Mastering
Embedded System
Online Diploma

School Management System Project

Mastering Embedded Systems Diploma

www.learn-in-depth.com

First Term (Final Project 2)

Eng. Amr Esaam Mohammed Zidan

My Profile: https://www.learn-in-depth.com/online-diploma/amresaam1342000%40gmail.com

Contents

1. Case Study

- ### Requirements

The client requires a system with the following specifications:

1. Adding students to the system manually and using a text file.
2. Finding student's data through (roll number, first name and course id).
3. Find the count of the total students on the system.
4. Delete student's data from the system.
5. Update student's data on the system.
6. Show all students' data on the system.

- ### Assumptions

The following assumptions are made:

1. The school's computers meet the minimum requirements for the application to run.
2. No storage on the computer's hard drive.

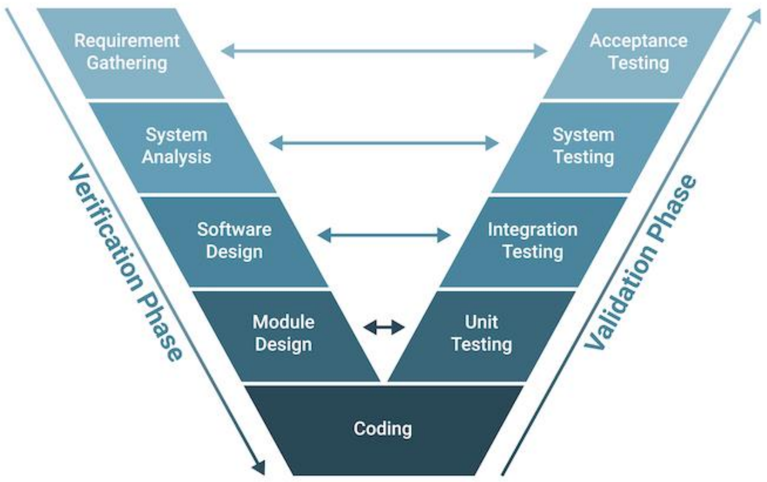
- ### Versioning

1. The possibility of adding a function to delete all students.
2. The enhancement of error handling in all corner cases.

2.Method

Software Development Lifecycle and Software Testing Lifecycle

The (SDLC) and (STLC) will be approached based on the V-Model.



Requirements Gathering and Analysis: The first phase of the V-Model is the requirements gathering and analysis phase, where the customer's requirements for the software are gathered and analyzed to determine the scope of the project.

Design: In the design phase, the software architecture and design are developed, including the high-level design and detailed design.

Implementation: In the implementation phase, the software is actually built based on the design.

Testing: In the testing phase, the software is tested to ensure that it meets the customer's requirements and is of high quality.

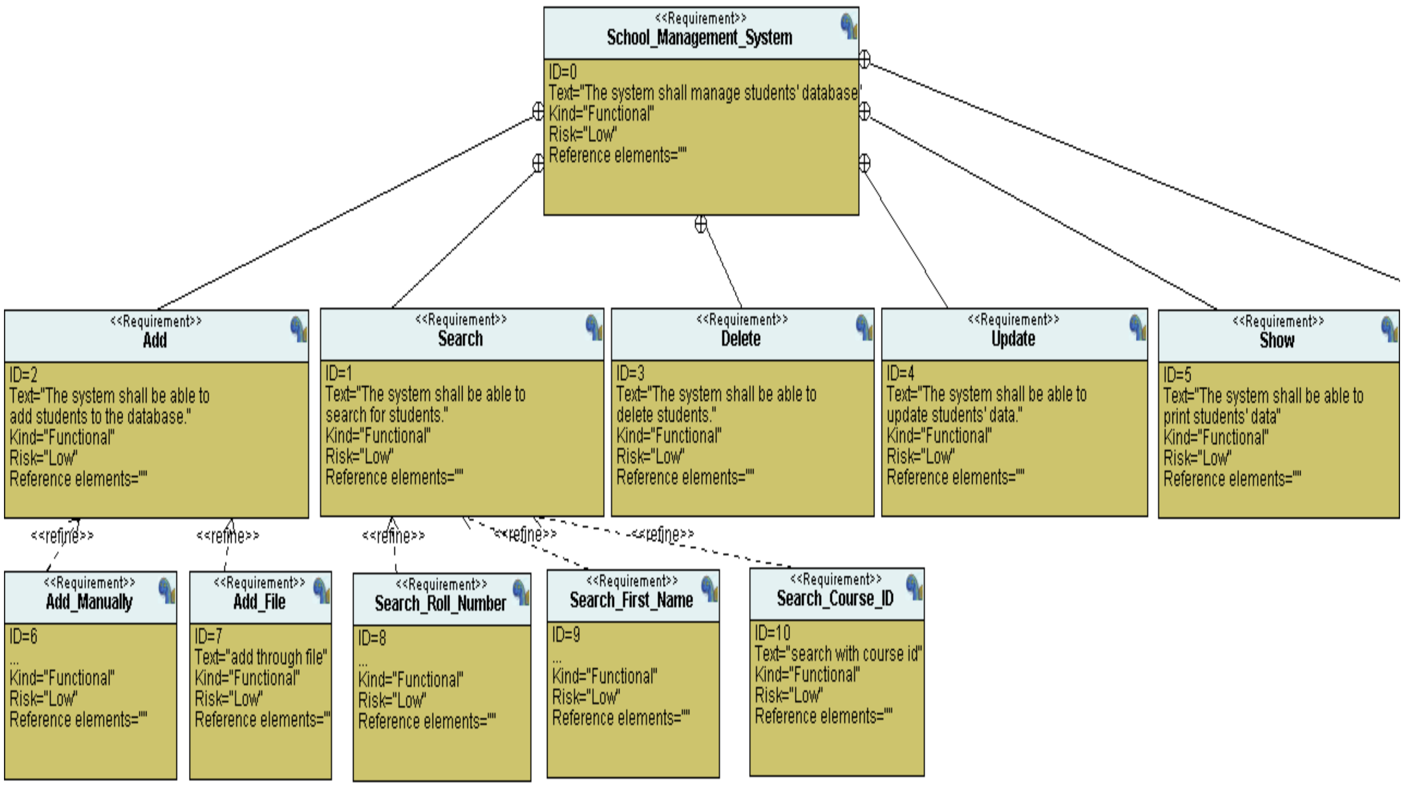
Deployment: In the deployment phase, the software is deployed and put into use.

Maintenance: In the maintenance phase, the software is maintained to ensure that it continues to meet the customer's needs and expectations.

The V-Model is often used in safety-critical systems, such as aerospace and defense systems, because of its emphasis on thorough testing and its ability to clearly define the steps involved in the software development process.

3. System Requirements

Requirement model



4. Design Space Exploration

To run the system on the school's computers a PC with at least windows 10 must be available.

