Learn-In-Depth

Student Management System

Prepared by:
Mostafa Mohamed Edrees

Supervisor:
Eng. Keroles Shenouda

My Progress Page:

[Mostafa Edrees]

[GitHub]

[Drive]

AUGUST 1, 2023

LEARN-IN-DEPTH

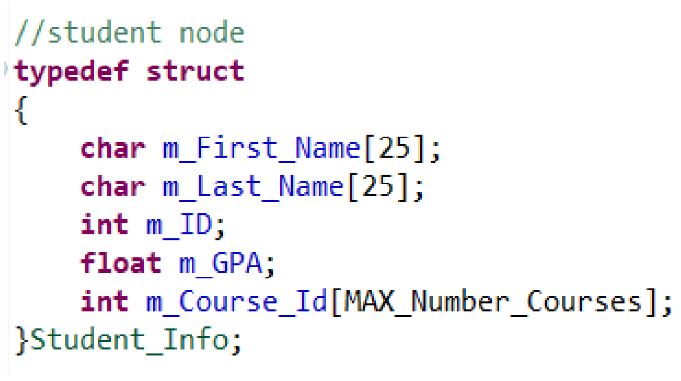
Table of Contents

Problem Statement

Write a program to build a simple software for Student Information Management System which can perform the following operations:

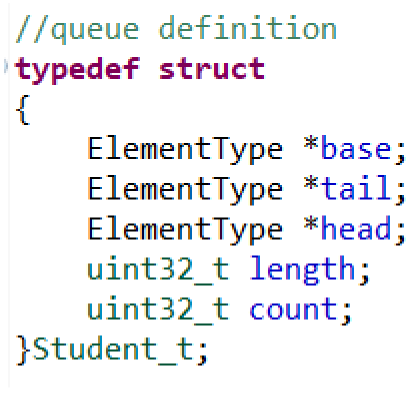
- Store first name of the student.
- Store last name of the student.
- Store unique id number for each student.
- Store GPA for each student.
- Store courses registered by the student.

Student Structure



```
//student node
typedef struct
{
    char m_First_Name[25];
    char m_Last_Name[25];
    int m_ID;
    float m_GPA;
    int m_Course_Id[MAX_Number_Courses];
}Student_Info;
```

Database Structure



```
//queue definition
typedef struct
{
    ElementType *base;
    ElementType *tail;
    ElementType *head;
    uint32_t length;
    uint32_t count;
}Student_t;
```

Functions

There are 11 options for the student to use it to handle his information.

- Add The Student Details Manually.
- Add The Student Details From Text File.
- Find Student Details by ID Number.
- Find Student Details by First Name.
- Find Student Details by Course ID.
- Find the Total Number of Students.
- Delete Student Details by ID Number.
- Delete All Students.
- Update Student Details by ID Number.
- Show All Information.
- Exit.

Functions Prototypes

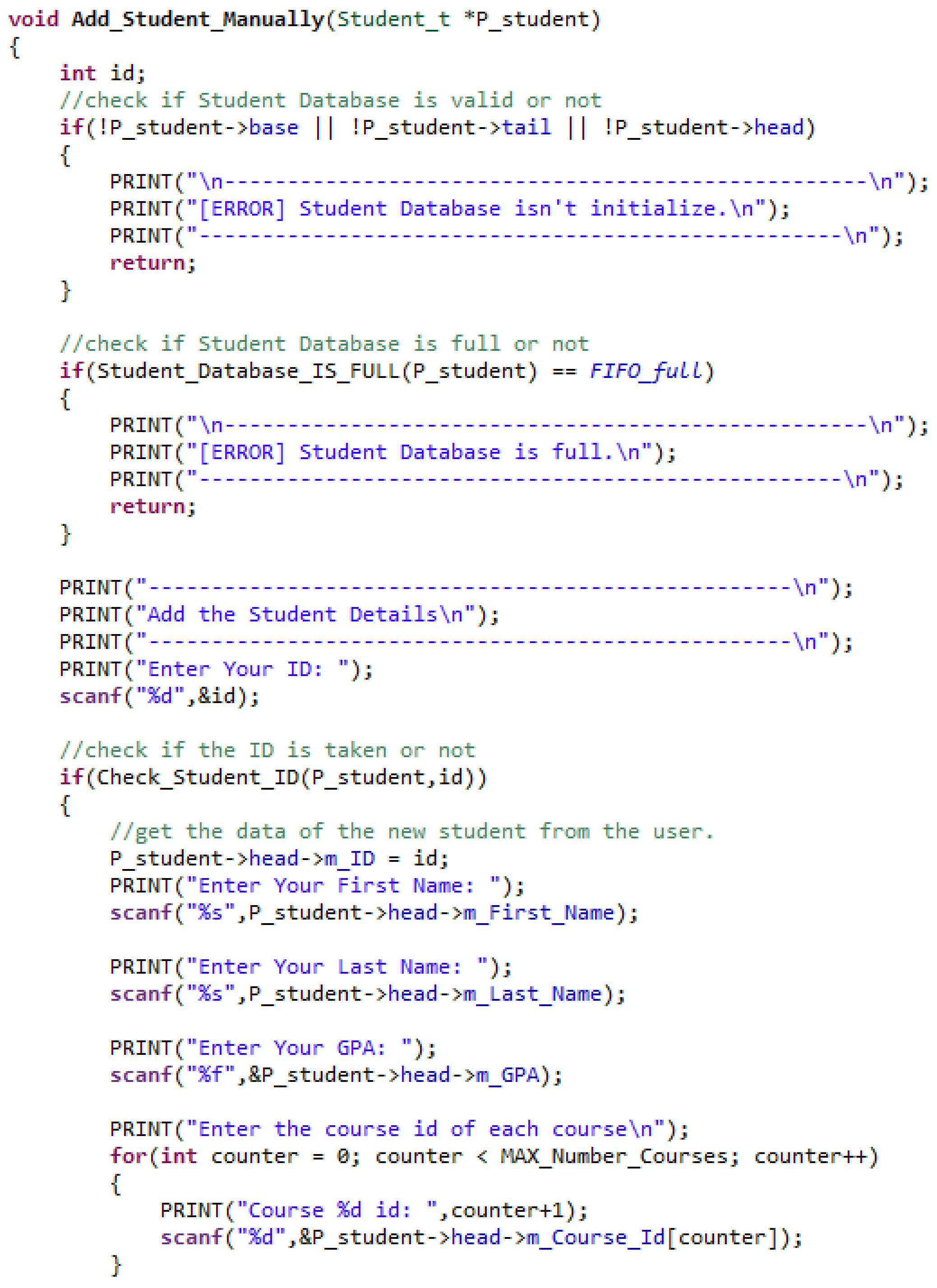
```c
//prototypes of functions
FIFO_Buf_Status Student_Database_init (Student_t *P_student, ElementType *buf, uint32_t length);
void Add_Student_Manually(Student_t *P_student);
void Add_Student_From_File(Student_t *P_student,FILE *pf);
void Find_Student_By_ID(Student_t *P_student);
void Find_Student_By_First_Name(Student_t *P_student);
void Find_Students_Enroll_Course(Student_t *P_student);
void Total_Students(Student_t *P_student);
void Delete_Student_By_ID(Student_t *P_student);
void Delete_All_Students(Student_t *P_student);
void Updata_Student(Student_t *P_student);
void Show_All_Students(Student_t *P_student);
void Show_Student_Information(ElementType *P_student_data);
int Check_Student_ID(Student_t *P_student,int id);
FIFO_Buf_Status Student_Database_IS_FULL(Student_t *P_student);
```

