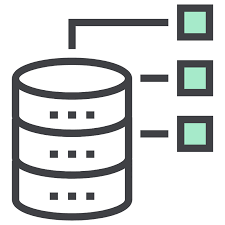
Mastering Embedded System

Online Diploma

[www.learn-in-depth.com](http://www.learn-in-depth.com)

First Term Project 2

Student Database System

BY :

Eng. Mohamed Kamel Aly

My Profile : No progess page created,

Email: [mohamed.kamel.m73@gmail.com](mailto:mohamed.kamel.m73@gmail.com)

Github Code: [https://github.com/Kamelz70/LID\_Diploma/tree/main/First\_Term\_Projects/Project\_2](%20https:/github.com/Kamelz70/LID_Diploma/tree/main/First_Term_Projects/Project_2)

Contents

[Problem Statement 3](#_Toc157593154)

[Approach 3](#_Toc157593155)

[Configuration 3](#_Toc157593156)

[Code APIs And Helper Methods 4](#_Toc157593157)

[APIs 4](#_Toc157593158)

[addManually 4](#_Toc157593159)

[addFromFile 4](#_Toc157593160)

[printStudentByID 4](#_Toc157593161)

[printByFirstName 4](#_Toc157593162)

[printCourseStudents 4](#_Toc157593163)

[printAll 4](#_Toc157593164)

[deleteByID 4](#_Toc157593165)

[updateByID 4](#_Toc157593166)

[countStudents 5](#_Toc157593167)

[HelperMethods’ 5](#_Toc157593168)

[isUniqueID 5](#_Toc157593169)

[getStudentByID 5](#_Toc157593170)

[printStudentInfo 5](#_Toc157593171)

[FIFO\_validator 5](#_Toc157593172)

[enterStudentInfo 5](#_Toc157593173)

[Tests 5](#_Toc157593174)

[Add students from file: 6](#_Toc157593175)

[Print All Students 7](#_Toc157593176)

[Add Student Manually 7](#_Toc157593177)

[Print Student by ID 8](#_Toc157593178)

[Print Student By First Name 8](#_Toc157593179)

[Print Students In Course 9](#_Toc157593180)

[Delete Student By ID 10](#_Toc157593181)

[Update Student By ID 10](#_Toc157593182)

[Get Total Number Of Students 11](#_Toc157593183)

# Problem Statement

Write a program to build a simple Software for Student Information Management

System which can perform the following operations:

● Store the First name of the student.

● Store the Last name of the student.

● Store the unique Roll number for every student.

● Store the GPA of every student.

● Store the courses registered by the student.

# Approach

The idea is to form an individual functions for every operation. All the functions are

unified together to form software.

● Add Student Details From File

● Add Student Details Manually

● Find the student by the given roll number

● Find the student by the given first name

● Find the students registered in a course

● Count of students

● Delete a student

● Update Student

# Configuration

In students.h, there are some config options to se

// //////////////////////////////////////////

// Configuration

// //////////////////////////////////////////

#define BUFFER\_LENGTH 50

#define MAX\_NUM\_OF\_COURSES 10

#define MAX\_NAME\_CHARACTERS 50

#define FILENAME "studentsDB.txt"

BUFFER\_LENGTH:

Sets the maximum number of students to save in the DB

MAX\_NUM\_OF\_COURSES:

Sets the maximum No. of courses each single student can take

MAX\_NAME\_CHARACTERS:

Sets the maximum first/last name No. of characters

# Code APIs And Helper Methods

Github Code: [https://github.com/Kamelz70/LID\_Diploma/tree/main/First\_Term\_Projects/Project\_2](%20https:/github.com/Kamelz70/LID_Diploma/tree/main/First_Term_Projects/Project_2)

## APIs

These are the main APIs used to provide functionality

### addManually

void addManually(FIFO\_Buf\_t \*S\_FIFO);

This API adds a student manually, it uses helper methods of **FIFO\_validator,** **enterStudentInfo,** **countStudents**

### addFromFile

void addFromFile(FIFO\_Buf\_t \*S\_FIFO);

This API adds students from an existing text file, it uses helper methods of **FIFO\_validator** **,isUniqueID**

### printStudentByID

void printStudentByID(FIFO\_Buf\_t \*S\_FIFO);