RAM: 4GB

CPU: Core i3 7th Generation

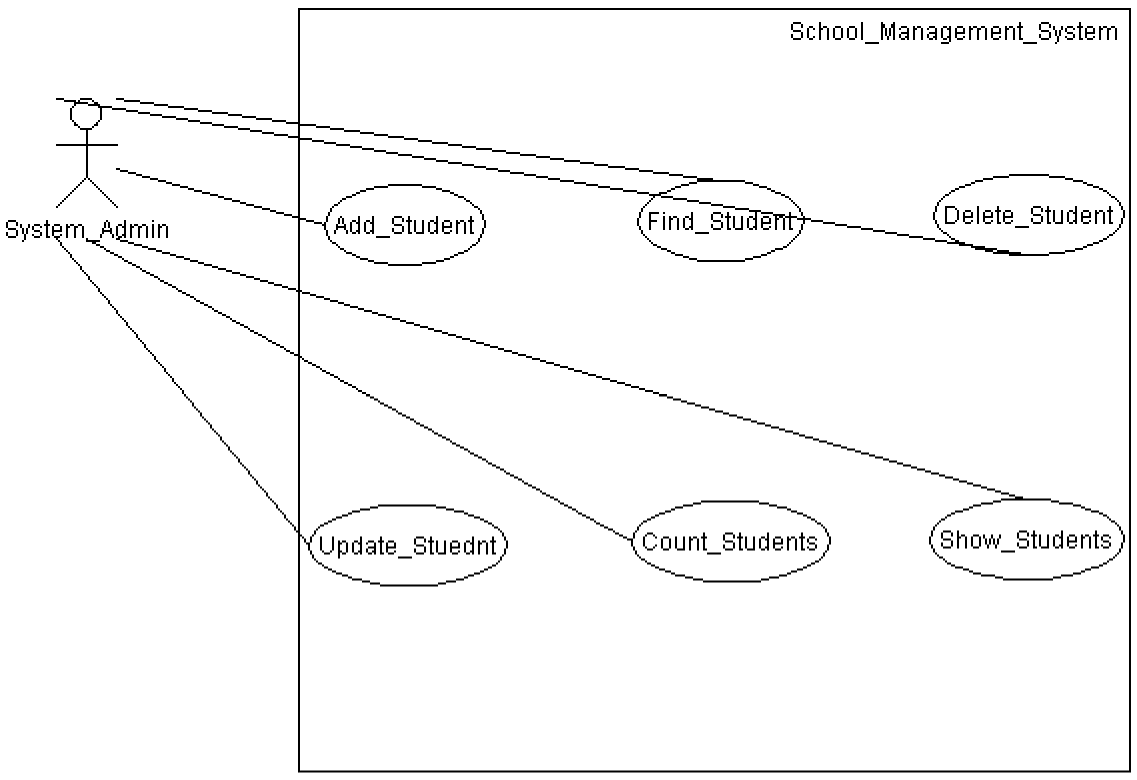
Graphics Card: No recommendations

GCC Compiler

Standard C Libraries

4.System Analysis

- Use Case Diagram



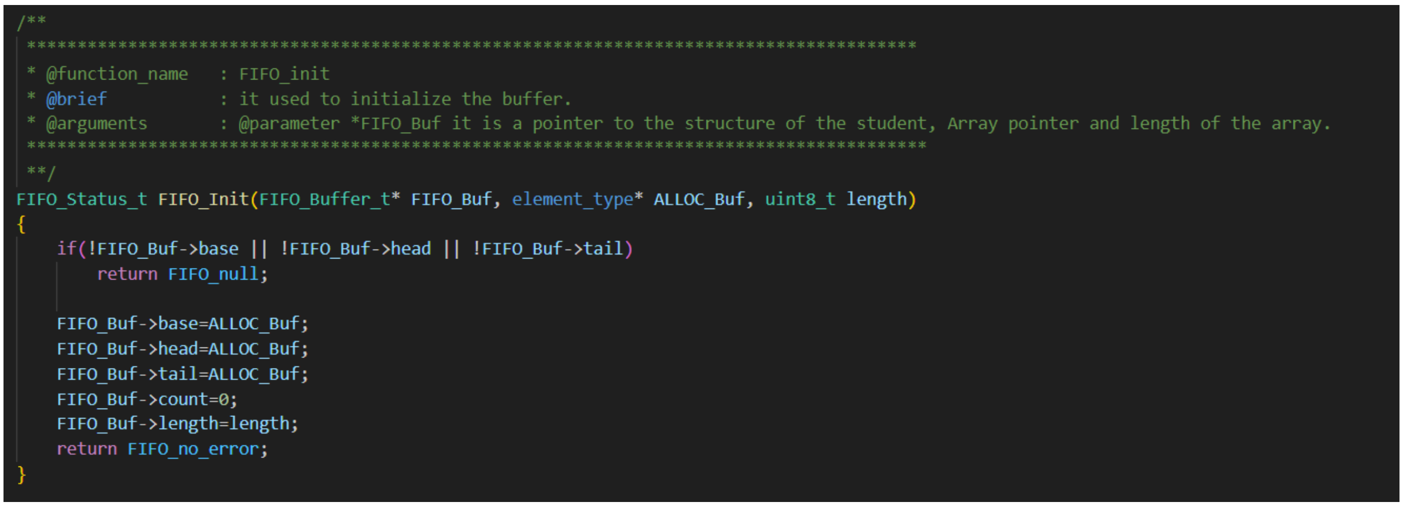
5.System Design

- C Codes

school_fifo

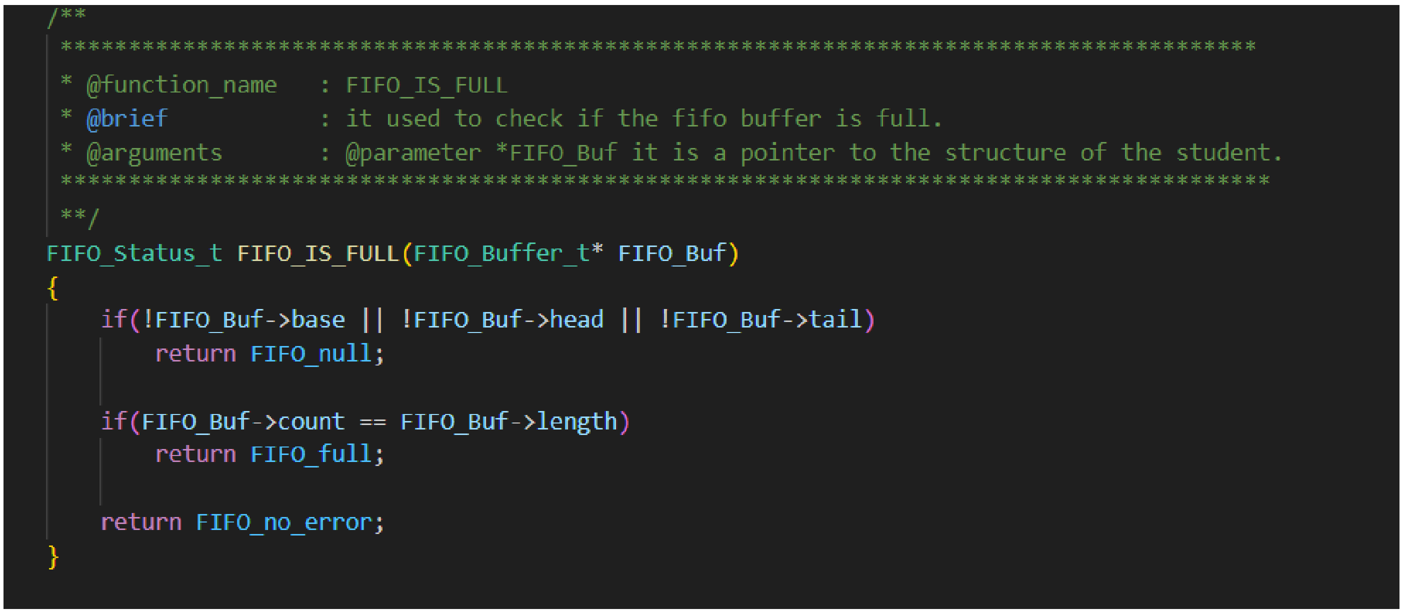
.c files

FIFO_init



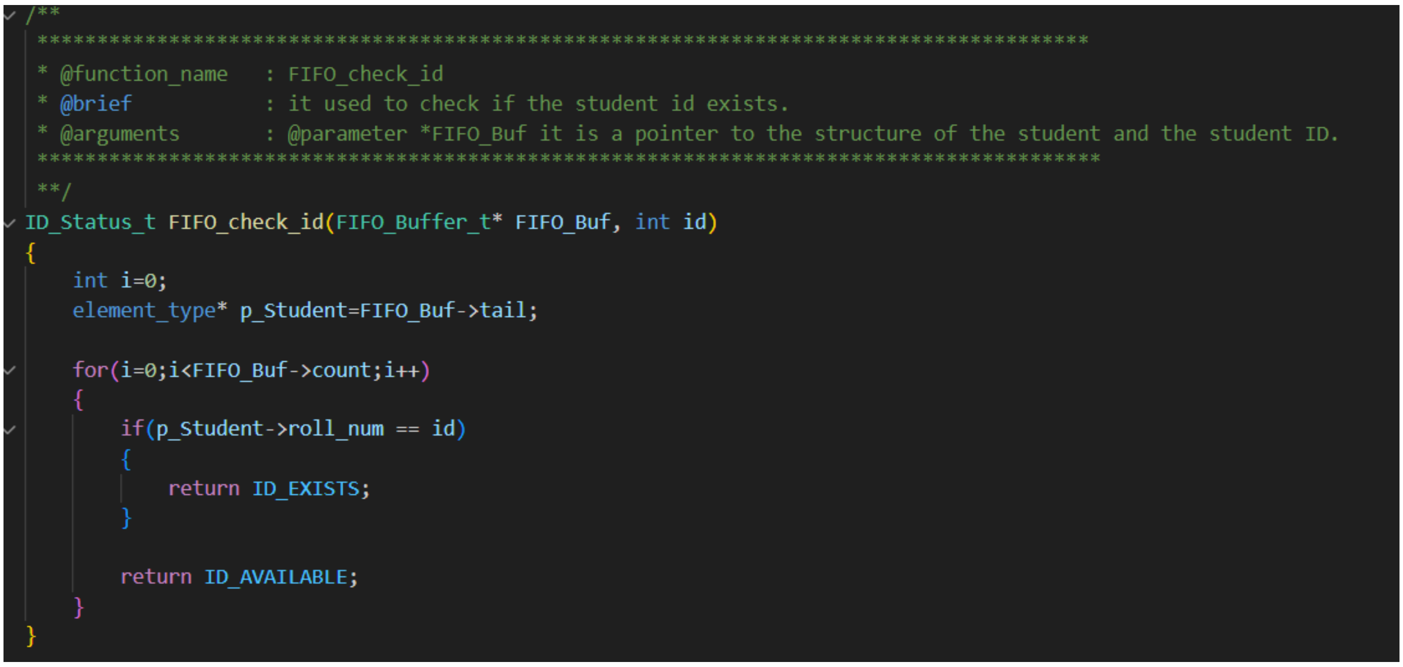
```c
/**
 **************************************************************************************
 * @function_name   : FIFO_init
 * @brief           : it used to initialize the buffer.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student, Array pointer and length of the array.
 **************************************************************************************
 **/
FIFO_Status_t FIFO_Init(FIFO_Buffer_t* FIFO_Buf, element_type* ALLOC_Buf, uint8_t length)
{
    if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
        return FIFO_null;

    FIFO_Buf->base=ALLOC_Buf;
    FIFO_Buf->head=ALLOC_Buf;
    FIFO_Buf->tail=ALLOC_Buf;
    FIFO_Buf->count=0;
    FIFO_Buf->length=length;
    return FIFO_no_error;
}
```

FIFO_IS_FULL



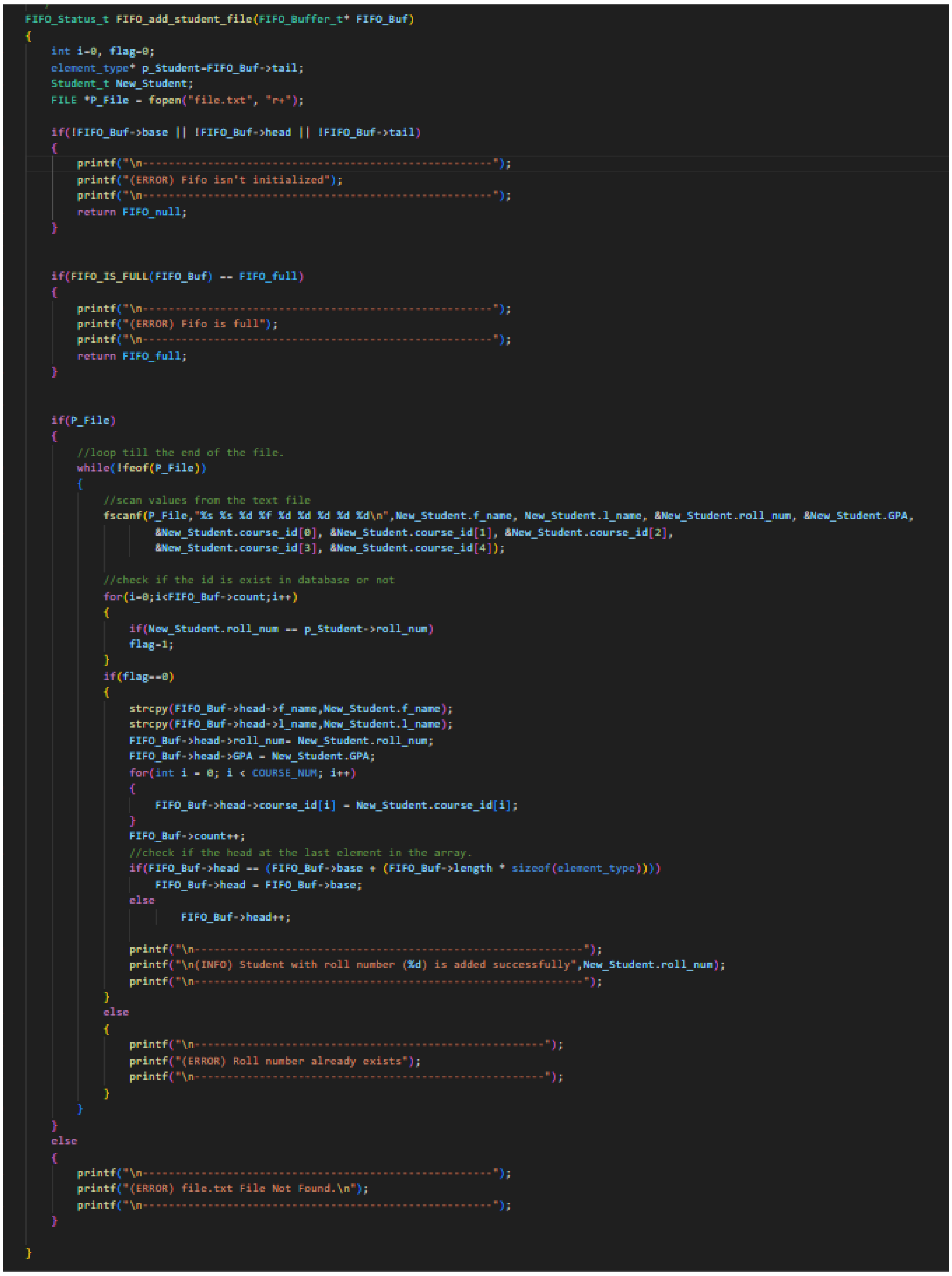
```c
/**
 **************************************************************************************
 * @function_name   : FIFO_IS_FULL
 * @brief           : it used to check if the fifo buffer is full.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student.
 **************************************************************************************
 **/
FIFO_Status_t FIFO_IS_FULL(FIFO_Buffer_t* FIFO_Buf)
{
    if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
        return FIFO_null;

    if(FIFO_Buf->count == FIFO_Buf->length)
        return FIFO_full;

    return FIFO_no_error;
}
```

FIFO_check_id



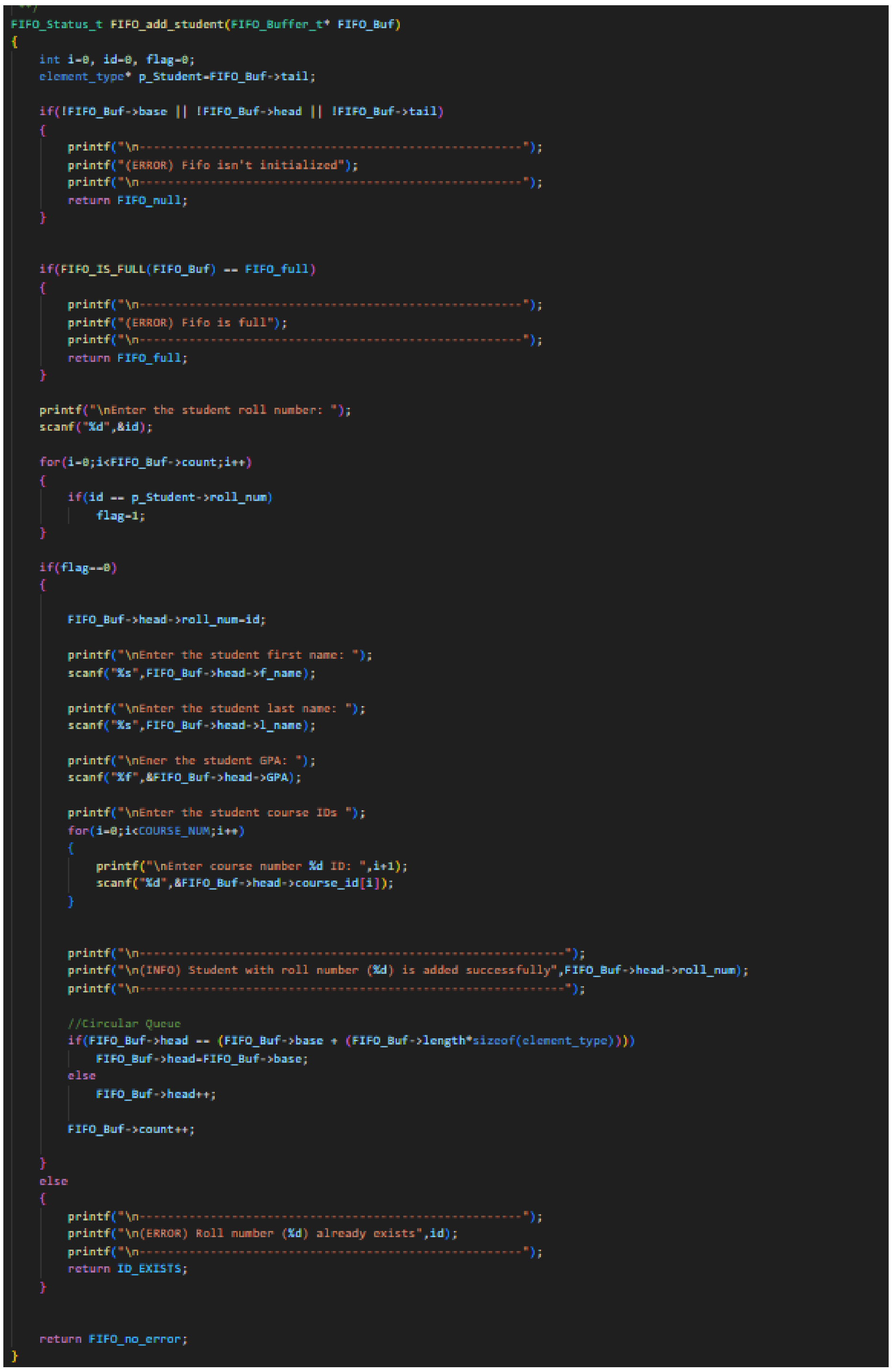
```c
/**
 ****************************************************************************************
 * @function_name   : FIFO_check_id
 * @brief           : it used to check if the student id exists.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student and the student ID.
 ****************************************************************************************
 **/
ID_Status_t FIFO_check_id(FIFO_Buffer_t* FIFO_Buf, int id)
{
    int i=0;
    element_type* p_Student=FIFO_Buf->tail;

    for(i=0;i<FIFO_Buf->count;i++)
    {
        if(p_Student->roll_num == id)
        {
            return ID_EXISTS;
        }

        return ID_AVAILABLE;
    }
}
```