Student Database initialize

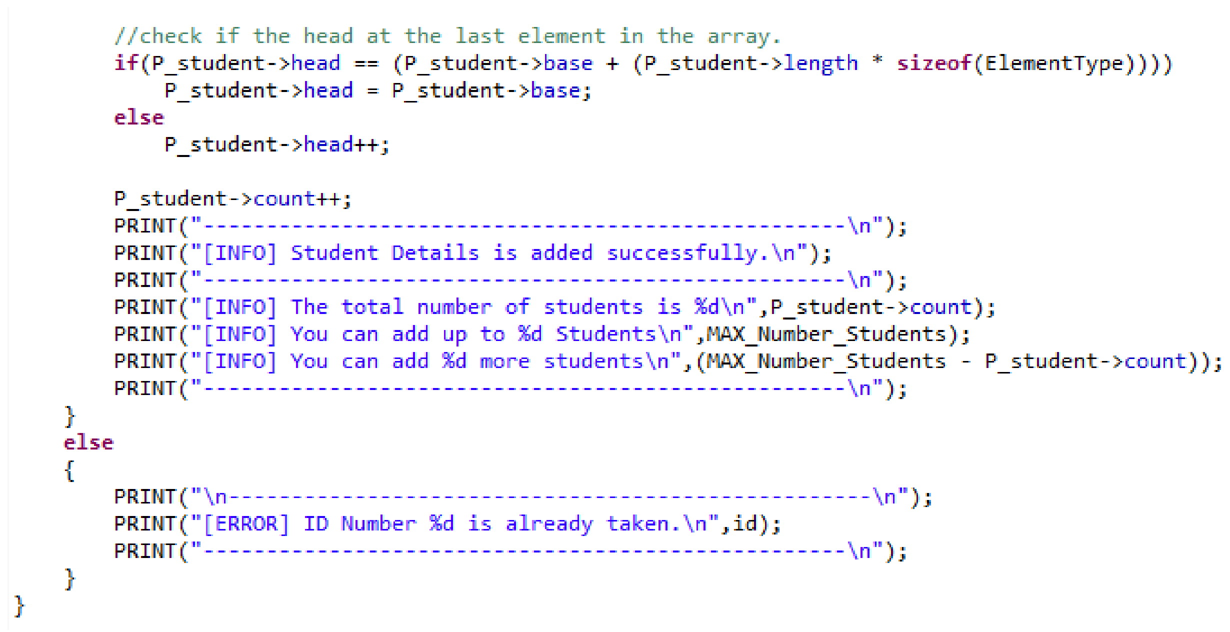
```c
FIFO_Buf_Status Student_Database_init (Student_t *P_student, ElementType *buf, uint32_t length)
{
    //check if buffer is reserved at memory or not
    if(buf == NULL)
        return FIFO_null;

    //initialize Student Database
    P_student->base = buf;
    P_student->tail = buf;
    P_student->head = buf;
    P_student->length = length;
    P_student->count = 0;
    return FIFO_no_error;
}
```

Add the student details manually.

It’s used to get data from the user and check if the ID number entered by user is unique or not if we find that ID is unique then we add a student to the Database.



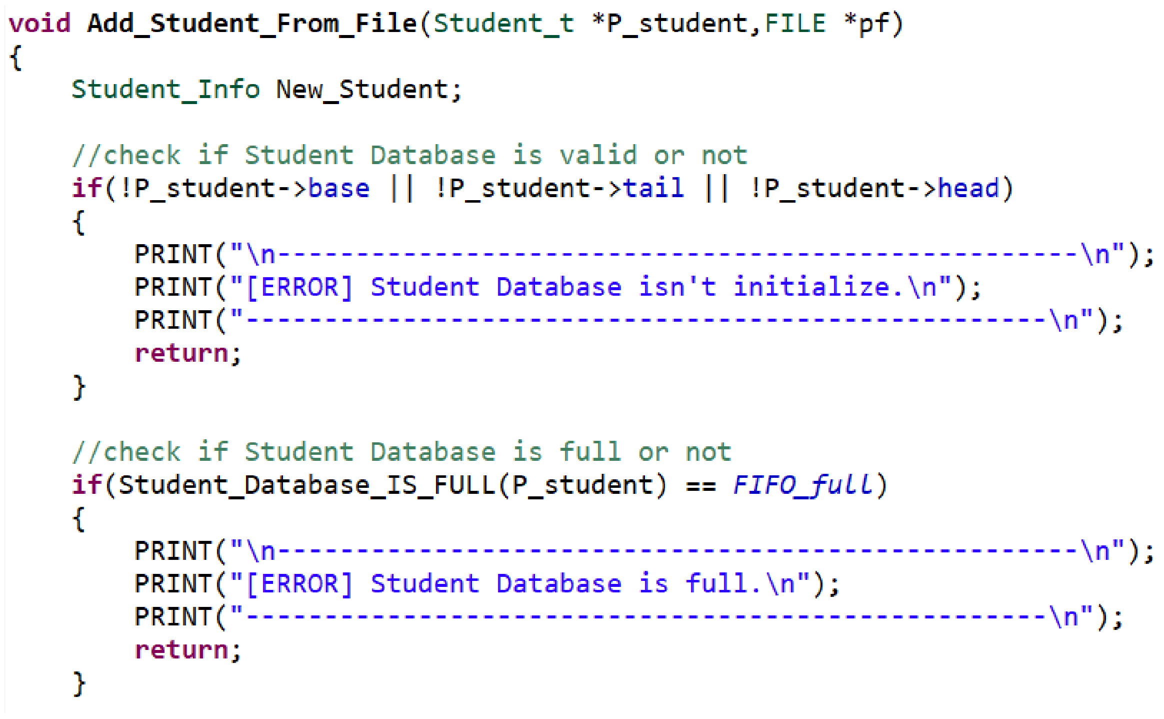
```c
void Add_Student_Manually(Student_t *P_student)
{
    int id;
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    //check if Student Database is full or not
    if(Student_Database_IS_FULL(P_student) == FIFO_full)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is full.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    PRINT("---------------------------------------------------\n");
    PRINT("Add the Student Details\n");
    PRINT("---------------------------------------------------\n");
    PRINT("Enter Your ID: ");
    scanf("%d",&id);

    //check if the ID is taken or not
    if(Check_Student_ID(P_student,id))
    {
        //get the data of the new student from the user.
        P_student->head->m_ID = id;
        PRINT("Enter Your First Name: ");
        scanf("%s",P_student->head->m_First_Name);

        PRINT("Enter Your Last Name: ");
        scanf("%s",P_student->head->m_Last_Name);

        PRINT("Enter Your GPA: ");
        scanf("%f",&P_student->head->m_GPA);

        PRINT("Enter the course id of each course\n");
        for(int counter = 0; counter < MAX_Number_Courses; counter++)
        {
            PRINT("Course %d id: ",counter+1);
            scanf("%d",&P_student->head->m_Course_Id[counter]);
        }
```



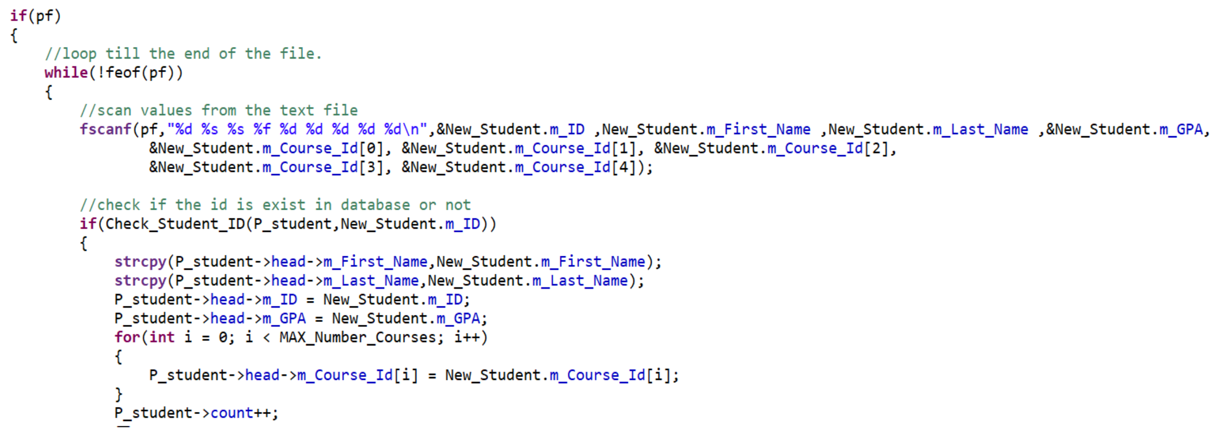
```c
        //check if the head at the last element in the array.
        if(P_student->head == (P_student->base + (P_student->length * sizeof(ElementType))))
            P_student->head = P_student->base;
        else
            P_student->head++;

        P_student->count++;
        PRINT("---------------------------------------------------\n");
        PRINT("[INFO] Student Details is added successfully.\n");
        PRINT("---------------------------------------------------\n");
        PRINT("[INFO] The total number of students is %d\n",P_student->count);
        PRINT("[INFO] You can add up to %d Students\n",MAX_Number_Students);
        PRINT("[INFO] You can add %d more students\n",(MAX_Number_Students - P_student->count));
        PRINT("---------------------------------------------------\n");
    }
    else
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] ID Number %d is already taken.\n",id);
        PRINT("---------------------------------------------------\n");
    }
}
```

