
        printf("2. Add a student manually\n");
        printf("3. Find a student by ID\n");
        printf("4. Find a student by first name\n");
        printf("5. Find students by course\n");
        printf("6. Total number of students\n");
        printf("7. Delete a student by ID\n");
        printf("8. Update a student by ID\n");
        printf("9. Show all students' information\n");
        printf("0. Exit\n");

        printf("Enter your choice: ");
        clear;
        scanf("%d", &choice);
        printf("========================================================\n");

        switch (choice) {
            case 1:
                add_student_file();
                break;
            case 2:
                add_student_manually();
                break;
            case 3:
                find_by_id();
                break;
            case 4:
                find_by_firstName();
                break;
            case 5:
                find_by_course();
                break;
            case 6:
                total_number_of_students();
                break;
            case 7:
                del_student_by_id();
                break;
            case 8:
                update_student_by_id();
                break;
            case 9:
                show_all_students_information();
                break;
            case 0:
                printf("\t\t----------->Exiting program.\n");
                break;
            default:
                printf("\t\t----------->Invalid choice. Please try again.\n");
        }
    } while (choice != 0);

    return 0;
}

/**
 * Function: add_student_file
 * Description: Reads student information from a file and adds it to the student
database.
 * Parameters: None
 * Returns: None
 */
int studentCount = 0; // Keep track of the number of students
void add_student_file() {
    FILE *fp;
    fp = fopen("database.txt", "r");

    if (fp == NULL) {
        printf("Failed to open the file.\n");
        return;
    }

    char line[100];

    while (fgets(line, sizeof(line), fp)) {
        int counter = 1;
        struct Sinfo new_s;
        char *token = strtok(line, " ");

        while (token != NULL) {
            switch (counter) {
                case 1:
                    new_s.roll = atoi(token);
                    break;
                case 2:
                    strcpy(new_s.fname, token);
                    break;
                case 3:
                    strcpy(new_s.lname, token);
                    break;
                case 4:
                    new_s.GPA = atof(token);
                    break;
                default:
                    if (counter - 5 <= 2) {
                        strcpy(new_s.courses[counter - 5], token);
                    }
            }
            token = strtok(NULL, " ");
            counter++;
        }

        // Add the new student to st_base array
        if (studentCount < 55) {
            st_base[studentCount] = new_s;
            studentCount++;
        } else {
            printf("Maximum number of students reached.\n");
            break;
        }
    }

    fclose(fp);
    printf("Student data imported successfully.\n");
}

/**
 * Function: add_student_manually
 * Description: Manually adds a student's information to the student database.
 * Parameters: None
 * Returns: None
 */

void add_student_manually() {
    if (studentCount < 55) {
        printf("Enter student's first name: ");
        clear;
        scanf("%s", st_base[studentCount].fname);
        printf("Enter student's last name: ");
        clear;
        scanf("%s", st_base[studentCount].lname);
        clear;
        printf("Enter student's roll number: ");
        clear;
        scanf("%d", &st_base[studentCount].roll);
        printf("Enter student's GPA: ");
        clear;
        scanf("%f", &st_base[studentCount].GPA);
        printf("Enter up to 3 courses (one per line, empty line to finish):\n");
        for (int i = 0; i < 3; i++) {
            printf("Course %d: ", i + 1);
            clear;
            scanf("%s", st_base[studentCount].courses[i]);

            // Stop input if the user enters an empty line
            if (st_base[studentCount].courses[i][0] == '\0') {
                break;
            }
        }

        studentCount++;
        printf("\t\t----------->Student added successfully.\n");
        printf("=========================================================\n");
    } else {
        printf("\t\t----------->Maximum number of students reached.\n");
        printf("=========================================================\n");
    }
}

/**
 * Function: find_by_id
 * Description: Finds and displays a student's information by their roll number.
 * Parameters: None
 * Returns: None
 */
void find_by_id() {
    int searchRoll;
    printf("Enter student's roll number to find: ");
    clear;
    scanf("%d", &searchRoll);

    printf("===============================================================\n");

    int found = 0; // To check if the student was found

    for (int i = 0; i < studentCount; i++) {
        if (st_base[i].roll == searchRoll) {
            printf("\t\t----------->Student found:\n");
            printf("First Name: %s\n", st_base[i].fname);
            printf("Last Name: %s\n", st_base[i].lname);
            printf("Roll Number: %d\n", st_base[i].roll);
            printf("GPA: %.2f\n", st_base[i].GPA);
            printf("Courses:\n");
            for (int j = 0; j < 3; j++) {
                if (st_base[i].courses[j][0] != '\0') {
                    printf("  %s\n", st_base[i].courses[j]);
                }
            }
            printf("========================================================\n");
            found = 1;
            break; // Student found, exit loop
        }
    }

    if (!found) {
        printf("\t\t----------->Student with roll number %d not found.\n", searchRoll);
        printf("===============================================================\n");
    }
}