FIFO_add_student_file



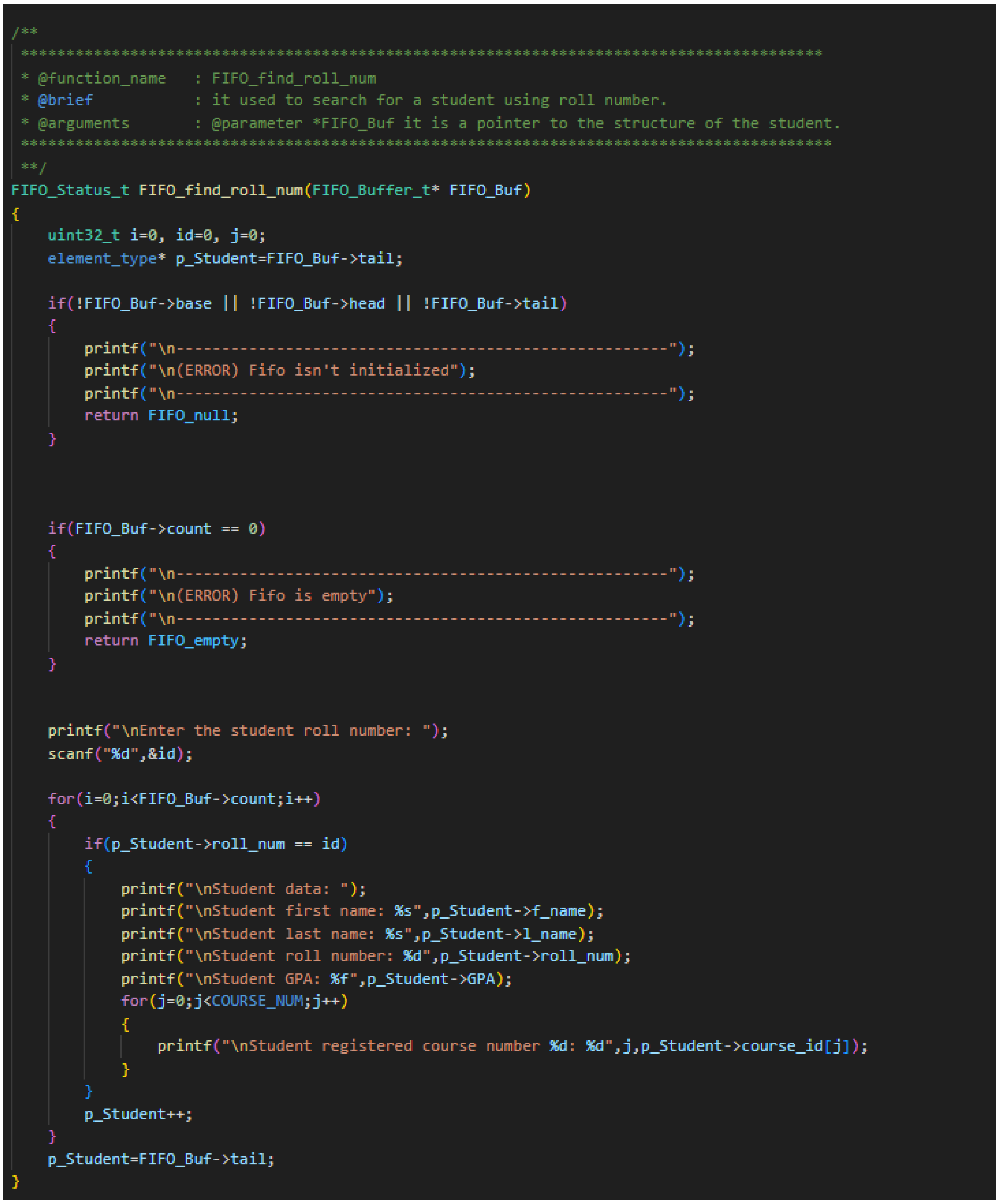
```c
FIFO_Status_t FIFO_add_student_file(FIFO_Buffer_t* FIFO_Buf)
{
    int i=0, flag=0;
    element_type* p_Student=FIFO_Buf->tail;
    Student_t New_Student;
    FILE *P_File = fopen("file.txt", "r+");

    if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n----------------------------------------------------");
        printf("(ERROR) Fifo isn't initialized");
        printf("\n----------------------------------------------------");
        return FIFO_null;
    }


    if(FIFO_IS_FULL(FIFO_Buf) == FIFO_full)
    {
        printf("\n----------------------------------------------------");
        printf("(ERROR) Fifo is full");
        printf("\n----------------------------------------------------");
        return FIFO_full;
    }


    if(P_File)
    {
        //loop till the end of the file.
        while(!feof(P_File))
        {
            //scan values from the text file
            fscanf(P_File,"%s %s %d %f %d %d %d %d %d\n",New_Student.f_name, New_Student.l_name, &New_Student.roll_num, &New_Student.GPA,
                    &New_Student.course_id[0], &New_Student.course_id[1], &New_Student.course_id[2],
                    &New_Student.course_id[3], &New_Student.course_id[4]);

            //check if the id is exist in database or not
            for(i=0;i<FIFO_Buf->count;i++)
            {
                if(New_Student.roll_num == p_Student->roll_num)
                flag=1;
            }
            if(flag==0)
            {
                strcpy(FIFO_Buf->head->f_name,New_Student.f_name);
                strcpy(FIFO_Buf->head->l_name,New_Student.l_name);
                FIFO_Buf->head->roll_num= New_Student.roll_num;
                FIFO_Buf->head->GPA = New_Student.GPA;
                for(int i = 0; i < COURSE_NUM; i++)
                {
                    FIFO_Buf->head->course_id[i] = New_Student.course_id[i];
                }
                FIFO_Buf->count++;
                //check if the head at the last element in the array.
                if(FIFO_Buf->head == (FIFO_Buf->base + (FIFO_Buf->length * sizeof(element_type))))
                    FIFO_Buf->head = FIFO_Buf->base;
                else
                        FIFO_Buf->head++;

                printf("\n------------------------------------------------------------");
                printf("\n(INFO) Student with roll number (%d) is added successfully",New_Student.roll_num);
                printf("\n------------------------------------------------------------");
            }
            else
            {
                printf("\n-----------------------------------------------------");
                printf("(ERROR) Roll number already exists");
                printf("\n-----------------------------------------------------");
            }
        }
    }
    else
    {
        printf("\n----------------------------------------------------");
        printf("(ERROR) file.txt File Not Found.\n");
        printf("\n----------------------------------------------------");
    }

}
```

FIFO_add_student



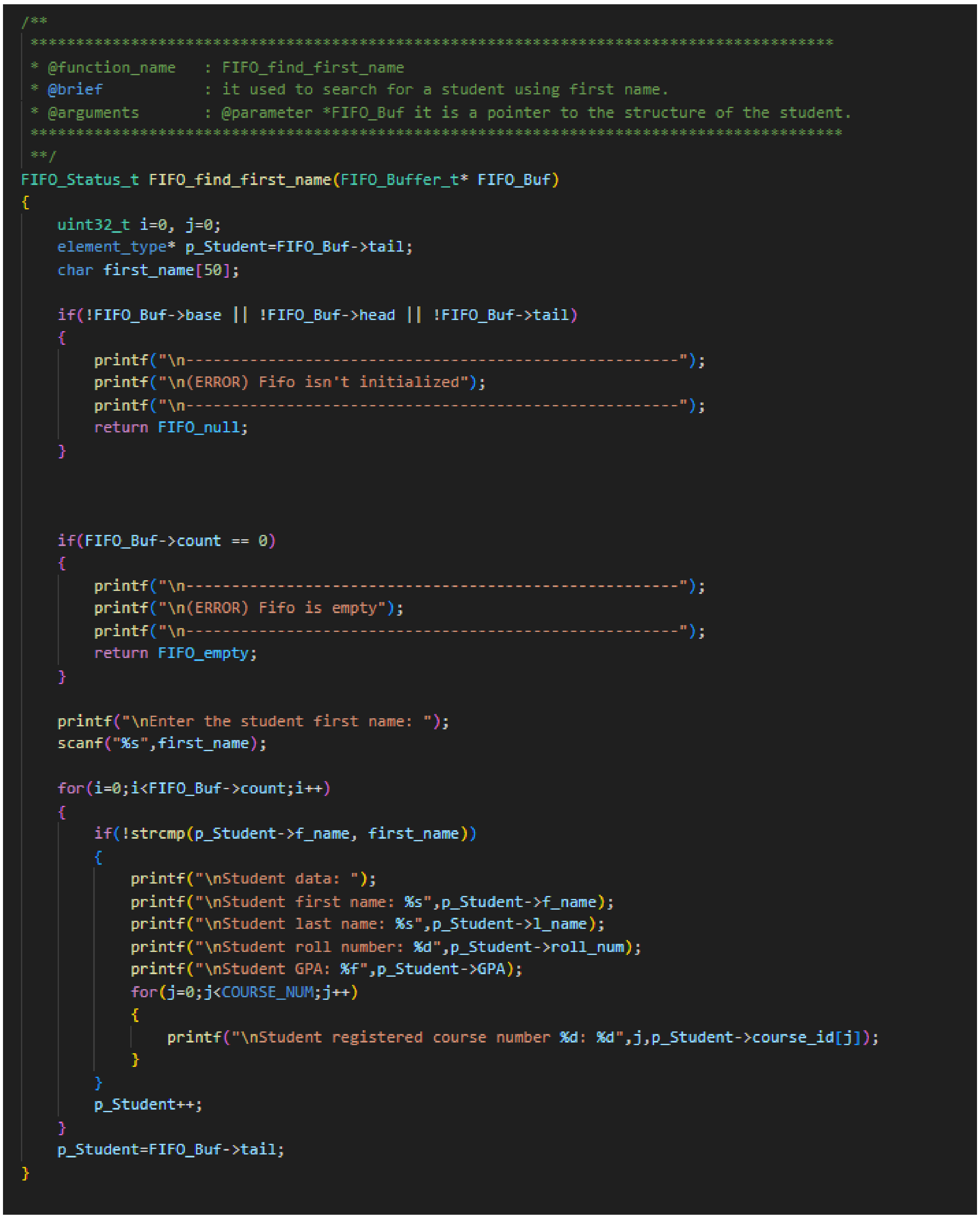
```c
FIFO_Status_t FIFO_add_student(FIFO_Buffer_t* FIFO_Buf)
{
    int i=0, id=0, flag=0;
    element_type* p_Student=FIFO_Buf->tail;

    if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n-----------------------------------------------------");
        printf("(ERROR) Fifo isn't initialized");
        printf("\n-----------------------------------------------------");
        return FIFO_null;
    }


    if(FIFO_IS_FULL(FIFO_Buf) == FIFO_full)
    {
        printf("\n-----------------------------------------------------");
        printf("(ERROR) Fifo is full");
        printf("\n-----------------------------------------------------");
        return FIFO_full;
    }

    printf("\nEnter the student roll number: ");
    scanf("%d",&id);

    for(i=0;i<FIFO_Buf->count;i++)
    {
        if(id == p_Student->roll_num)
            flag=1;
    }

    if(flag==0)
    {

        FIFO_Buf->head->roll_num=id;

        printf("\nEnter the student first name: ");
        scanf("%s",FIFO_Buf->head->f_name);

        printf("\nEnter the student last name: ");
        scanf("%s",FIFO_Buf->head->l_name);

        printf("\nEner the student GPA: ");
        scanf("%f",&FIFO_Buf->head->GPA);

        printf("\nEnter the student course IDs ");
        for(i=0;i<COURSE_NUM;i++)
        {
            printf("\nEnter course number %d ID: ",i+1);
            scanf("%d",&FIFO_Buf->head->course_id[i]);
        }


        printf("\n-----------------------------------------------------------");
        printf("\n(INFO) Student with roll number (%d) is added successfully",FIFO_Buf->head->roll_num);
        printf("\n-----------------------------------------------------------");

        //Circular Queue
        if(FIFO_Buf->head == (FIFO_Buf->base + (FIFO_Buf->length*sizeof(element_type))))
            FIFO_Buf->head=FIFO_Buf->base;
        else
            FIFO_Buf->head++;

        FIFO_Buf->count++;

    }
    else
    {
        printf("\n-----------------------------------------------------");
        printf("\n(ERROR) Roll number (%d) already exists",id);
        printf("\n-----------------------------------------------------");
        return ID_EXISTS;
    }


    return FIFO_no_error;
}
```

FIFO_find_roll_num



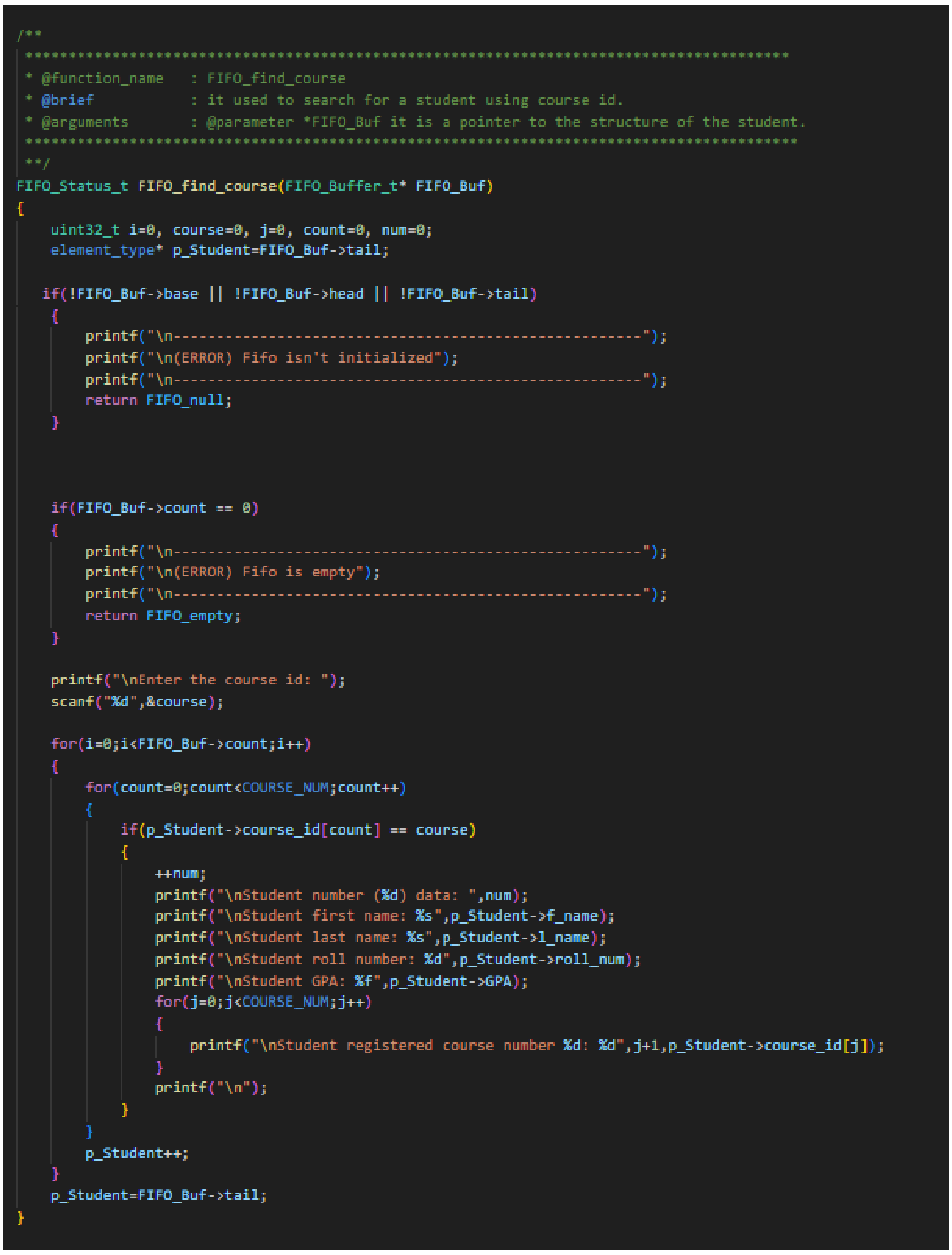
```c
/**
 ****************************************************************************************
 * @function_name   : FIFO_find_roll_num
 * @brief           : it used to search for a student using roll number.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student.
 *****************************************************************************************
 **/
FIFO_Status_t FIFO_find_roll_num(FIFO_Buffer_t* FIFO_Buf)
{
    uint32_t i=0, id=0, j=0;
    element_type* p_Student=FIFO_Buf->tail;

    if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n-------------------------------------------------------");
        printf("\n(ERROR) Fifo isn't initialized");
        printf("\n-------------------------------------------------------");
        return FIFO_null;
    }



    if(FIFO_Buf->count == 0)
    {
        printf("\n-------------------------------------------------------");
        printf("\n(ERROR) Fifo is empty");
        printf("\n-------------------------------------------------------");
        return FIFO_empty;
    }


    printf("\nEnter the student roll number: ");
    scanf("%d",&id);

    for(i=0;i<FIFO_Buf->count;i++)
    {
        if(p_Student->roll_num == id)
        {
            printf("\nStudent data: ");
            printf("\nStudent first name: %s",p_Student->f_name);
            printf("\nStudent last name: %s",p_Student->l_name);
            printf("\nStudent roll number: %d",p_Student->roll_num);
            printf("\nStudent GPA: %f",p_Student->GPA);
            for(j=0;j<COURSE_NUM;j++)
            {
                printf("\nStudent registered course number %d: %d",j,p_Student->course_id[j]);
            }
        }
        p_Student++;
    }
    p_Student=FIFO_Buf->tail;
}
```