Add the student details from text file

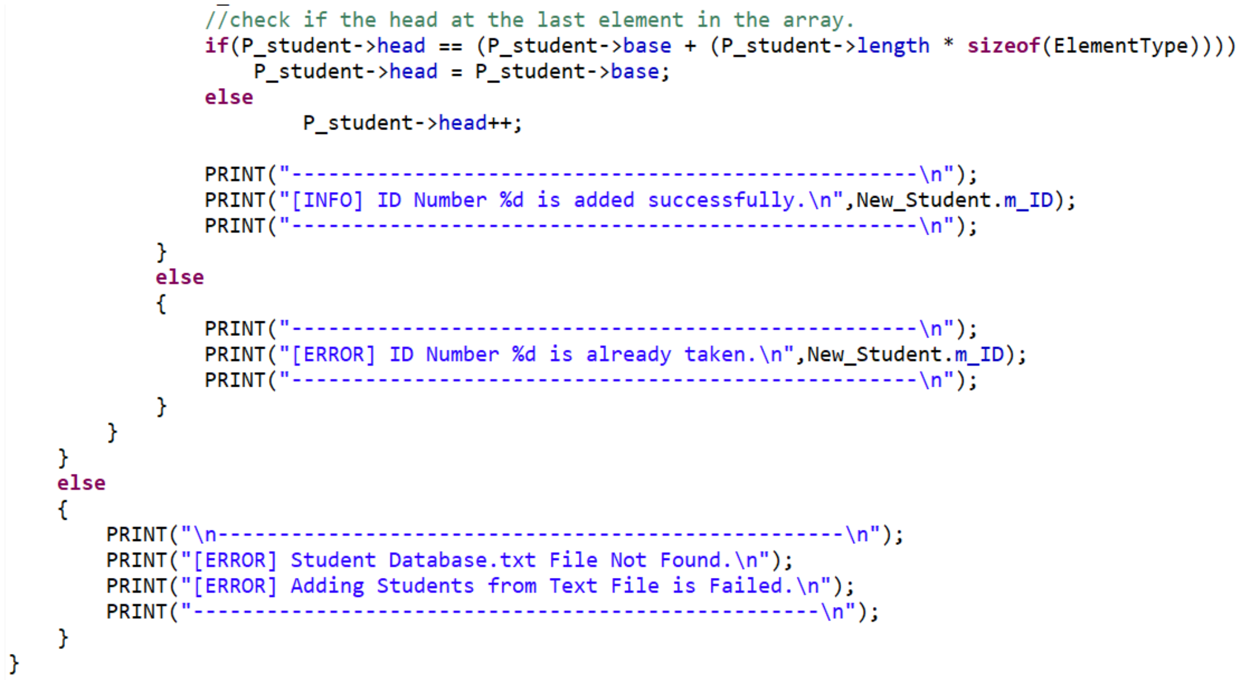
It's used to get data from a text file and check if the ID number is unique or not if we find that ID is unique then we add a student to the Database.



```c
void Add_Student_From_File(Student_t *P_student,FILE *pf)
{
    Student_Info New_Student;

    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    //check if Student Database is full or not
    if(Student_Database_IS_FULL(P_student) == FIFO_full)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is full.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }
```



```c
    if(pf)
    {
        //loop till the end of the file.
        while(!feof(pf))
        {
            //scan values from the text file
            fscanf(pf,"%d %s %s %f %d %d %d %d %d\n",&New_Student.m_ID ,New_Student.m_First_Name ,New_Student.m_Last_Name ,&New_Student.m_GPA,
                    &New_Student.m_Course_Id[0], &New_Student.m_Course_Id[1], &New_Student.m_Course_Id[2],
                    &New_Student.m_Course_Id[3], &New_Student.m_Course_Id[4]);

            //check if the id is exist in database or not
            if(Check_Student_ID(P_student,New_Student.m_ID))
            {
                strcpy(P_student->head->m_First_Name,New_Student.m_First_Name);
                strcpy(P_student->head->m_Last_Name,New_Student.m_Last_Name);
                P_student->head->m_ID = New_Student.m_ID;
                P_student->head->m_GPA = New_Student.m_GPA;
                for(int i = 0; i < MAX_Number_Courses; i++)
                {
                    P_student->head->m_Course_Id[i] = New_Student.m_Course_Id[i];
                }
                P_student->count++;
```



```c
                //check if the head at the last element in the array.
                if(P_student->head == (P_student->base + (P_student->length * sizeof(ElementType))))
                    P_student->head = P_student->base;
                else
                        P_student->head++;

                PRINT("---------------------------------------------------\n");
                PRINT("[INFO] ID Number %d is added successfully.\n",New_Student.m_ID);
                PRINT("---------------------------------------------------\n");
            }
            else
            {
                PRINT("---------------------------------------------------\n");
                PRINT("[ERROR] ID Number %d is already taken.\n",New_Student.m_ID);
                PRINT("---------------------------------------------------\n");
            }
        }
    }
    else
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database.txt File Not Found.\n");
        PRINT("[ERROR] Adding Students from Text File is Failed.\n");
        PRINT("---------------------------------------------------\n");
    }
}
```

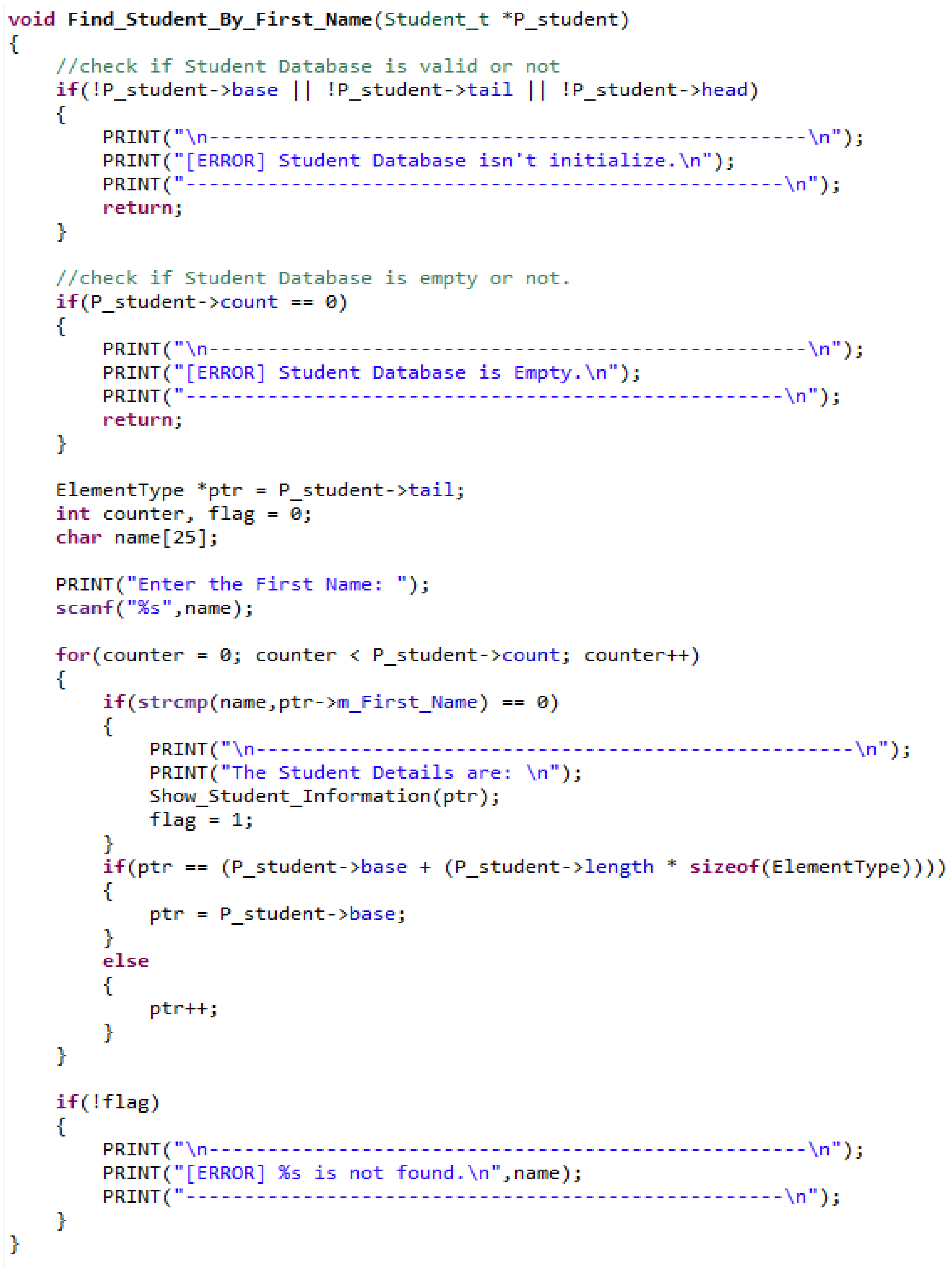
Find student details by ID number

It’s used to get the student details from database by its own id.

```
void Find_Student_By_ID(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    ElementType *ptr = P_student->tail;
    int ID, counter, flag = 0;

    PRINT("Enter your ID: ");
    scanf("%d",&ID);

    for(counter = 0; counter < P_student->count; counter++)
    {
        if(ID == ptr->m_ID)
        {
            PRINT("\n---------------------------------------------------\n");
            PRINT("The Student Details are: \n");
            Show_Student_Information(ptr);
            flag = 1;
            break;
        }
        if(ptr == (P_student->base + (P_student->length * sizeof(ElementType))))
        {
            ptr = P_student->base;
        }
        else
        {
            ptr++;
        }
    }

    if(!flag)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] ID Number %d not found.\n",ID);
        PRINT("---------------------------------------------------\n");
    }
}
```