This API prints a student by ID, it uses helper methods of **FIFO\_validator,** **getStudentByID,** **printStudentInfo**

### printByFirstName

void printByFirstName(FIFO\_Buf\_t \*S\_FIFO);

This API prints a student by ID, it uses helper methods of **FIFO\_validator,** **printStudentInfo**

### printCourseStudents

void printCourseStudents(FIFO\_Buf\_t \*S\_FIFO);

This API prints students taking a certain course ID, it uses helper methods of **FIFO\_validator,** **printStudentInfo**

### printAll

void printAll(FIFO\_Buf\_t \*S\_FIFO);

This API prints All students, it uses helper methods of **FIFO\_validator,** **printStudentInfo**

### deleteByID

void deleteByID(FIFO\_Buf\_t \*S\_FIFO);

This API deletes a student by ID, it uses helper methods of **FIFO\_validator,** **printStudentInfo,** **getStudentByID**

### updateByID

void updateByID(FIFO\_Buf\_t \*S\_FIFO);

This API updates a student by ID, it uses helper methods of **FIFO\_validato,** **getStudentByID,** **enterStudentInfo**

### countStudents

void countStudents(FIFO\_Buf\_t \*S\_FIFO);

This API counts all students, it uses the helper method **FIFO\_validator**

## HelperMethods

These are the helper methods used by the main APIs to enhance code readability/maintainability

### isUniqueID

bool isUniqueID(FIFO\_Buf\_t \*S\_FIFO, uint32 ID);

This method is called to check if a given ID input is unique or not, it uses the helper method **getStudentByID**

### getStudentByID

Student \*getStudentByID(FIFO\_Buf\_t \*S\_FIFO, uint32 ID, int \*studentListingNumP);

This method iterates on student IDs to return a pointer to the student with the ID, if not found, it returns NULL, it uses the helper method **FIFO\_validator**

### printStudentInfo

void printStudentInfo(Student \*s);

This method prints a given student’s information

### FIFO\_validator

bool FIFO\_validator(FIFO\_Buf\_t \*S\_FIFO, bool isEmptyValidate, bool isFullValidate);

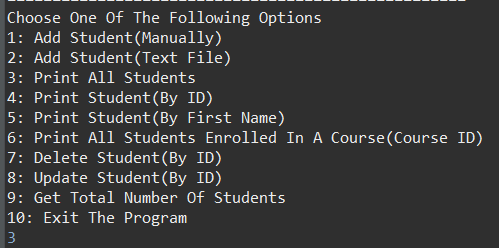
This method is used to validate the FIFO buffer and its’ data to check if it’s invalid, empty, or full

### enterStudentInfo

void enterStudentInfo(FIFO\_Buf\_t \*S\_FIFO, Student \*targetStudent, uint8 choice);

This method is used to enter student data and save it the student pointer

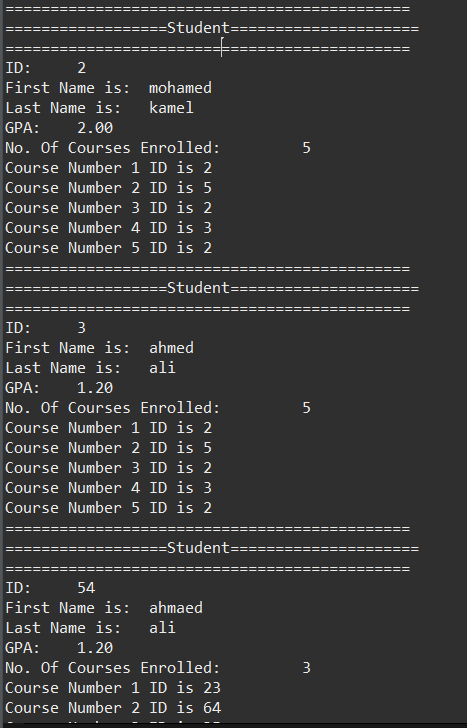
# Tests

These are the options which are presented to the user upon starting the program

## Add students from file:

While reading each line, any non-unique ID entry is skipped and the program continues saving from the next line