FIFO_find_first_name



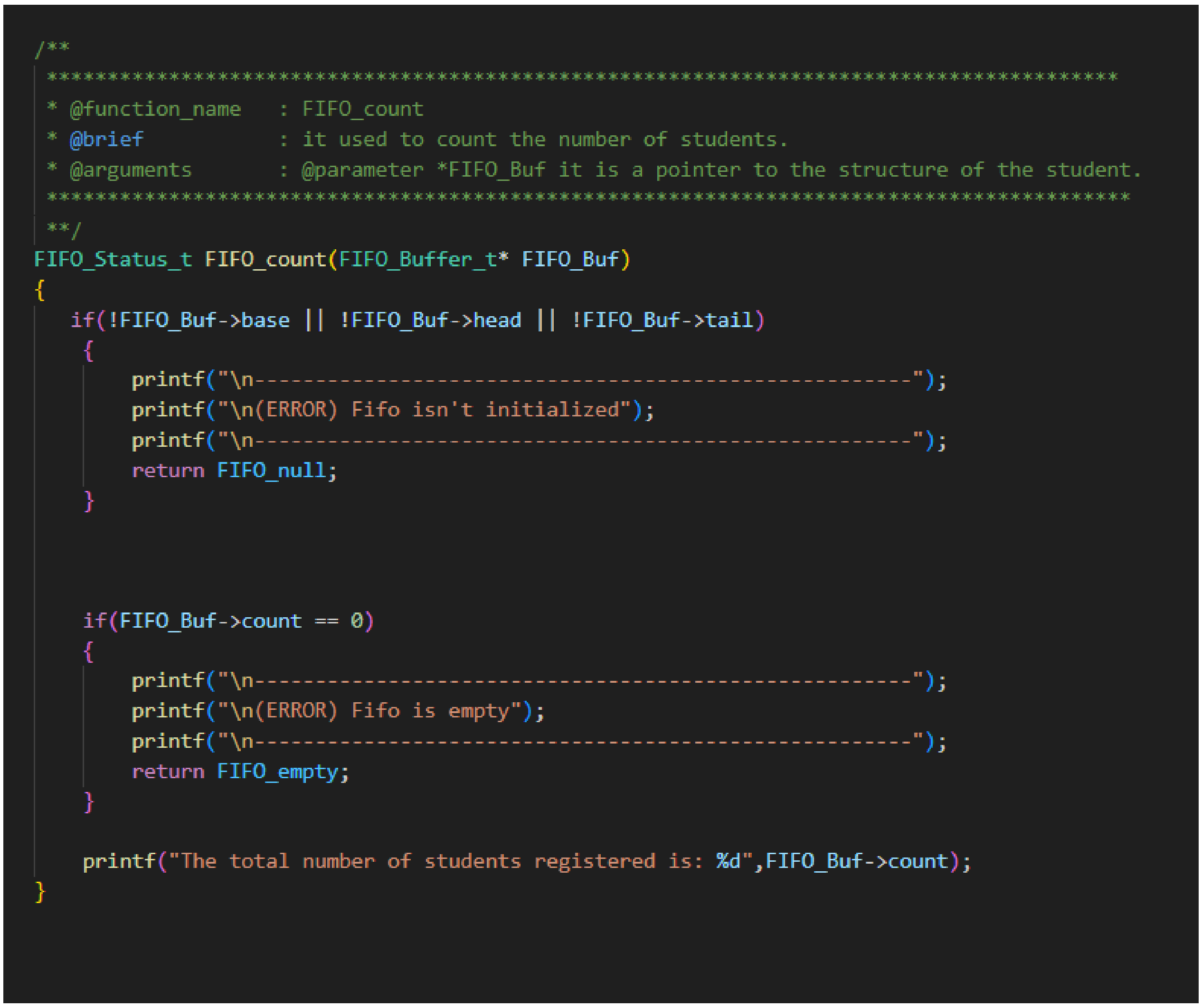
```c
/**
 ****************************************************************************************
 * @function_name   : FIFO_find_first_name
 * @brief           : it used to search for a student using first name.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student.
 *****************************************************************************************
 **/
FIFO_Status_t FIFO_find_first_name(FIFO_Buffer_t* FIFO_Buf)
{
    uint32_t i=0, j=0;
    element_type* p_Student=FIFO_Buf->tail;
    char first_name[50];

    if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n------------------------------------------------------");
        printf("\n(ERROR) Fifo isn't initialized");
        printf("\n------------------------------------------------------");
        return FIFO_null;
    }



    if(FIFO_Buf->count == 0)
    {
        printf("\n------------------------------------------------------");
        printf("\n(ERROR) Fifo is empty");
        printf("\n------------------------------------------------------");
        return FIFO_empty;
    }

    printf("\nEnter the student first name: ");
    scanf("%s",first_name);

    for(i=0;i<FIFO_Buf->count;i++)
    {
        if(!strcmp(p_Student->f_name, first_name))
        {
            printf("\nStudent data: ");
            printf("\nStudent first name: %s",p_Student->f_name);
            printf("\nStudent last name: %s",p_Student->l_name);
            printf("\nStudent roll number: %d",p_Student->roll_num);
            printf("\nStudent GPA: %f",p_Student->GPA);
            for(j=0;j<COURSE_NUM;j++)
            {
                printf("\nStudent registered course number %d: %d",j,p_Student->course_id[j]);
            }
        }
        p_Student++;
    }
    p_Student=FIFO_Buf->tail;
}
```

FIFO_find_course



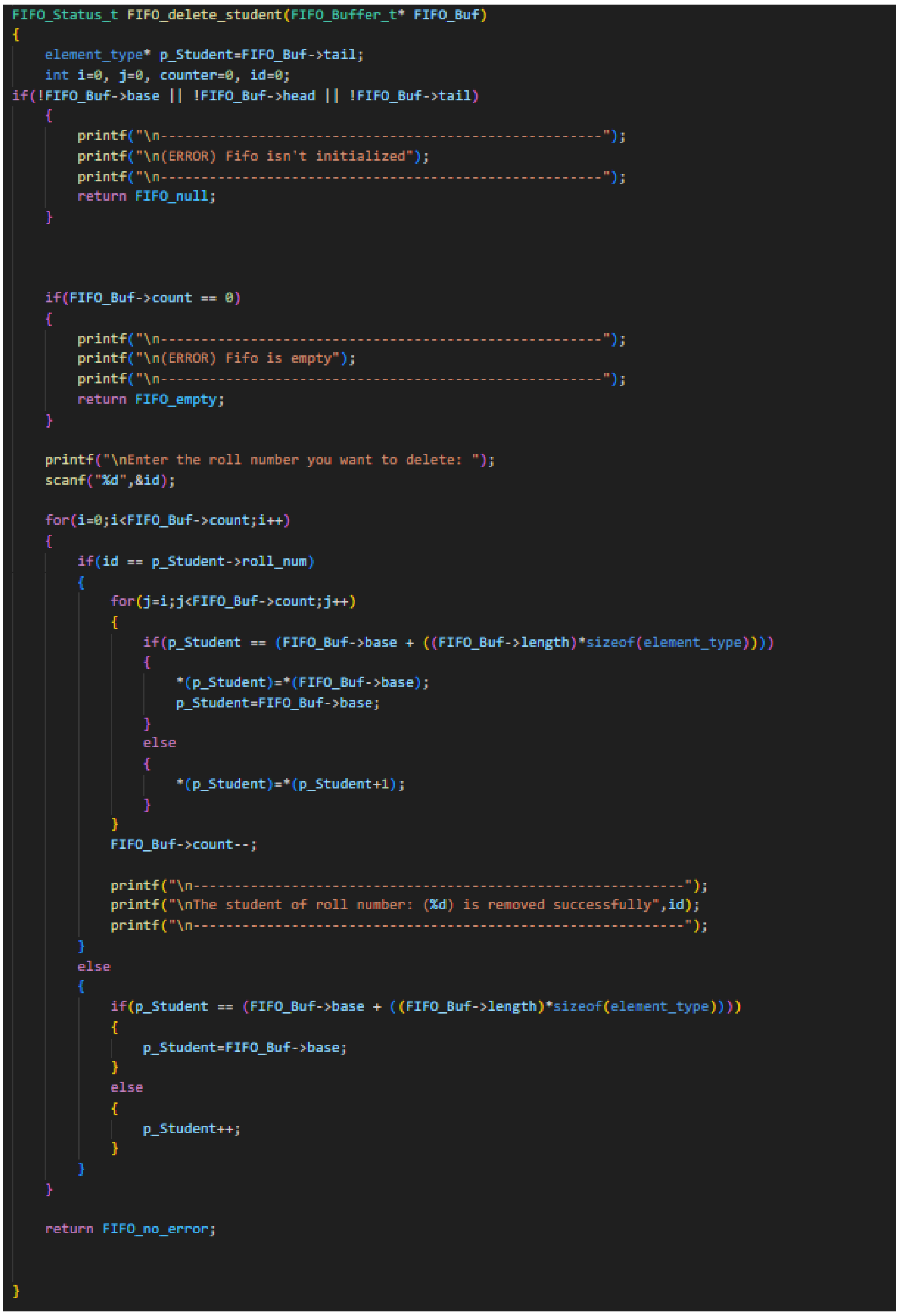
```c
/**
 *******************************************************************************************
 * @function_name   : FIFO_find_course
 * @brief           : it used to search for a student using course id.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student.
 *******************************************************************************************
 **/
FIFO_Status_t FIFO_find_course(FIFO_Buffer_t* FIFO_Buf)
{
    uint32_t i=0, course=0, j=0, count=0, num=0;
    element_type* p_Student=FIFO_Buf->tail;

   if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n-----------------------------------------------------");
        printf("\n(ERROR) Fifo isn't initialized");
        printf("\n-----------------------------------------------------");
        return FIFO_null;
    }



    if(FIFO_Buf->count == 0)
    {
        printf("\n-----------------------------------------------------");
        printf("\n(ERROR) Fifo is empty");
        printf("\n-----------------------------------------------------");
        return FIFO_empty;
    }

    printf("\nEnter the course id: ");
    scanf("%d",&course);

    for(i=0;i<FIFO_Buf->count;i++)
    {
        for(count=0;count<COURSE_NUM;count++)
        {
            if(p_Student->course_id[count] == course)
            {
                ++num;
                printf("\nStudent number (%d) data: ",num);
                printf("\nStudent first name: %s",p_Student->f_name);
                printf("\nStudent last name: %s",p_Student->l_name);
                printf("\nStudent roll number: %d",p_Student->roll_num);
                printf("\nStudent GPA: %f",p_Student->GPA);
                for(j=0;j<COURSE_NUM;j++)
                {
                    printf("\nStudent registered course number %d: %d",j+1,p_Student->course_id[j]);
                }
                printf("\n");
            }
        }
        p_Student++;
    }
    p_Student=FIFO_Buf->tail;
}
```