Find student details by first name

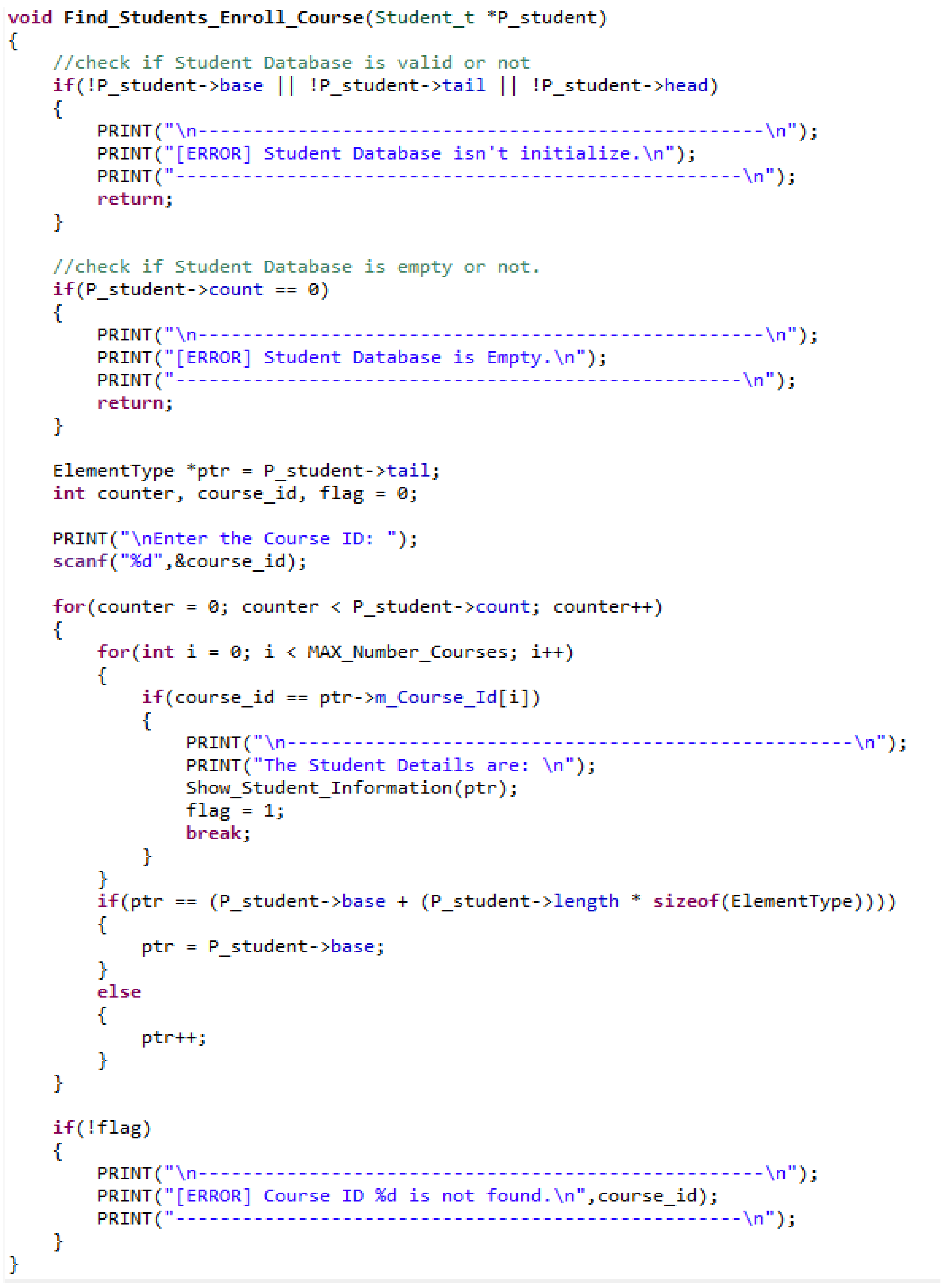
It’s used to get student details by its own first name.



```
void Find_Student_By_First_Name(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n--------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("--------------------------------------------------\n");
        return;
    }

    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n--------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("--------------------------------------------------\n");
        return;
    }

    ElementType *ptr = P_student->tail;
    int counter, flag = 0;
    char name[25];

    PRINT("Enter the First Name: ");
    scanf("%s",name);

    for(counter = 0; counter < P_student->count; counter++)
    {
        if(strcmp(name,ptr->m_First_Name) == 0)
        {
            PRINT("\n--------------------------------------------------\n");
            PRINT("The Student Details are: \n");
            Show_Student_Information(ptr);
            flag = 1;
        }
        if(ptr == (P_student->base + (P_student->length * sizeof(ElementType))))
        {
            ptr = P_student->base;
        }
        else
        {
            ptr++;
        }
    }

    if(!flag)
    {
        PRINT("\n--------------------------------------------------\n");
        PRINT("[ERROR] %s is not found.\n",name);
        PRINT("--------------------------------------------------\n");
    }
}
```

Find student details by course ID

It’s used to get the students registered in specific course.



```
void Find_Students_Enroll_Course(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n--------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("--------------------------------------------------\n");
        return;
    }

    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n--------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("--------------------------------------------------\n");
        return;
    }

    ElementType *ptr = P_student->tail;
    int counter, course_id, flag = 0;

    PRINT("\nEnter the Course ID: ");
    scanf("%d",&course_id);

    for(counter = 0; counter < P_student->count; counter++)
    {
        for(int i = 0; i < MAX_Number_Courses; i++)
        {
            if(course_id == ptr->m_Course_Id[i])
            {
                PRINT("\n--------------------------------------------------\n");
                PRINT("The Student Details are: \n");
                Show_Student_Information(ptr);
                flag = 1;
                break;
            }
        }
        if(ptr == (P_student->base + (P_student->length * sizeof(ElementType))))
        {
            ptr = P_student->base;
        }
        else
        {
            ptr++;
        }
    }

    if(!flag)
    {
        PRINT("\n--------------------------------------------------\n");
        PRINT("[ERROR] Course ID %d is not found.\n",course_id);
        PRINT("--------------------------------------------------\n");
    }
}
```

Find the total number of students

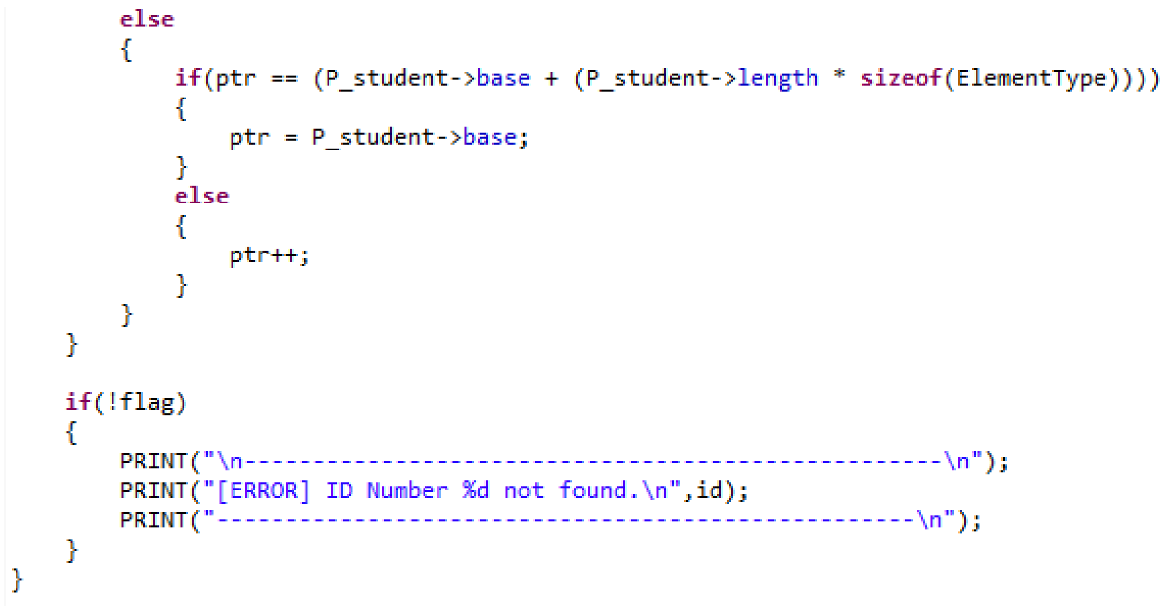
It’s used to get the total number of students in the database.

```
void Total_Students(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    PRINT("\n---------------------------------------------------\n");
    PRINT("[INFO] The total number of students is %d\n",P_student->count);
    PRINT("[INFO] You can add up to %d Students\n",MAX_Number_Students);
    PRINT("[INFO] You can add %d more students\n",(MAX_Number_Students - P_student->count));
    PRINT("---------------------------------------------------\n");
}
```

Delete student details by ID number

It’s used to delete a student from database with its own id.