/**
 * Function: find_by_firstName
 * Description: Finds and displays students' information by their first name.
 * Parameters: None
 * Returns: None
 */
void find_by_firstName() {
    char searchFirstName[50];
    printf("Enter student's first name to find: ");
    clear;
    scanf("%s", searchFirstName);


printf("===========================================================================\n");

    int found = 0; // To check if any students were found with the given first name

    for (int i = 0; i < studentCount; i++) {
        if (strcmp(st_base[i].fname, searchFirstName) == 0) {
            printf("\t\t----------->Student found:\n");
            printf("First Name: %s\n", st_base[i].fname);
            printf("Last Name: %s\n", st_base[i].lname);
            printf("Roll Number: %d\n", st_base[i].roll);
            printf("GPA: %.2f\n", st_base[i].GPA);
            printf("Courses:\n");
            for (int j = 0; j < 3; j++) {
                if (st_base[i].courses[j][0] != '\0') {
                    printf("  %s\n", st_base[i].courses[j]);
                }
            }
            printf("=========================================================\n");
            found = 1;
        }
    }

    if (!found) {
        printf("\t\t----------->No students found with the first name: %s\n",
searchFirstName);

printf("=====================================================================\n");
    }
}

/**
 * Function: find_by_course
 * Description: Finds and displays students' information by a specific course.
 * Parameters: None
 * Returns: None
 */

void find_by_course() {
    char searchCourse[100];
    printf("Enter course ID to find students: ");
    clear;
    scanf("%s", searchCourse);

    printf("=================================================================\n");

    int found = 0; // To check if any students were found for the given course

    for (int i = 0; i < studentCount; i++) {
        for (int j = 0; j < 3; j++) {
            if (strcmp(st_base[i].courses[j], searchCourse) == 0) {
                found = 1;
                break; // Student found for the course, exit inner loop
            }
        }
    }
    if (found)
    {
       printf("\t\t----------->Students found for course ID: %s\n", searchCourse);
        for (int i = 0; i < studentCount; i++) {
            for (int j = 0; j < 3; j++) {
                if (strcmp(st_base[i].courses[j], searchCourse) == 0) {
                    printf("First Name: %s\n", st_base[i].fname);
                    break; // Student found for the course, exit inner loop
                }
            }
        }
        printf("=================================================================\n");
    }

    if (!found) {
        printf("\t\t----------->No students found for course ID: %s\n", searchCourse);
        printf("=================================================================\n");
```

```c
    }
}



/**
 * Function: total_number_of_students
 * Description: Displays the total number of students in the database.
 * Parameters: None
 * Returns: None
 */

void total_number_of_students() {
    printf("\t\t----------->Total number of students: %d\n", studentCount);
    printf("==========================================================\n");
}


/**
 * Function: del_student_by_id
 * Description: Deletes a student from the database by their roll number.
 * Parameters: None
 * Returns: None
 */

void del_student_by_id() {
    int deleteRoll;
    printf("Enter student's roll number to delete: ");
    clear;
    scanf("%d", &deleteRoll);

    for (int i = 0; i < studentCount; i++) {
        if (st_base[i].roll == deleteRoll) {
            // Shift all elements after the found student to the left
            for (int j = i; j < studentCount - 1; j++) {
                st_base[j] = st_base[j + 1];
            }
            studentCount--;
            printf("\t\t----------->Student with roll number %d deleted.\n", deleteRoll);
            printf("=================================================================\n");
            return;
        }
    }

    printf("\t\t----------->Student with roll number %d not found.\n", deleteRoll);
    printf("====================================================================\n");
}


/**
 * Function: update_student_by_id
 * Description: Updates a student's information in the database by their roll number.
 * Parameters: None
 * Returns: None
 */
void update_student_by_id() {
    int updateRoll;
    printf("Enter student's roll number to update: ");
    clear;
    scanf("%d", &updateRoll);

    printf("===================================================================\n");

    int found = 0; // To check if the student was found

    for (int i = 0; i < studentCount; i++) {
        if (st_base[i].roll == updateRoll) {
            printf("Enter updated information for the student:\n");
            printf("Enter student's first name: ");
            clear;
            scanf("%s", st_base[i].fname);
            printf("Enter student's last name: ");
            clear;
            scanf("%s", st_base[i].lname);
            printf("Enter student's roll number: ");
            clear;
            scanf("%d", &st_base[i].roll);
            printf("Enter student's GPA: ");
            clear;
            scanf("%f", &st_base[i].GPA);

            printf("Enter up to 3 courses (one per line, empty line to finish):\n");
            for (int j = 0; j < 3; j++) {
                printf("Course %d: ", j + 1);
                clear;
                scanf("%s", st_base[i].courses[j]);

                // Stop input if the user enters an empty line
                if (st_base[i].courses[j][0] == '\0') {
                    break;
                }
            }

            printf("\t\t----------->Student with roll number %d updated.\n", updateRoll);

printf("===================================================================\n");
            found = 1;
            break; // Student found, exit loop
        }
    }

    if (!found) {
        printf("\t\t----------->Student with roll number %d not found.\n", updateRoll);
        printf("===================================================================\n");
    }
}


/**
 * Function: show_all_students_information
 * Description: Displays information for all students in the database.
 * Parameters: None
 * Returns: None
 */
void show_all_students_information() {
    if (studentCount == 0) {
        printf("No students to display.\n");
        return;
    }

    printf("All Students:\n");
    for (int i = 0; i < studentCount; i++) {
        printf("Student %d:\n", i + 1);
        printf("First Name: %s\n", st_base[i].fname);
        printf("Last Name: %s\n", st_base[i].lname);
        printf("Roll Number: %d\n", st_base[i].roll);
        printf("GPA: %.2f\n", st_base[i].GPA);

        printf("Courses:\n");
        for (int j = 0; j < 3; j++) {
            if (st_base[i].courses[j][0] != '\0') {
                printf("  %s\n", st_base[i].courses[j]);
            }
        }
        printf("==========================\n");
    }
}
```