FIFO_count



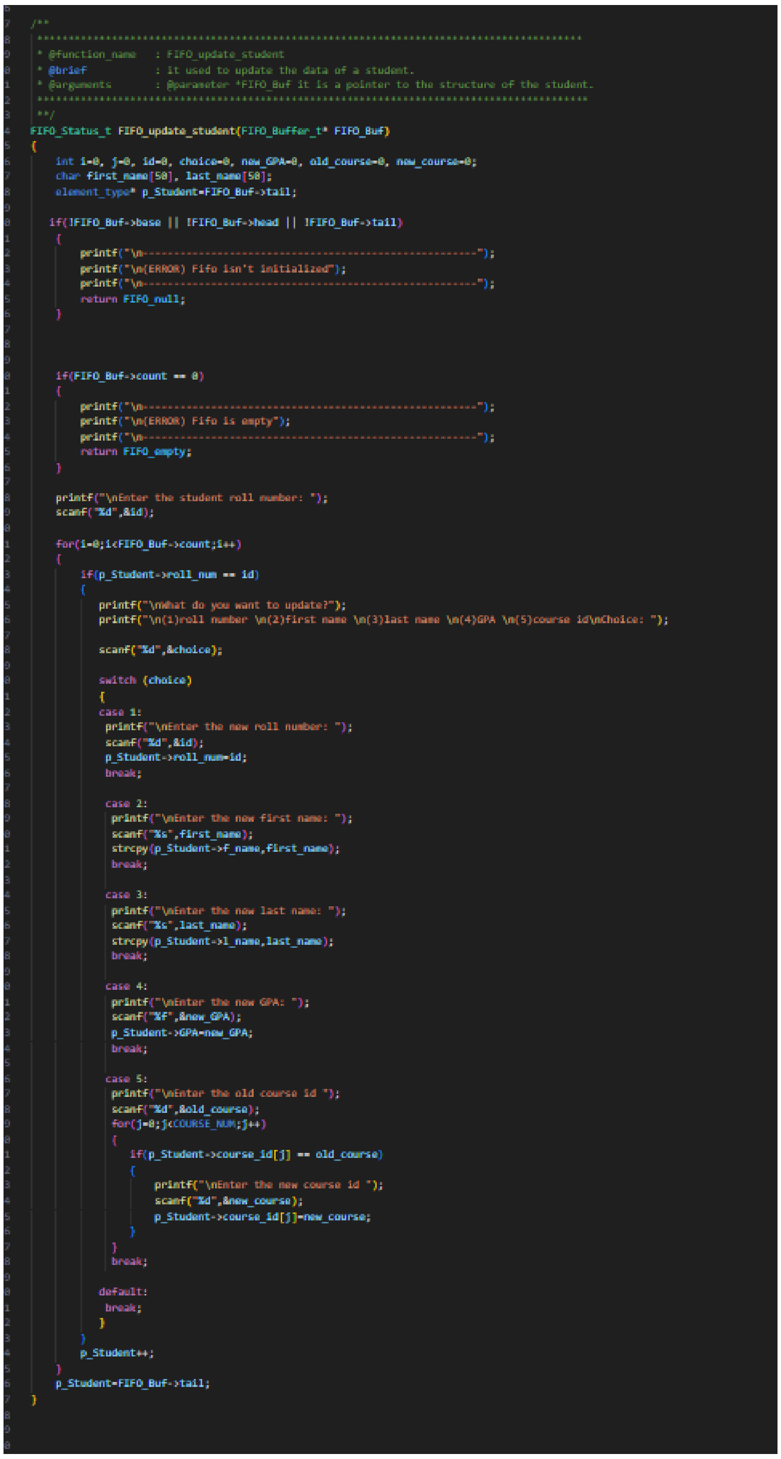
```c
/**
 *************************************************************************************
 * @function_name   : FIFO_count
 * @brief           : it used to count the number of students.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student.
 **************************************************************************************
 **/
FIFO_Status_t FIFO_count(FIFO_Buffer_t* FIFO_Buf)
{
   if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n------------------------------------------------------");
        printf("\n(ERROR) Fifo isn't initialized");
        printf("\n------------------------------------------------------");
        return FIFO_null;
    }



    if(FIFO_Buf->count == 0)
    {
        printf("\n------------------------------------------------------");
        printf("\n(ERROR) Fifo is empty");
        printf("\n------------------------------------------------------");
        return FIFO_empty;
    }

    printf("The total number of students registered is: %d",FIFO_Buf->count);
}
```

FIFO_ delete_student



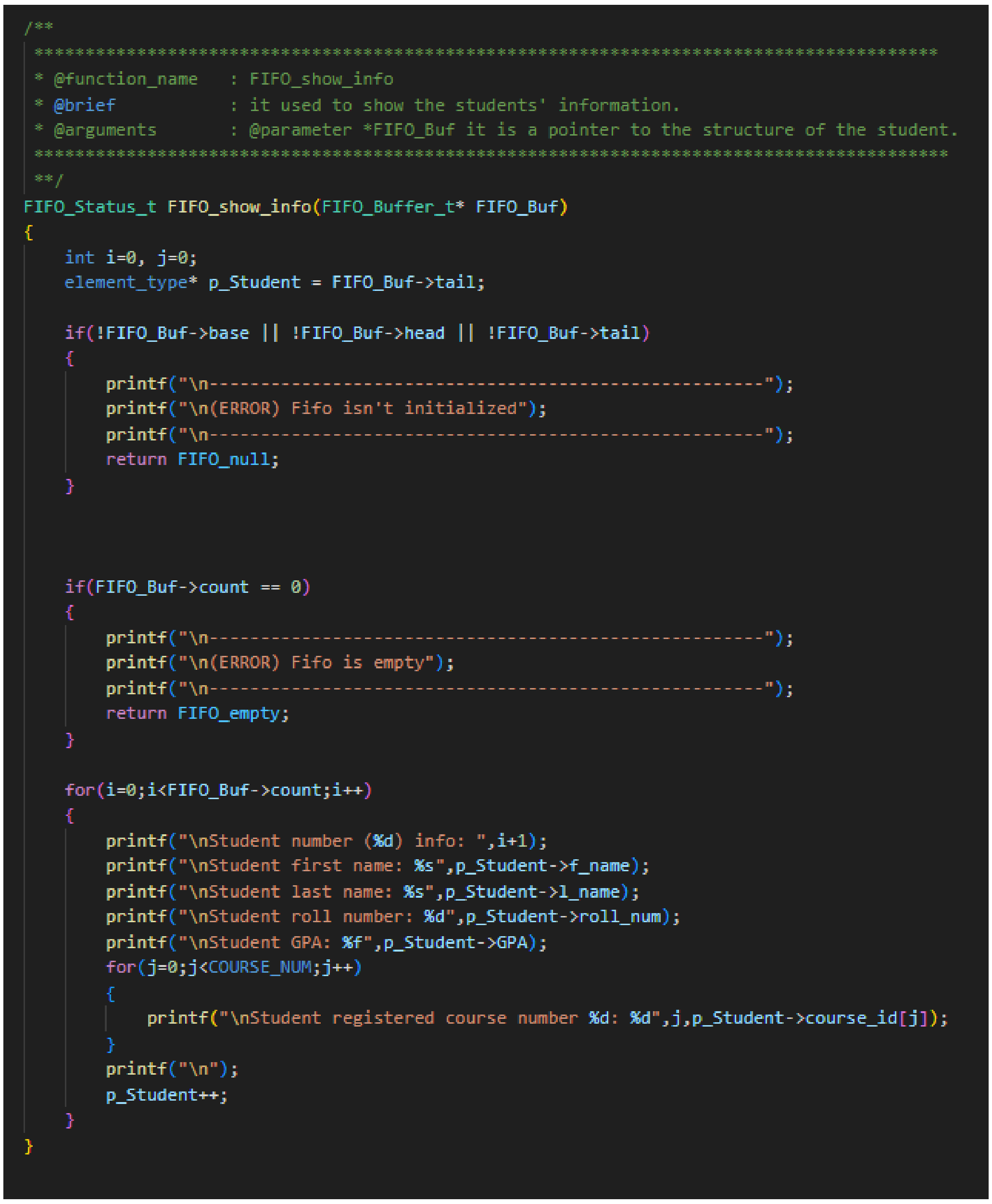
```
FIFO_Status_t FIFO_delete_student(FIFO_Buffer_t* FIFO_Buf)
{
    element_type* p_Student=FIFO_Buf->tail;
    int i=0, j=0, counter=0, id=0;
if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n------------------------------------------------------");
        printf("\n(ERROR) Fifo isn't initialized");
        printf("\n------------------------------------------------------");
        return FIFO_null;
    }



    if(FIFO_Buf->count == 0)
    {
        printf("\n------------------------------------------------------");
        printf("\n(ERROR) Fifo is empty");
        printf("\n------------------------------------------------------");
        return FIFO_empty;
    }

    printf("\nEnter the roll number you want to delete: ");
    scanf("%d",&id);

    for(i=0;i<FIFO_Buf->count;i++)
    {
        if(id == p_Student->roll_num)
        {
            for(j=i;j<FIFO_Buf->count;j++)
            {
                if(p_Student == (FIFO_Buf->base + ((FIFO_Buf->length)*sizeof(element_type))))
                {
                    *(p_Student)=*(FIFO_Buf->base);
                    p_Student=FIFO_Buf->base;
                }
                else
                {
                    *(p_Student)=*(p_Student+1);
                }
            }
            FIFO_Buf->count--;

            printf("\n------------------------------------------------------------");
            printf("\nThe student of roll number: (%d) is removed successfully",id);
            printf("\n------------------------------------------------------------");
        }
        else
        {
            if(p_Student == (FIFO_Buf->base + ((FIFO_Buf->length)*sizeof(element_type))))
            {
                p_Student=FIFO_Buf->base;
            }
            else
            {
                p_Student++;
            }
        }
    }

    return FIFO_no_error;


}
```

FIFO_ update_student



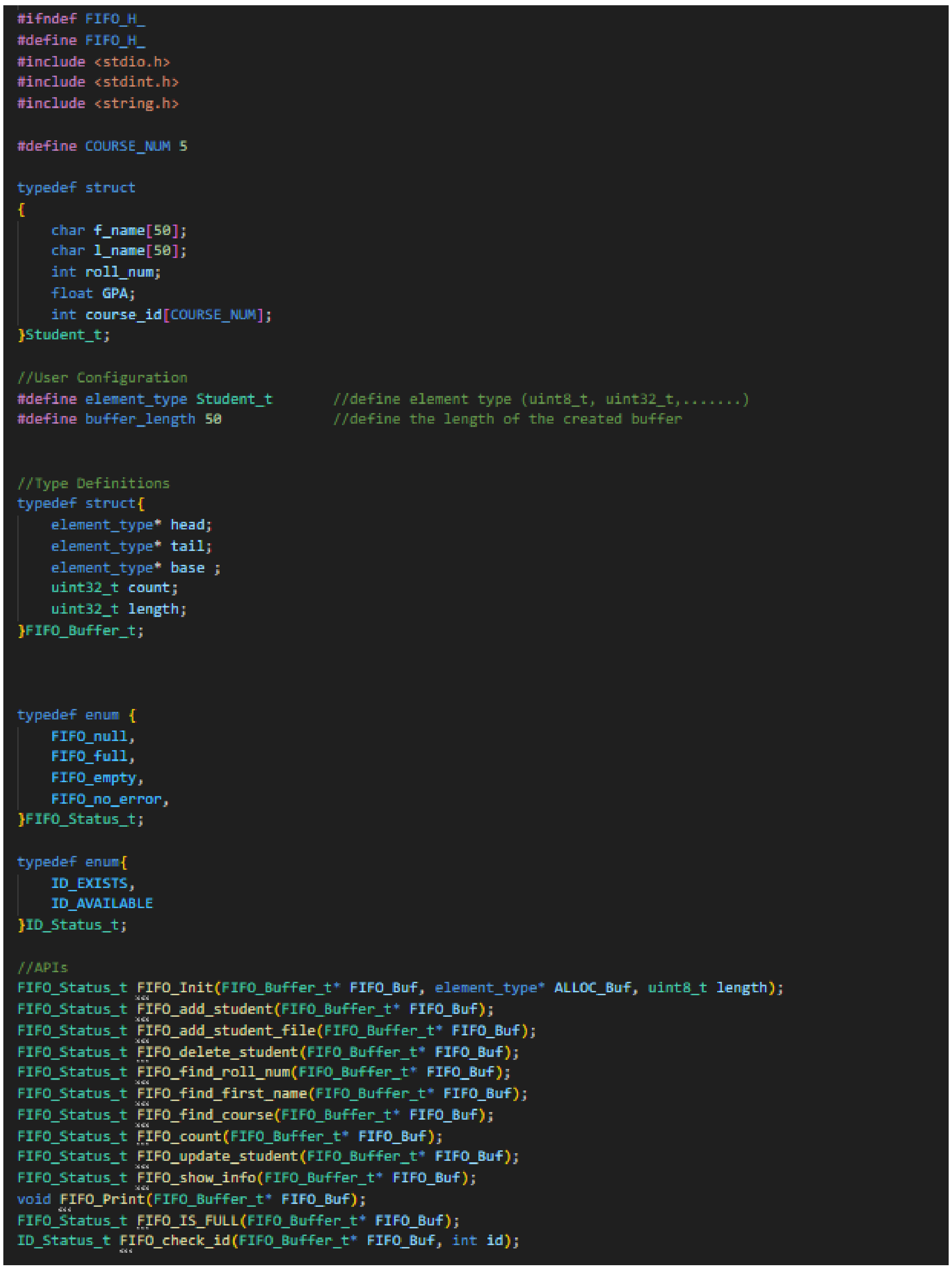
```c
/**
 ****************************************************************************************
 * @function_name   : FIFO_update_student
 * @brief           : it used to update the data of a student.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student.
 ****************************************************************************************
 **/
FIFO_Status_t FIFO_update_student(FIFO_Buffer_t* FIFO_Buf)
{
    int i=0, j=0, id=0, choice=0, new_GPA=0, old_course=0, new_course=0;
    char first_name[50], last_name[50];
    element_type* p_Student=FIFO_Buf->tail;

   if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n-------------------------------------------------------");
        printf("\n(ERROR) Fifo isn't initialized");
        printf("\n-------------------------------------------------------");
        return FIFO_null;
    }



    if(FIFO_Buf->count == 0)
    {
        printf("\n-------------------------------------------------------");
        printf("\n(ERROR) Fifo is empty");
        printf("\n-------------------------------------------------------");
        return FIFO_empty;
    }

    printf("\nEnter the student roll number: ");
    scanf("%d",&id);

    for(i=0;i<FIFO_Buf->count;i++)
    {
        if(p_Student->roll_num == id)
        {
           printf("\nWhat do you want to update?");
           printf("\n(1)roll number \n(2)first name \n(3)last name \n(4)GPA \n(5)course id\nChoice: ");

           scanf("%d",&choice);

           switch (choice)
           {
           case 1:
            printf("\nEnter the new roll number: ");
            scanf("%d",&id);
            p_Student->roll_num=id;
            break;

            case 2:
             printf("\nEnter the new first name: ");
             scanf("%s",first_name);
             strcpy(p_Student->f_name,first_name);
             break;

            case 3:
             printf("\nEnter the new last name: ");
             scanf("%s",last_name);
             strcpy(p_Student->l_name,last_name);
             break;

            case 4:
             printf("\nEnter the new GPA: ");
             scanf("%f",&new_GPA);
             p_Student->GPA=new_GPA;
             break;

            case 5:
             printf("\nEnter the old course id ");
             scanf("%d",&old_course);
             for(j=0;j<COURSE_NUM;j++)
             {
                if(p_Student->course_id[j] == old_course)
                {
                    printf("\nEnter the new course id ");
                    scanf("%d",&new_course);
                    p_Student->course_id[j]=new_course;
                }
             }
             break;

           default:
            break;
           }
        }
        p_Student++;
    }
    p_Student=FIFO_Buf->tail;
}
```