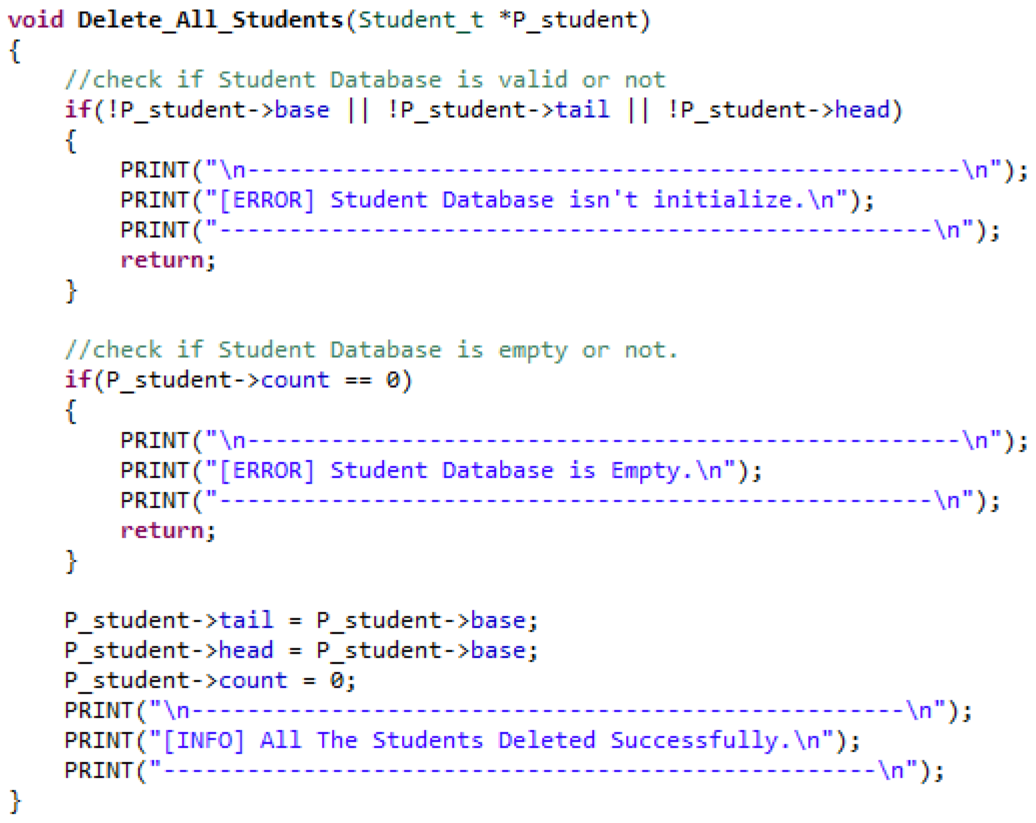
```c
void Delete_Student_By_ID(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    ElementType *ptr = P_student->tail;
    int id, counter, flag = 0;
    PRINT("\nEnter the ID that you want to delete it: ");
    scanf("%d",&id);

    for(counter = 0; counter < P_student->count; counter++)
    {
        if(id == ptr->m_ID)
        {
            for(int i = counter; i < P_student->count; i++)
            {
                if(ptr == (P_student->base + (P_student->length * sizeof(ElementType))))
                {
                    *ptr = *(P_student->base);
                    ptr = P_student->base;
                }
                else
                {
                    *ptr = *(ptr+1);
                }
            }
            P_student->count--;

            PRINT("\n---------------------------------------------------\n");
            PRINT("[INFO] The ID Number %d is removed successfully.\n",id);
            PRINT("---------------------------------------------------\n");
            flag = 1;
            break;
        }
```



```
        else
        {
            if(ptr == (P_student->base + (P_student->length * sizeof(ElementType))))
            {
                ptr = P_student->base;
            }
            else
            {
                ptr++;
            }
        }
    }

    if(!flag)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] ID Number %d not found.\n",id);
        PRINT("---------------------------------------------------\n");
    }
}
```

Delete all students

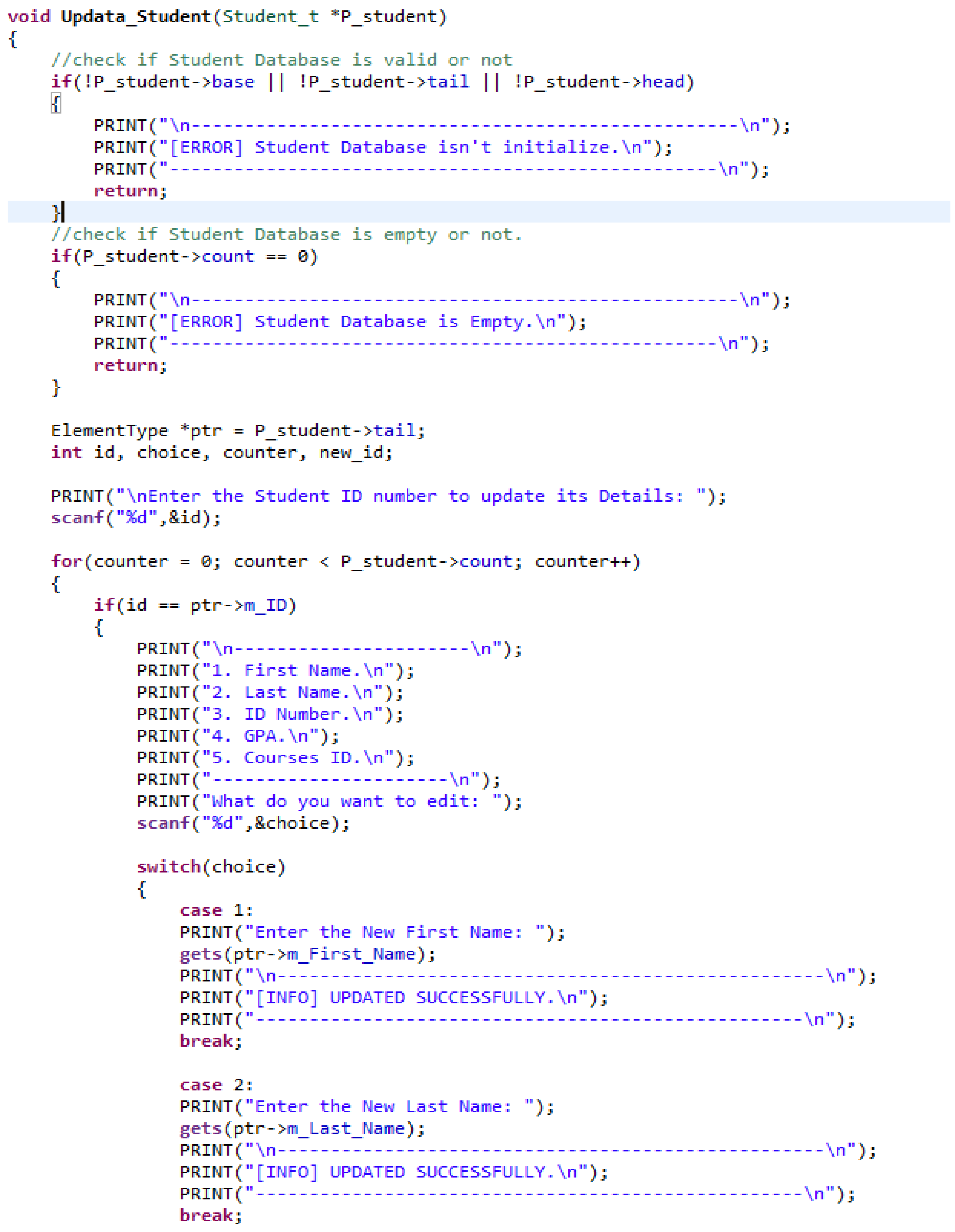
It’s used to delete all the students in the database.



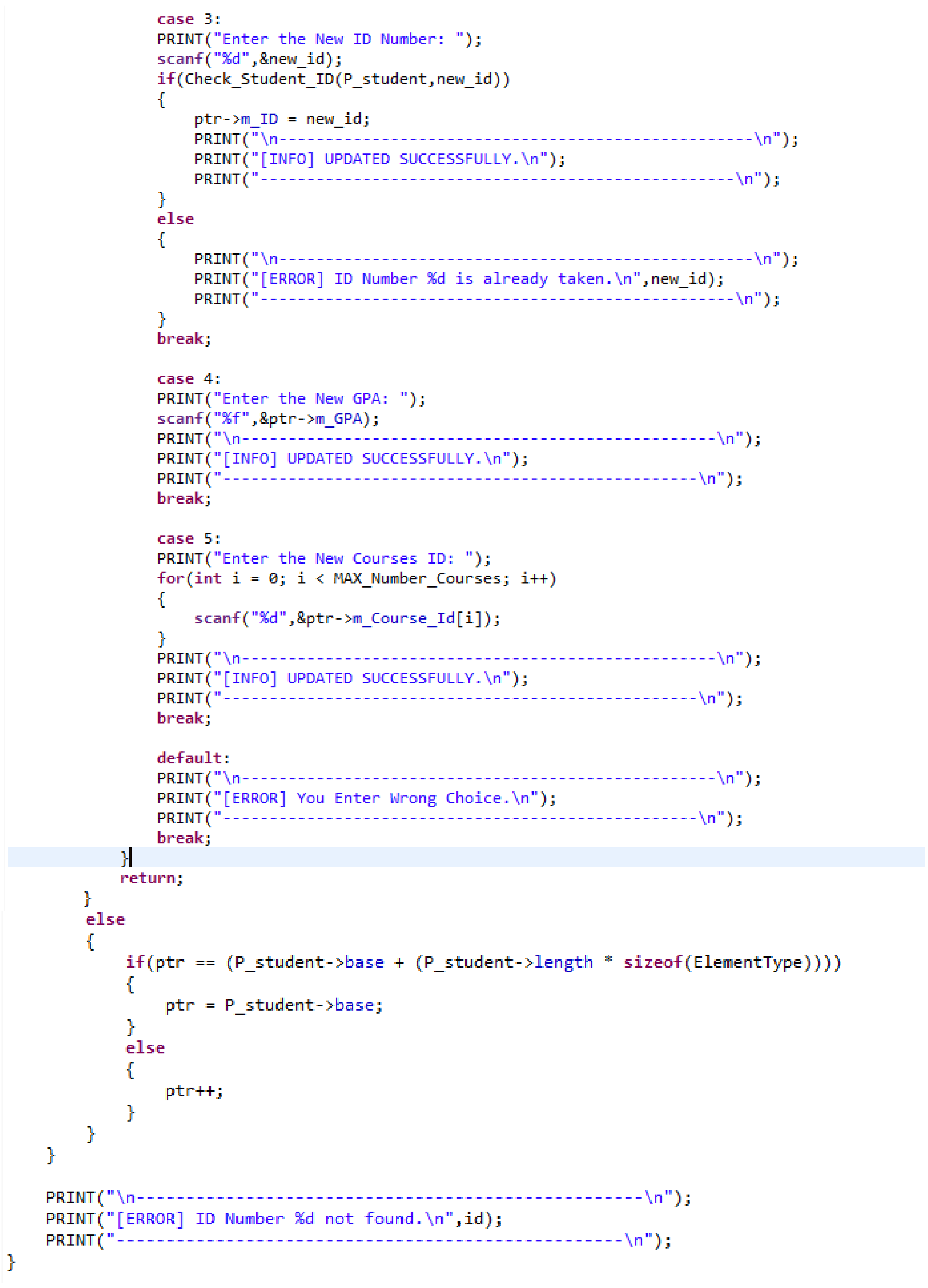
```
void Delete_All_Students(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    P_student->tail = P_student->base;
    P_student->head = P_student->base;
    P_student->count = 0;
    PRINT("\n---------------------------------------------------\n");
    PRINT("[INFO] All The Students Deleted Successfully.\n");
    PRINT("---------------------------------------------------\n");
}
```

Update student details by ID number

It’s used to update the details of any student in the database.



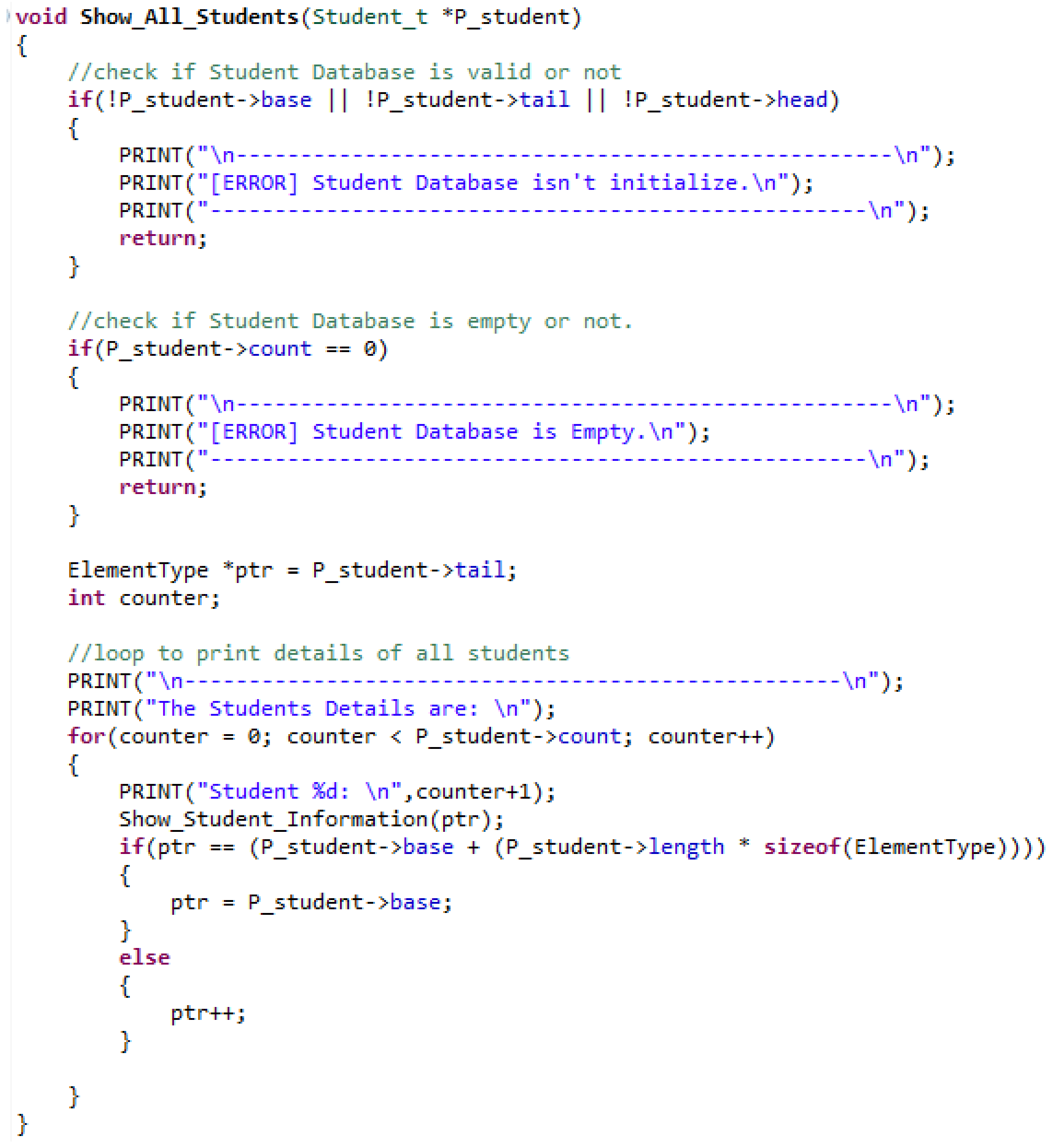
```c
void Updata_Student(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }
    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    ElementType *ptr = P_student->tail;
    int id, choice, counter, new_id;

    PRINT("\nEnter the Student ID number to update its Details: ");
    scanf("%d",&id);

    for(counter = 0; counter < P_student->count; counter++)
    {
        if(id == ptr->m_ID)
        {
            PRINT("\n----------------------\n");
            PRINT("1. First Name.\n");
            PRINT("2. Last Name.\n");
            PRINT("3. ID Number.\n");
            PRINT("4. GPA.\n");
            PRINT("5. Courses ID.\n");
            PRINT("----------------------\n");
            PRINT("What do you want to edit: ");
            scanf("%d",&choice);

            switch(choice)
            {
                case 1:
                PRINT("Enter the New First Name: ");
                gets(ptr->m_First_Name);
                PRINT("\n---------------------------------------------------\n");
                PRINT("[INFO] UPDATED SUCCESSFULLY.\n");
                PRINT("---------------------------------------------------\n");
                break;

                case 2:
                PRINT("Enter the New Last Name: ");
                gets(ptr->m_Last_Name);
                PRINT("\n---------------------------------------------------\n");
                PRINT("[INFO] UPDATED SUCCESSFULLY.\n");
                PRINT("---------------------------------------------------\n");
                break;
```



```c
                case 3:
                PRINT("Enter the New ID Number: ");
                scanf("%d",&new_id);
                if(Check_Student_ID(P_student,new_id))
                {
                    ptr->m_ID = new_id;
                    PRINT("\n-----------------------------------------------------\n");
                    PRINT("[INFO] UPDATED SUCCESSFULLY.\n");
                    PRINT("-----------------------------------------------------\n");
                }
                else
                {
                    PRINT("\n-----------------------------------------------------\n");
                    PRINT("[ERROR] ID Number %d is already taken.\n",new_id);
                    PRINT("-----------------------------------------------------\n");
                }
                break;

                case 4:
                PRINT("Enter the New GPA: ");
                scanf("%f",&ptr->m_GPA);
                PRINT("\n-----------------------------------------------------\n");
                PRINT("[INFO] UPDATED SUCCESSFULLY.\n");
                PRINT("-----------------------------------------------------\n");
                break;

                case 5:
                PRINT("Enter the New Courses ID: ");
                for(int i = 0; i < MAX_Number_Courses; i++)
                {
                    scanf("%d",&ptr->m_Course_Id[i]);
                }
                PRINT("\n-----------------------------------------------------\n");
                PRINT("[INFO] UPDATED SUCCESSFULLY.\n");
                PRINT("-----------------------------------------------------\n");
                break;

                default:
                PRINT("\n-----------------------------------------------------\n");
                PRINT("[ERROR] You Enter Wrong Choice.\n");
                PRINT("-----------------------------------------------------\n");
                break;
            }
            return;
        }
        else
        {
            if(ptr == (P_student->base + (P_student->length * sizeof(ElementType))))
            {
                ptr = P_student->base;
            }
            else
            {
                ptr++;
            }
        }
    }

    PRINT("\n-----------------------------------------------------\n");
    PRINT("[ERROR] ID Number %d not found.\n",id);
    PRINT("-----------------------------------------------------\n");
}
```