FIFO_ show_info



```c
/**
 ****************************************************************************************
 * @function_name   : FIFO_show_info
 * @brief           : it used to show the students' information.
 * @arguments       : @parameter *FIFO_Buf it is a pointer to the structure of the student.
 *****************************************************************************************
 **/
FIFO_Status_t FIFO_show_info(FIFO_Buffer_t* FIFO_Buf)
{
    int i=0, j=0;
    element_type* p_Student = FIFO_Buf->tail;

    if(!FIFO_Buf->base || !FIFO_Buf->head || !FIFO_Buf->tail)
    {
        printf("\n-------------------------------------------------------");
        printf("\n(ERROR) Fifo isn't initialized");
        printf("\n-------------------------------------------------------");
        return FIFO_null;
    }



    if(FIFO_Buf->count == 0)
    {
        printf("\n-------------------------------------------------------");
        printf("\n(ERROR) Fifo is empty");
        printf("\n-------------------------------------------------------");
        return FIFO_empty;
    }

    for(i=0;i<FIFO_Buf->count;i++)
    {
        printf("\nStudent number (%d) info: ",i+1);
        printf("\nStudent first name: %s",p_Student->f_name);
        printf("\nStudent last name: %s",p_Student->l_name);
        printf("\nStudent roll number: %d",p_Student->roll_num);
        printf("\nStudent GPA: %f",p_Student->GPA);
        for(j=0;j<COURSE_NUM;j++)
        {
            printf("\nStudent registered course number %d: %d",j,p_Student->course_id[j]);
        }
        printf("\n");
        p_Student++;
    }
}
```

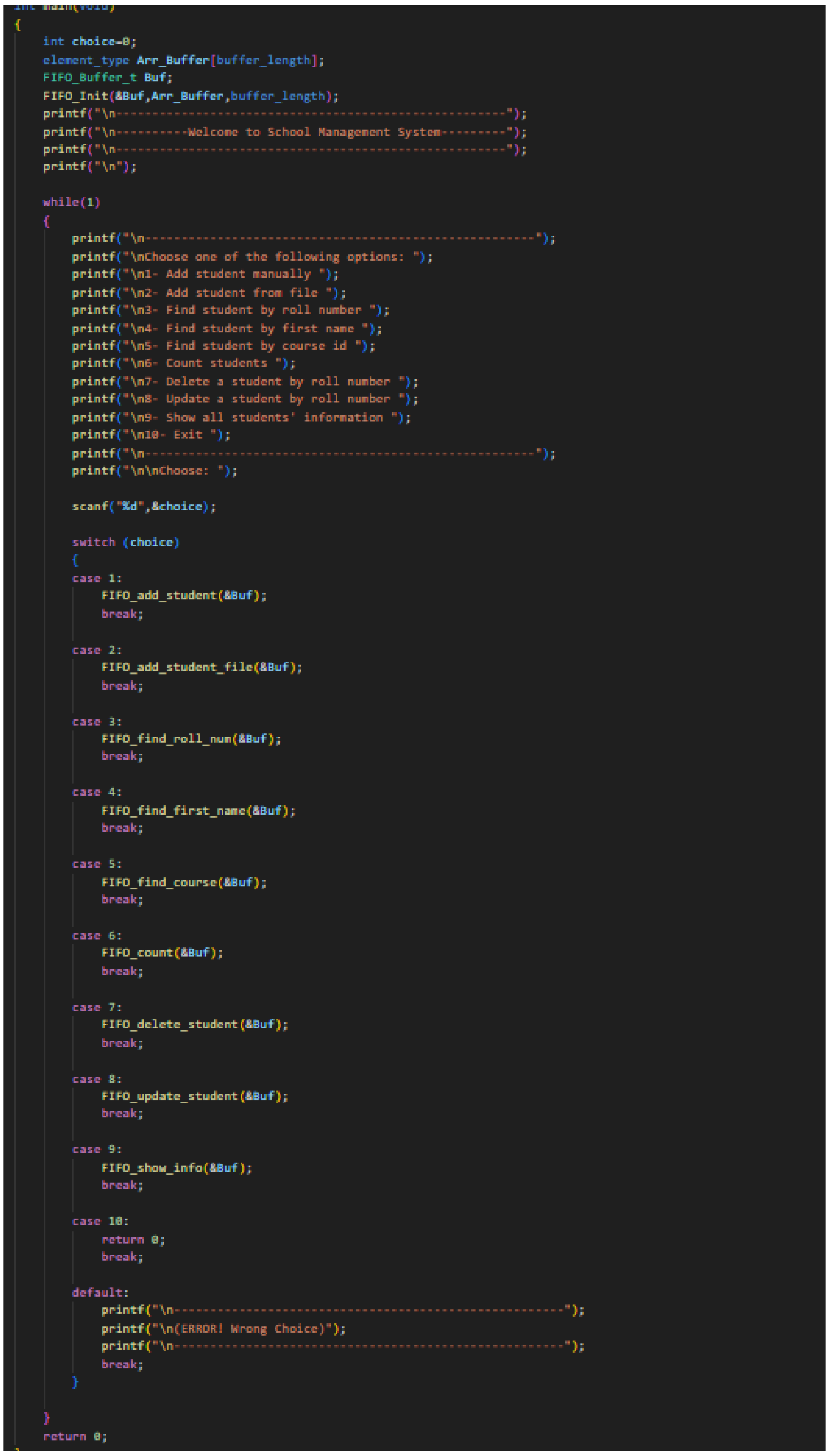
.h file



```c
#ifndef FIFO_H_
#define FIFO_H_
#include <stdio.h>
#include <stdint.h>
#include <string.h>

#define COURSE_NUM 5

typedef struct
{
    char f_name[50];
    char l_name[50];
    int roll_num;
    float GPA;
    int course_id[COURSE_NUM];
}Student_t;

//User Configuration
#define element_type Student_t       //define element type (uint8_t, uint32_t,.......)
#define buffer_length 50             //define the length of the created buffer


//Type Definitions
typedef struct{
    element_type* head;
    element_type* tail;
    element_type* base ;
    uint32_t count;
    uint32_t length;
}FIFO_Buffer_t;



typedef enum {
    FIFO_null,
    FIFO_full,
    FIFO_empty,
    FIFO_no_error,
}FIFO_Status_t;

typedef enum{
    ID_EXISTS,
    ID_AVAILABLE
}ID_Status_t;

//APIs
FIFO_Status_t FIFO_Init(FIFO_Buffer_t* FIFO_Buf, element_type* ALLOC_Buf, uint8_t length);
FIFO_Status_t FIFO_add_student(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_add_student_file(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_delete_student(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_find_roll_num(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_find_first_name(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_find_course(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_count(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_update_student(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_show_info(FIFO_Buffer_t* FIFO_Buf);
void FIFO_Print(FIFO_Buffer_t* FIFO_Buf);
FIFO_Status_t FIFO_IS_FULL(FIFO_Buffer_t* FIFO_Buf);
ID_Status_t FIFO_check_id(FIFO_Buffer_t* FIFO_Buf, int id);
```

Main

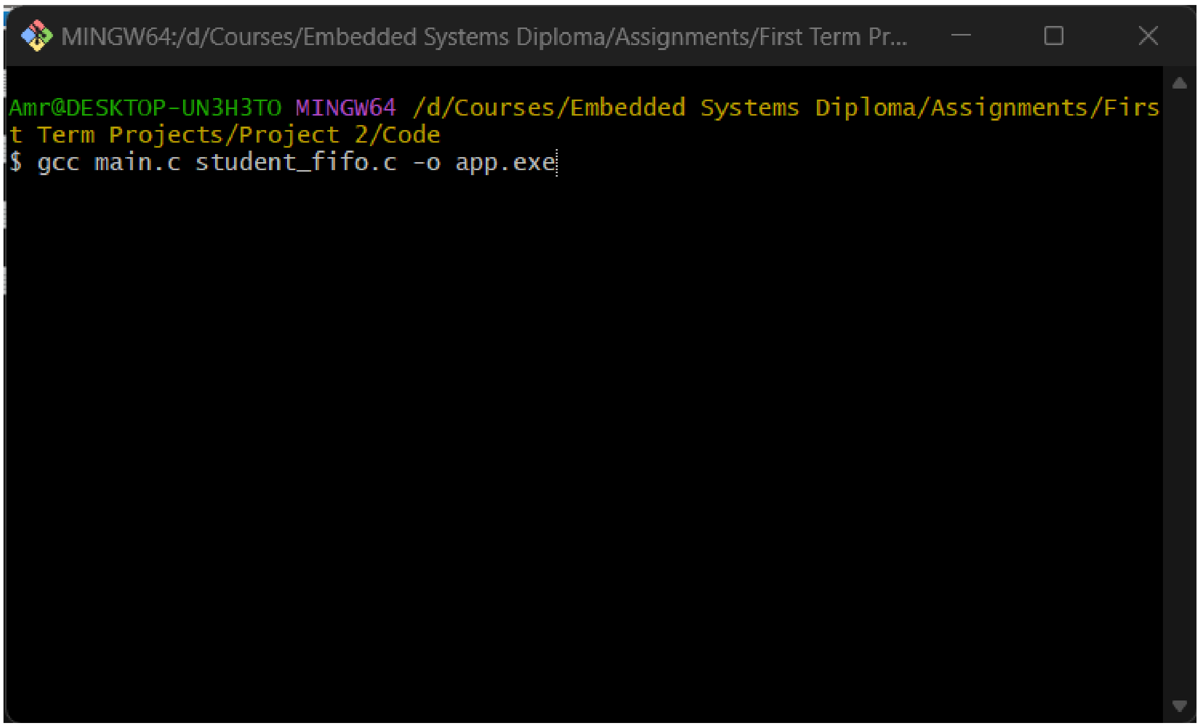
.c file



```c
int main(void)
{
    int choice=0;
    element_type Arr_Buffer[buffer_length];
    FIFO_Buffer_t Buf;
    FIFO_Init(&Buf,Arr_Buffer,buffer_length);
    printf("\n-----------------------------------------------------");
    printf("\n---------Welcome to School Management System---------");
    printf("\n-----------------------------------------------------");
    printf("\n");

    while(1)
    {
        printf("\n-----------------------------------------------------");
        printf("\nChoose one of the following options: ");
        printf("\n1- Add student manually ");
        printf("\n2- Add student from file ");
        printf("\n3- Find student by roll number ");
        printf("\n4- Find student by first name ");
        printf("\n5- Find student by course id ");
        printf("\n6- Count students ");
        printf("\n7- Delete a student by roll number ");
        printf("\n8- Update a student by roll number ");
        printf("\n9- Show all students' information ");
        printf("\n10- Exit ");
        printf("\n-----------------------------------------------------");
        printf("\n\nChoose: ");

        scanf("%d",&choice);

        switch (choice)
        {
        case 1:
            FIFO_add_student(&Buf);
            break;

        case 2:
            FIFO_add_student_file(&Buf);
            break;

        case 3:
            FIFO_find_roll_num(&Buf);
            break;

        case 4:
            FIFO_find_first_name(&Buf);
            break;

        case 5:
            FIFO_find_course(&Buf);
            break;

        case 6:
            FIFO_count(&Buf);
            break;

        case 7:
            FIFO_delete_student(&Buf);
            break;

        case 8:
            FIFO_update_student(&Buf);
            break;

        case 9:
            FIFO_show_info(&Buf);
            break;

        case 10:
            return 0;
            break;

        default:
            printf("\n-----------------------------------------------------");
            printf("\n(ERROR! Wrong Choice)");
            printf("\n-----------------------------------------------------");
            break;
        }

    }
    return 0;
```