Show all information

It’s used to show all details of the students in the database.



```c
void Show_All_Students(Student_t *P_student)
{
    //check if Student Database is valid or not
    if(!P_student->base || !P_student->tail || !P_student->head)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database isn't initialize.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    //check if Student Database is empty or not.
    if(P_student->count == 0)
    {
        PRINT("\n---------------------------------------------------\n");
        PRINT("[ERROR] Student Database is Empty.\n");
        PRINT("---------------------------------------------------\n");
        return;
    }

    ElementType *ptr = P_student->tail;
    int counter;

    //loop to print details of all students
    PRINT("\n---------------------------------------------------\n");
    PRINT("The Students Details are: \n");
    for(counter = 0; counter < P_student->count; counter++)
    {
        PRINT("Student %d: \n",counter+1);
        Show_Student_Information(ptr);
        if(ptr == (P_student->base + (P_student->length * sizeof(ElementType))))
        {
            ptr = P_student->base;
        }
        else
        {
            ptr++;
        }

    }
}
```

Additional Functions

Show student information

It's used to print the details of the student.

```
void Show_Student_Information(ElementType *P_student_data)
{
    PRINT("---------------------------------------------------\n");
    PRINT("The First Name is %s\n",P_student_data->m_First_Name);
    PRINT("The Last Name is %s\n",P_student_data->m_Last_Name);
    PRINT("ID Number is %d\n",P_student_data->m_ID);
    PRINT("The GPA is %0.2f\n",P_student_data->m_GPA);
    PRINT("The Courses you enrolled in: \n");
    for(int counter = 0; counter < MAX_Number_Courses; counter++)
    {
        PRINT("    The Course ID is %d\n",P_student_data->m_Course_Id[counter]);
    }
    PRINT("---------------------------------------------------\n");
}
```

Student database is full

It's used to check if the database if full or not.

```
FIFO_Buf_Status Student_Database_IS_FULL(Student_t *P_student)
{
    //check if fifo is valid or not.
    if(!P_student->base || !P_student->tail || !P_student->head)
        return FIFO_null;

    if(P_student->count == MAX_Number_Students)
        return FIFO_full;

    return FIFO_no_error;
}
```

Check student ID

It's used to check the student Id that is unique or not.

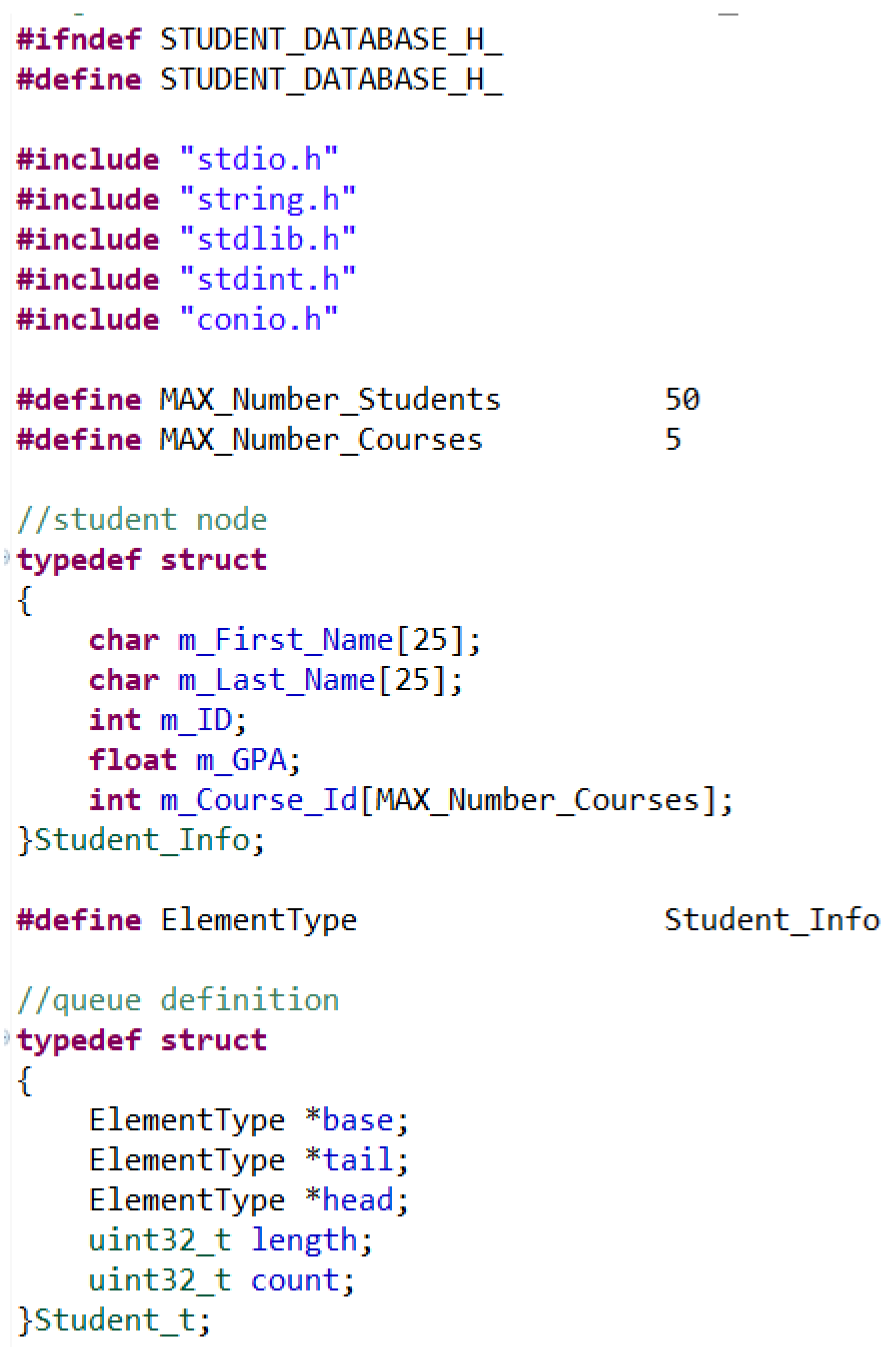
```
int Check_Student_ID(Student_t *P_student,int id)
{
    Student_Info *p_student_node = P_student->tail;
    for(int counter = 0; counter < P_student->count; counter++)
    {
        if(p_student_node->m_ID == id)
            return 0;
    }
    return 1;
}
```