Code Building Tools

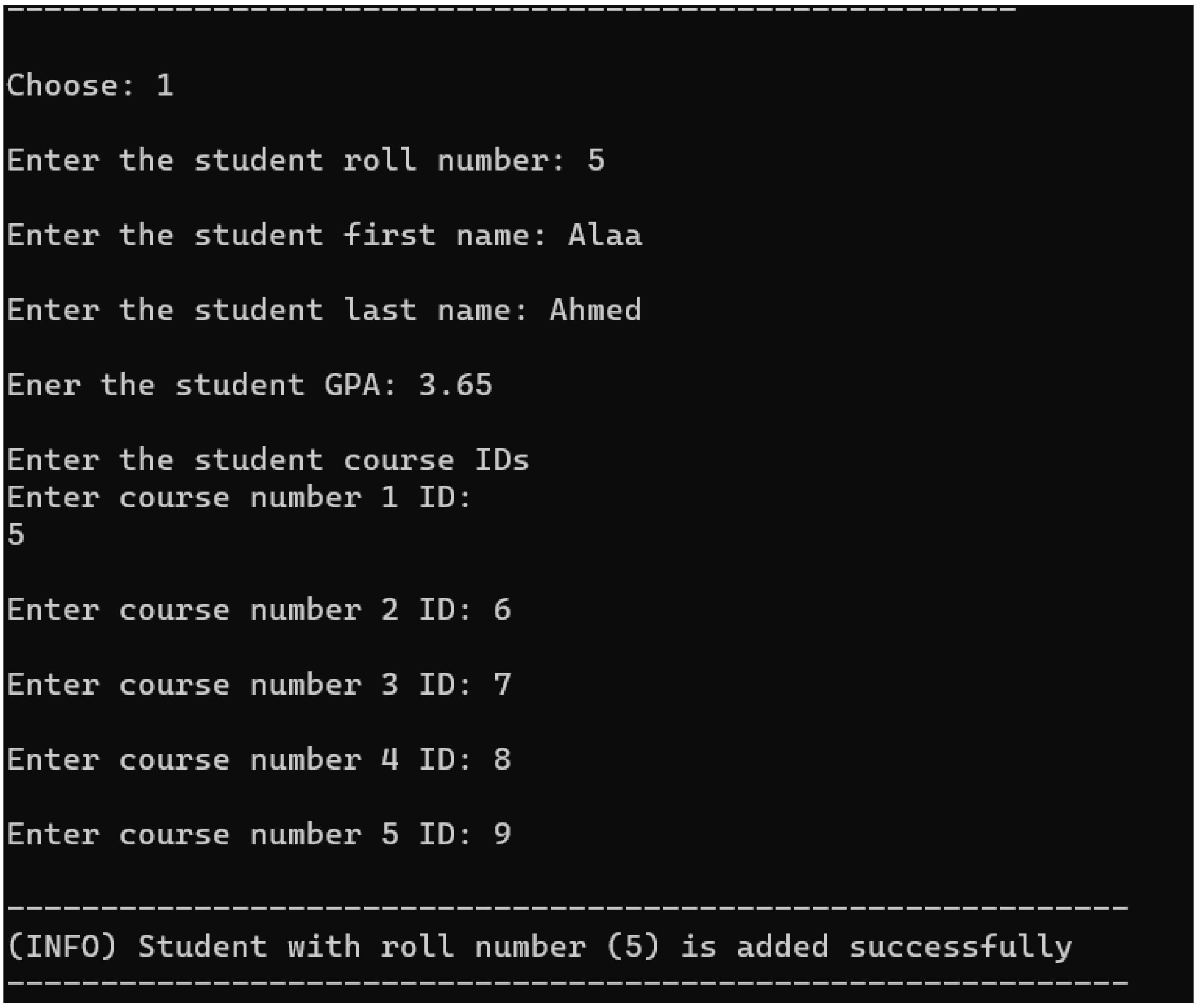
GCC Compiler


MINGW64:/d/Courses/Embedded Systems Diploma/Assignments/First Term Pr...

```
Amr@DESKTOP-UN3H3TO MINGW64 /d/Courses/Embedded Systems Diploma/Assignments/Firs
t Term Projects/Project 2/Code
$ gcc main.c student_fifo.c -o app.exe
```

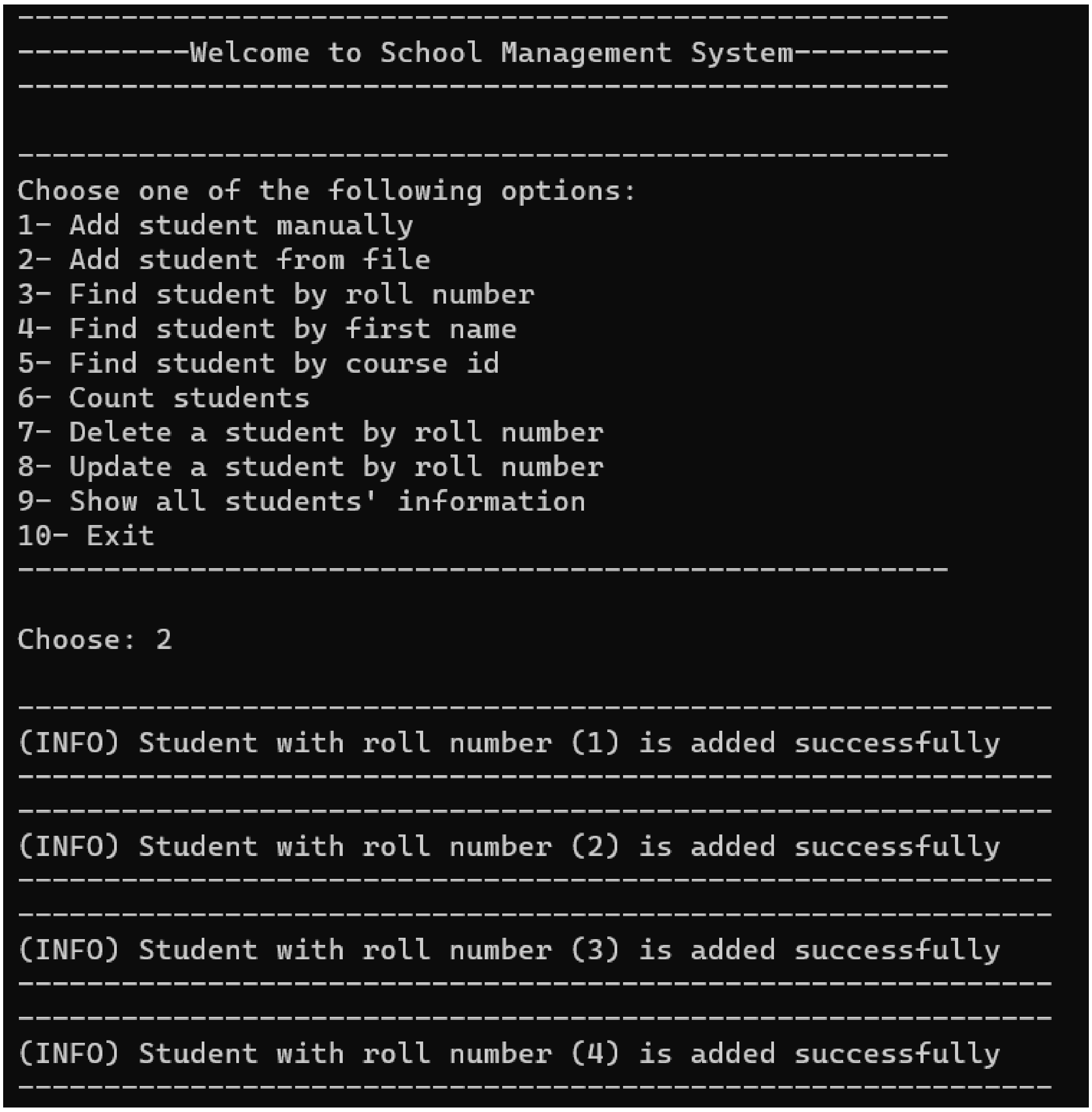
Working Code

1. Add students manually



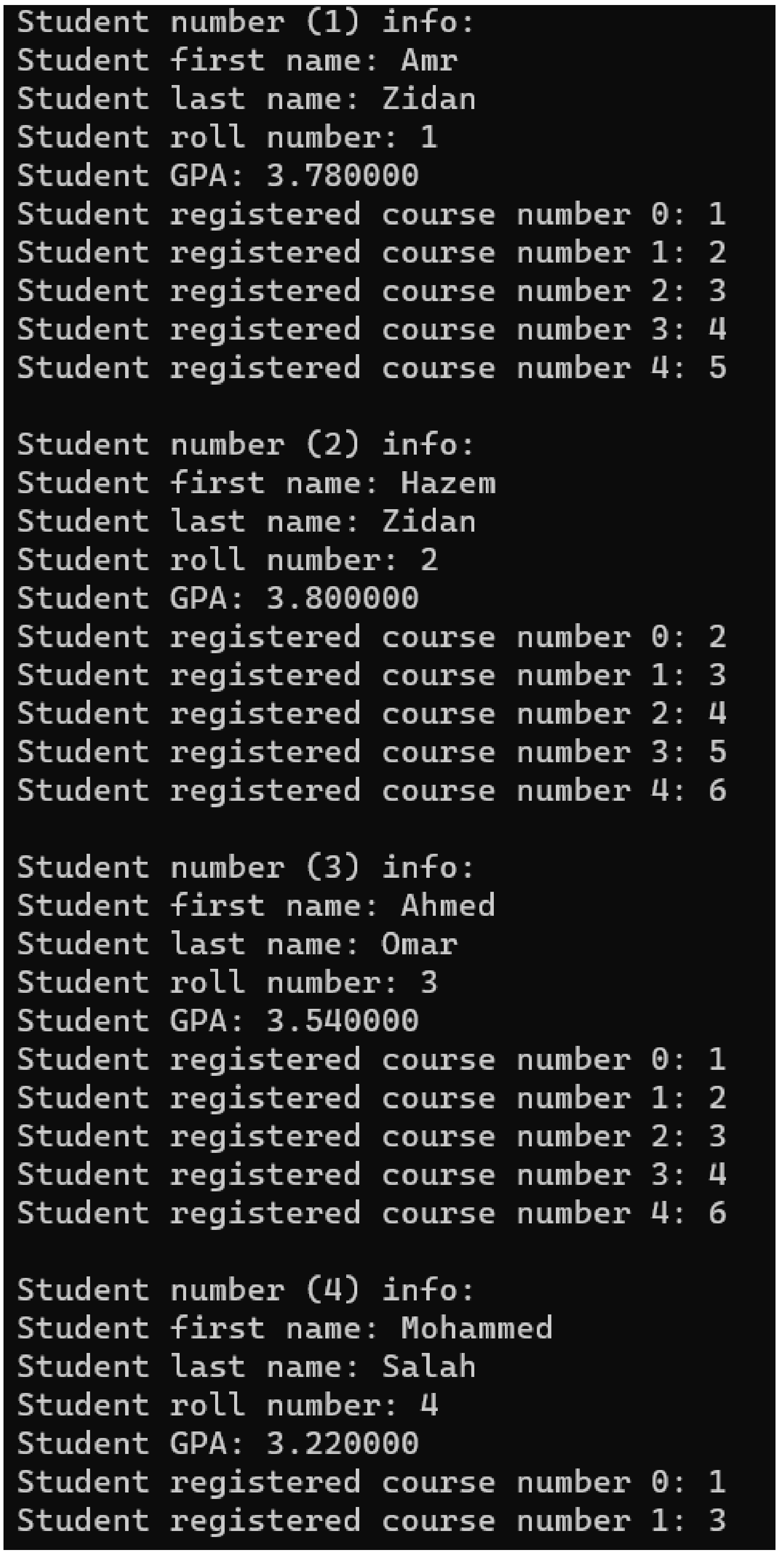
```
-------------------------------------------------------

Choose: 1

Enter the student roll number: 5

Enter the student first name: Alaa

Enter the student last name: Ahmed

Ener the student GPA: 3.65

Enter the student course IDs
Enter course number 1 ID:
5

Enter course number 2 ID: 6

Enter course number 3 ID: 7

Enter course number 4 ID: 8

Enter course number 5 ID: 9

------------------------------------------------------------
(INFO) Student with roll number (5) is added successfully
------------------------------------------------------------
```

2. Add students from file



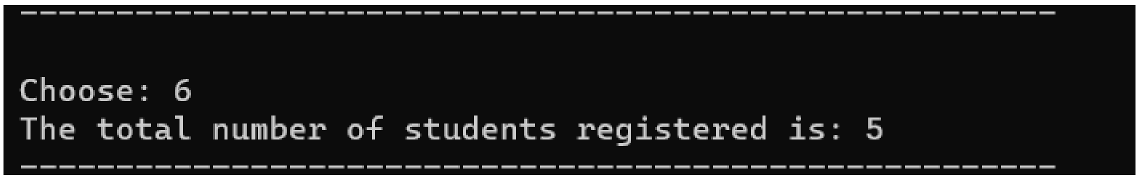
```
-------------------------------------------------------
----------Welcome to School Management System---------
-------------------------------------------------------


-------------------------------------------------------
Choose one of the following options:
1- Add student manually
2- Add student from file
3- Find student by roll number
4- Find student by first name
5- Find student by course id
6- Count students
7- Delete a student by roll number
8- Update a student by roll number
9- Show all students' information
10- Exit
-------------------------------------------------------

Choose: 2

-------------------------------------------------------------
(INFO) Student with roll number (1) is added successfully
-------------------------------------------------------------
-------------------------------------------------------------
(INFO) Student with roll number (2) is added successfully
-------------------------------------------------------------
-------------------------------------------------------------
(INFO) Student with roll number (3) is added successfully
-------------------------------------------------------------
-------------------------------------------------------------
(INFO) Student with roll number (4) is added successfully
-------------------------------------------------------------
```

3. Show students from added file



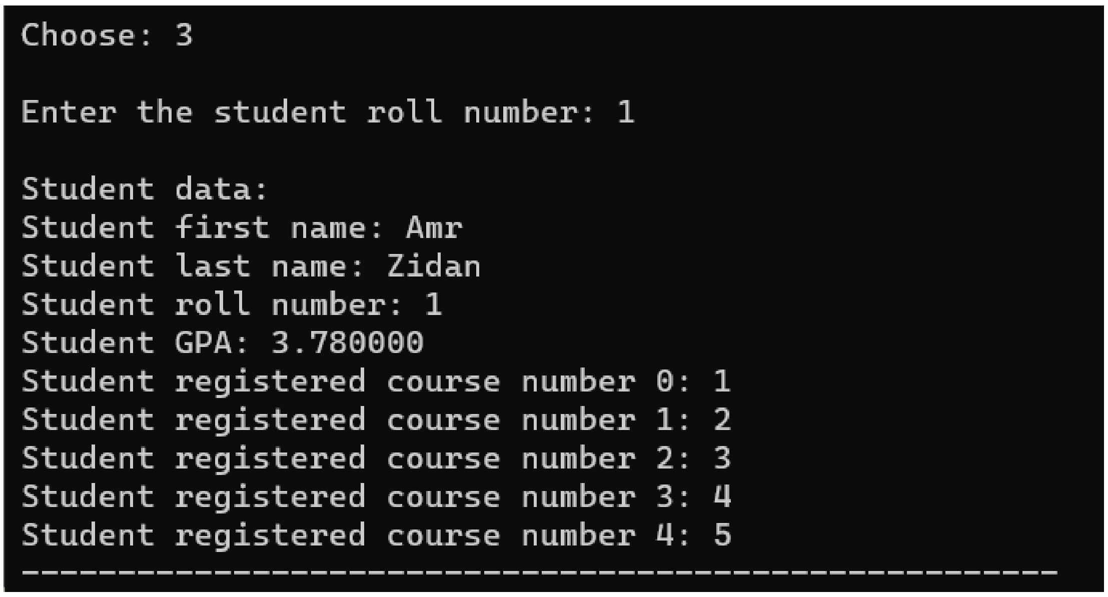
```
Student number (1) info:
Student first name: Amr
Student last name: Zidan
Student roll number: 1
Student GPA: 3.780000
Student registered course number 0: 1
Student registered course number 1: 2
Student registered course number 2: 3
Student registered course number 3: 4
Student registered course number 4: 5

Student number (2) info:
Student first name: Hazem
Student last name: Zidan
Student roll number: 2
Student GPA: 3.800000
Student registered course number 0: 2
Student registered course number 1: 3
Student registered course number 2: 4
Student registered course number 3: 5
Student registered course number 4: 6

Student number (3) info:
Student first name: Ahmed
Student last name: Omar
Student roll number: 3
Student GPA: 3.540000
Student registered course number 0: 1
Student registered course number 1: 2
Student registered course number 2: 3
Student registered course number 3: 4
Student registered course number 4: 6

Student number (4) info:
Student first name: Mohammed
Student last name: Salah
Student roll number: 4
Student GPA: 3.220000
Student registered course number 0: 1
Student registered course number 1: 3
```

4. Counting students



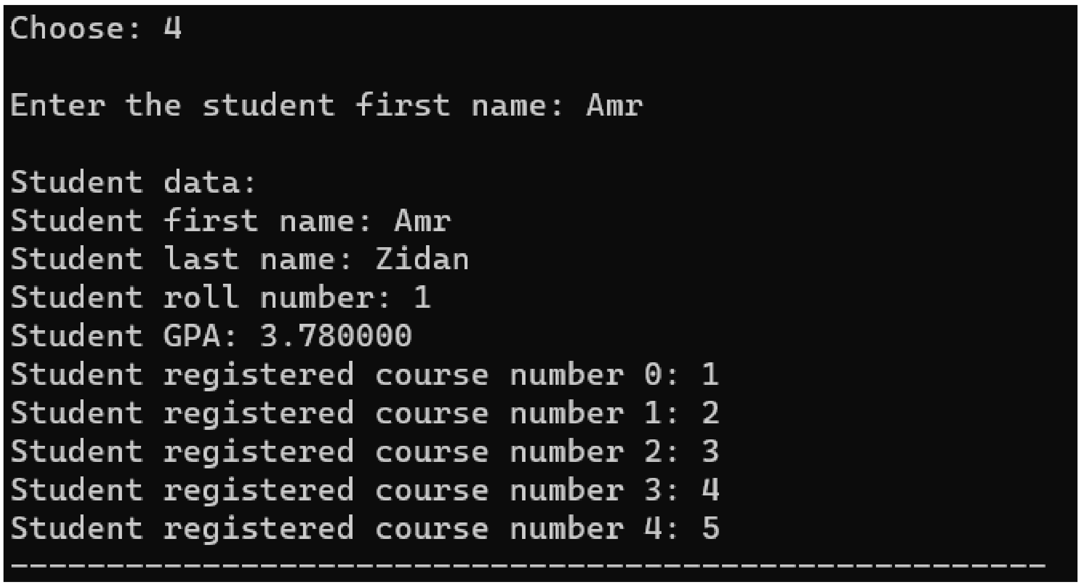
```
------------------------------------------------------


Choose: 6
The total number of students registered is: 5
------------------------------------------------------
```

5. Finding a student with roll number



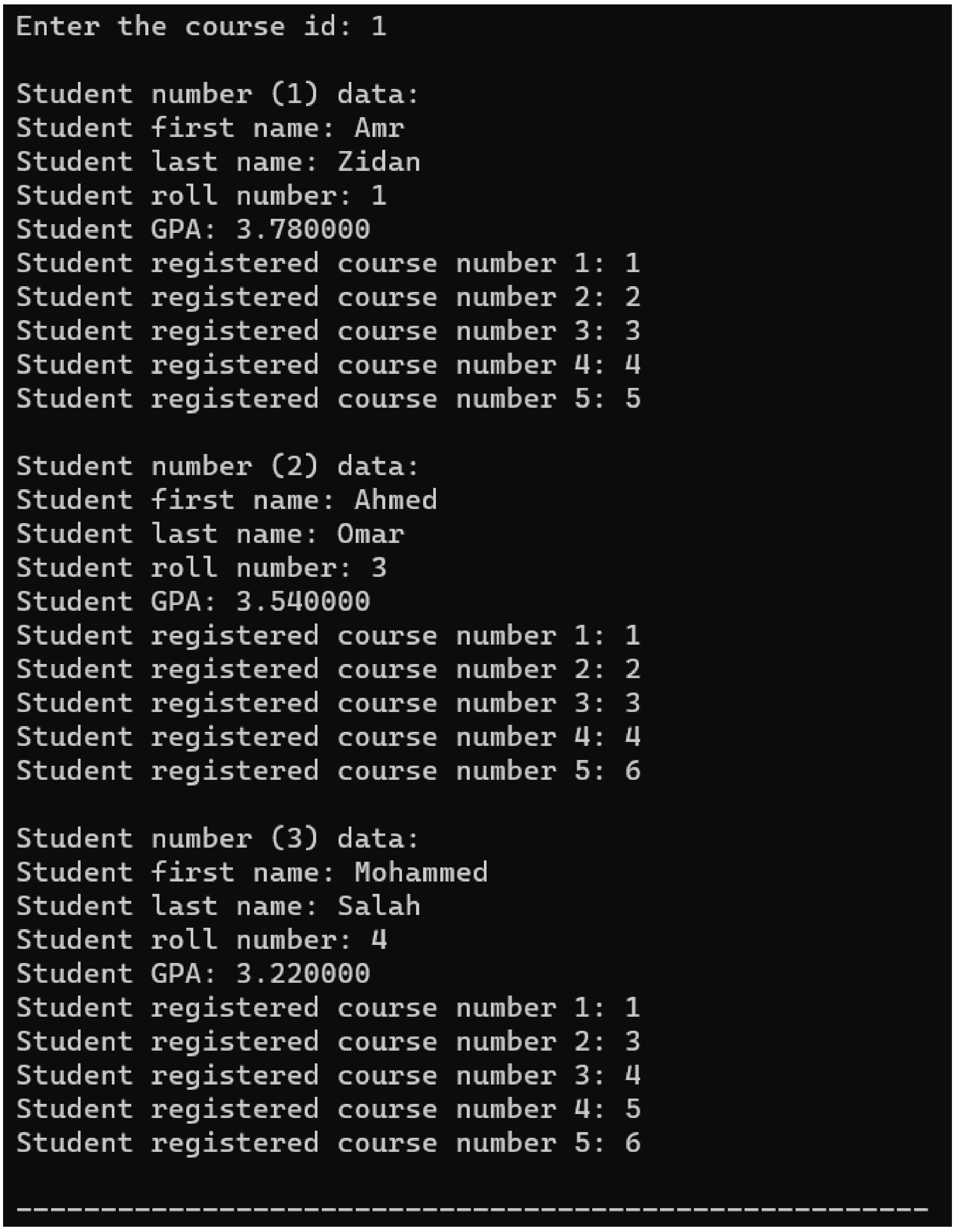
```
Choose: 3

Enter the student roll number: 1

Student data:
Student first name: Amr
Student last name: Zidan
Student roll number: 1
Student GPA: 3.780000
Student registered course number 0: 1
Student registered course number 1: 2
Student registered course number 2: 3
Student registered course number 3: 4
Student registered course number 4: 5
------------------------------------------------------
```

6. Finding a student with first name



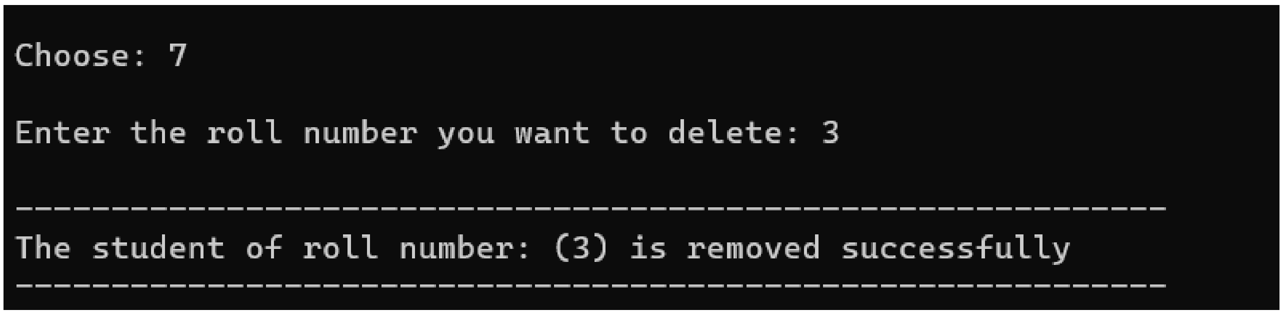
```
Choose: 4

Enter the student first name: Amr

Student data:
Student first name: Amr
Student last name: Zidan
Student roll number: 1
Student GPA: 3.780000
Student registered course number 0: 1
Student registered course number 1: 2
Student registered course number 2: 3
Student registered course number 3: 4
Student registered course number 4: 5
------------------------------------------------------
```

7. Finding a student with course id

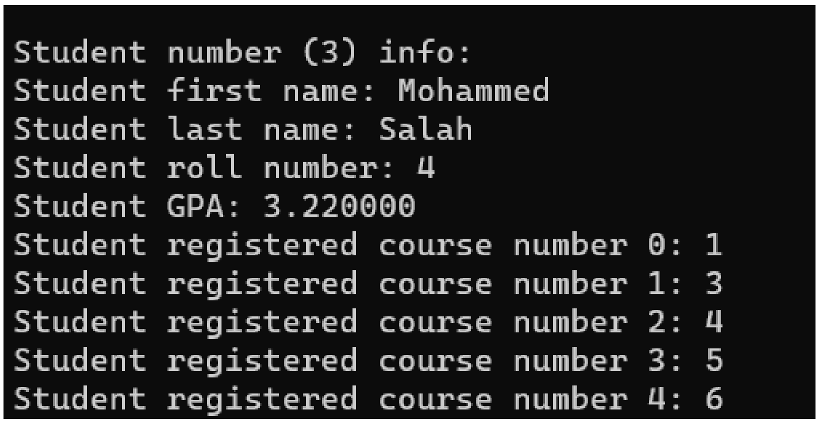


```
Enter the course id: 1

Student number (1) data:
Student first name: Amr
Student last name: Zidan
Student roll number: 1
Student GPA: 3.780000
Student registered course number 1: 1
Student registered course number 2: 2
Student registered course number 3: 3
Student registered course number 4: 4
Student registered course number 5: 5

Student number (2) data:
Student first name: Ahmed
Student last name: Omar
Student roll number: 3
Student GPA: 3.540000
Student registered course number 1: 1
Student registered course number 2: 2
Student registered course number 3: 3
Student registered course number 4: 4
Student registered course number 5: 6

Student number (3) data:
Student first name: Mohammed
Student last name: Salah
Student roll number: 4
Student GPA: 3.220000
Student registered course number 1: 1
Student registered course number 2: 3
Student registered course number 3: 4
Student registered course number 4: 5
Student registered course number 5: 6

------------------------------------------------------
```

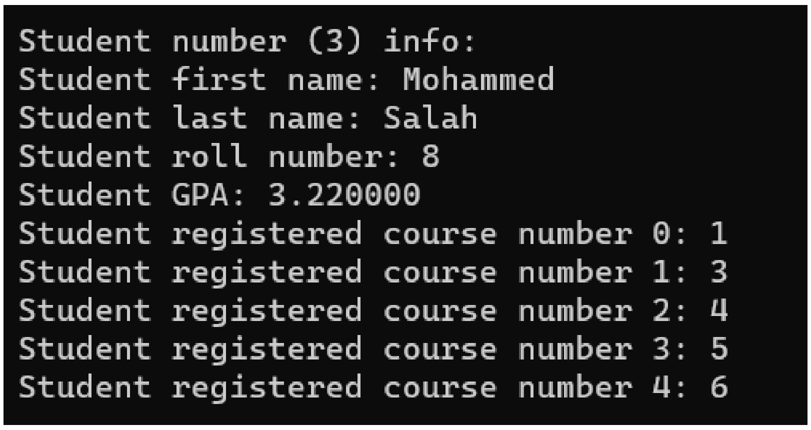
8. Deleting a student


```
Choose: 7

Enter the roll number you want to delete: 3

------------------------------------------------------------
The student of roll number: (3) is removed successfully
------------------------------------------------------------
```

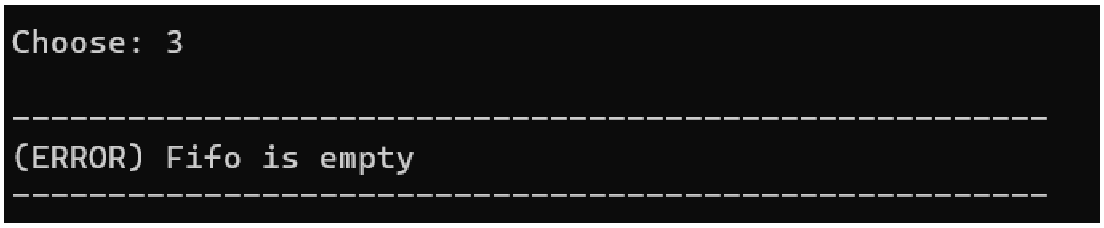
After deletion:


```
Student number (3) info:
Student first name: Mohammed
Student last name: Salah
Student roll number: 4
Student GPA: 3.220000
Student registered course number 0: 1
Student registered course number 1: 3
Student registered course number 2: 4
Student registered course number 3: 5
Student registered course number 4: 6
```

9. Updating student's roll number


```
Student number (3) info:
Student first name: Mohammed
Student last name: Salah
Student roll number: 8
Student GPA: 3.220000
Student registered course number 0: 1
Student registered course number 1: 3
Student registered course number 2: 4
Student registered course number 3: 5
Student registered course number 4: 6
```

10. Trying operation while FIFO is empty



```
Choose: 3

-----------------------------------------------------------
(ERROR) Fifo is empty
-----------------------------------------------------------
```