Main

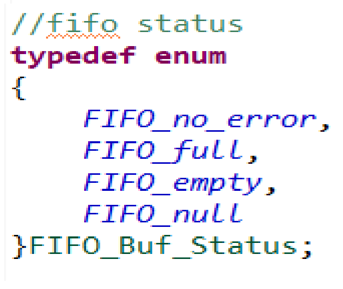
```
#include "Student Database.h"

int main()
{
    //initialize Student Database
    ElementType Student_Buffer[MAX_Number_Students];
    Student_t Student_Database;

    FILE *P_file = fopen("Student Database.txt", "r+");

    Student_Database_init(&Student_Database,Student_Buffer,MAX_Number_Students);

    int choice;

    PRINT("\n********************************************************\n");
    PRINT("Welcome to the Student Management System\n");
    PRINT("********************************************************\n");

    while(1)
    {
        PRINT("\n===================================================\n");
        PRINT("Choose The Task that you want to perform\n");
        PRINT("==================================================\n");
        PRINT("1.  Add The Student Details Manually.\n");
        PRINT("2.  Add The Student Details From Text File.\n");
        PRINT("3.  Find Student Details by ID Number.\n");
        PRINT("4.  Find Student Details by First Name.\n");
        PRINT("5.  Find Student Details by Course ID.\n");
        PRINT("6.  Find the Total Number of Students.\n");
        PRINT("7.  Delete Student Details by ID Number.\n");
        PRINT("8.  Delete All Students.\n");
        PRINT("9.  Update Student Details by ID Number.\n");
        PRINT("10.  Show All Information.\n");
        PRINT("11. Exit.\n");
        PRINT("================================================\n");
        PRINT("\nEnter your choose to perform the task: ");
        scanf("%d",&choice);

         switch(choice)
         {
             case 1:
             Add_Student_Manually(&Student_Database);
             break;

             case 2:
             Add_Student_From_File(&Student_Database,P_file);
             break;
```

```c
            case 3:
            Find_Student_By_ID(&Student_Database);
            break;

            case 4:
            Find_Student_By_First_Name(&Student_Database);
            break;

            case 5:
            Find_Students_Enroll_Course(&Student_Database);
            break;

            case 6:
            Total_Students(&Student_Database);
            break;

            case 7:
            Delete_Student_By_ID(&Student_Database);
            break;

            case 8:
            Delete_All_Students(&Student_Database);
            break;

            case 9:
            Updata_Student(&Student_Database);
            break;

            case 10:
            Show_All_Students(&Student_Database);
            break;

            case 11:
            PRINT("\n---------------------------------------------------\n");
            PRINT("[INFO] End of the program.\n");
            PRINT("---------------------------------------------------\n");
            exit(1);
            break;

            default:
            PRINT("\n---------------------------------------------------\n");
            PRINT("[ERROR] You Enter a Wrong Choice.\n");
            PRINT("---------------------------------------------------\n");
            break;

        }
    }


    return 0;
}
```

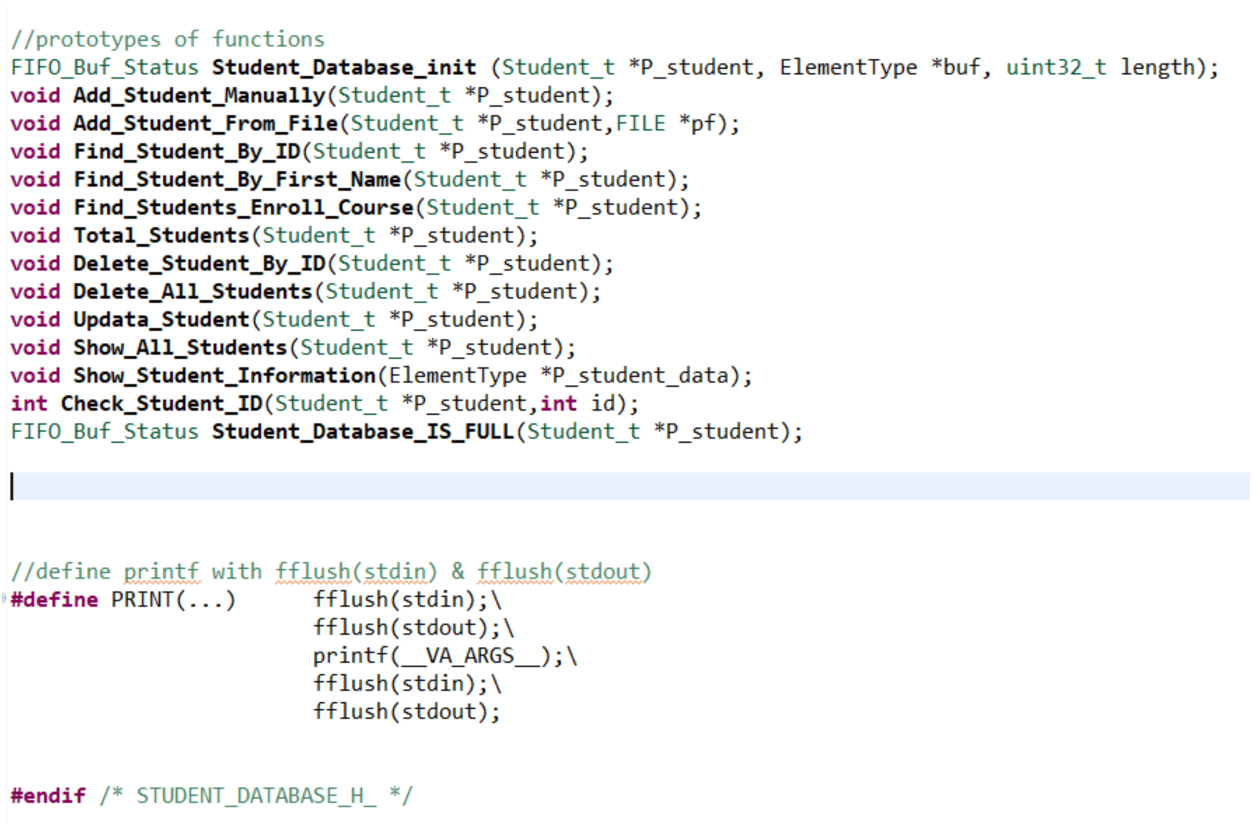
Student Database.h



```
#ifndef STUDENT_DATABASE_H_
#define STUDENT_DATABASE_H_

#include "stdio.h"
#include "string.h"
#include "stdlib.h"
#include "stdint.h"
#include "conio.h"

#define MAX_Number_Students         50
#define MAX_Number_Courses          5

//student node
typedef struct
{
    char m_First_Name[25];
    char m_Last_Name[25];
    int m_ID;
    float m_GPA;
    int m_Course_Id[MAX_Number_Courses];
}Student_Info;

#define ElementType                 Student_Info

//queue definition
typedef struct
{
    ElementType *base;
    ElementType *tail;
    ElementType *head;
    uint32_t length;
    uint32_t count;
}Student_t;
```

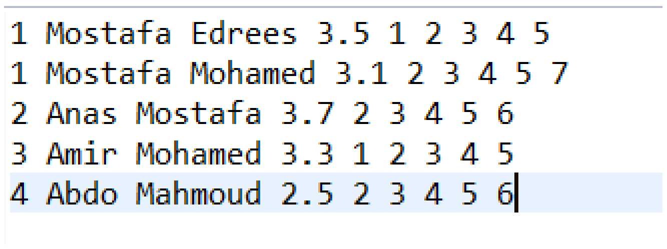


```
//fifo status
typedef enum
{
    FIFO_no_error,
    FIFO_full,
    FIFO_empty,
    FIFO_null
}FIFO_Buf_Status;
```



```c
//prototypes of functions
FIFO_Buf_Status Student_Database_init (Student_t *P_student, ElementType *buf, uint32_t length);
void Add_Student_Manually(Student_t *P_student);
void Add_Student_From_File(Student_t *P_student,FILE *pf);
void Find_Student_By_ID(Student_t *P_student);
void Find_Student_By_First_Name(Student_t *P_student);
void Find_Students_Enroll_Course(Student_t *P_student);
void Total_Students(Student_t *P_student);
void Delete_Student_By_ID(Student_t *P_student);
void Delete_All_Students(Student_t *P_student);
void Updata_Student(Student_t *P_student);
void Show_All_Students(Student_t *P_student);
void Show_Student_Information(ElementType *P_student_data);
int Check_Student_ID(Student_t *P_student,int id);
FIFO_Buf_Status Student_Database_IS_FULL(Student_t *P_student);



//define printf with fflush(stdin) & fflush(stdout)
#define PRINT(...)       fflush(stdin);\
                         fflush(stdout);\
                         printf(__VA_ARGS__);\
                         fflush(stdin);\
                         fflush(stdout);


#endif /* STUDENT_DATABASE_H_ */
```

Student Database.txt



```
1 Mostafa Edrees 3.5 1 2 3 4 5
1 Mostafa Mohamed 3.1 2 3 4 5 7
2 Anas Mostafa 3.7 2 3 4 5 6
3 Amir Mohamed 3.3 1 2 3 4 5
4 Abdo Mahmoud 2.5 2 3 4 5 6
```