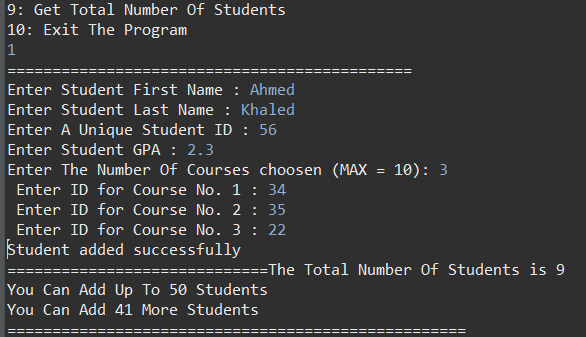
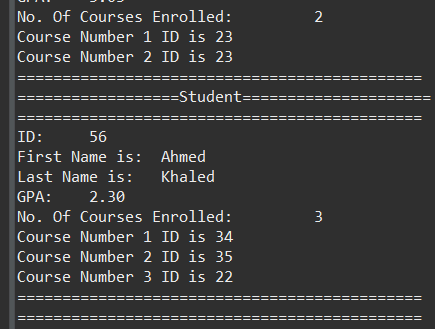
## Print All Students



The Students added from the previous file are present when choosing option 3 (PrintAll)

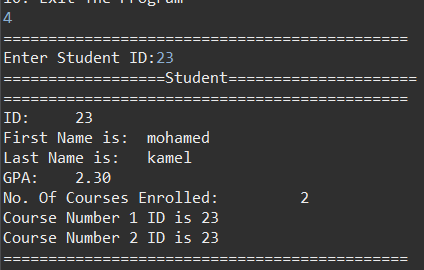
## Add Student Manually

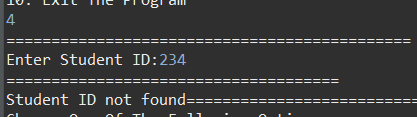
The left screenshot show the process of adding the student

, and the right one shows it’s been added by printing all students

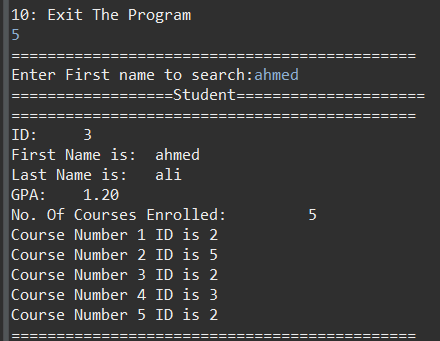
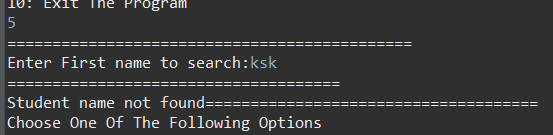
## Print Student by ID

From the above added students, searching for a student with an ID prints the target correctly



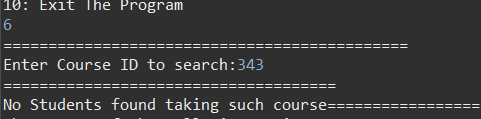
Non-existing IDs print the following

## Print Student By First Name

This also works by entering the name, then the found student is printed, entering a non-existing name is shown in the right screenshot

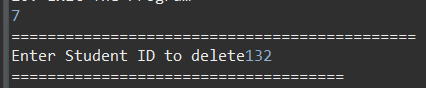
## Print Students In Course

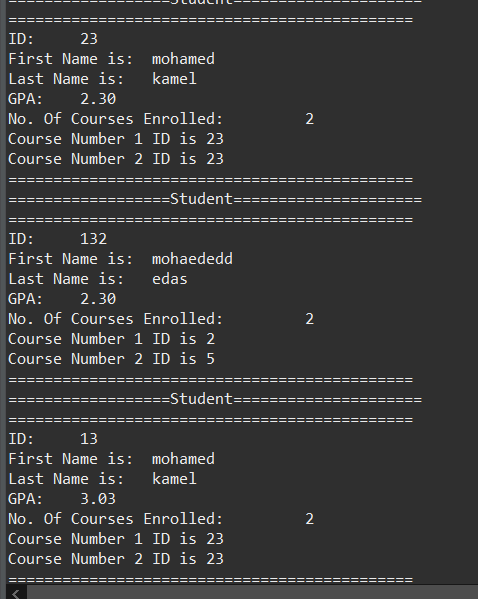
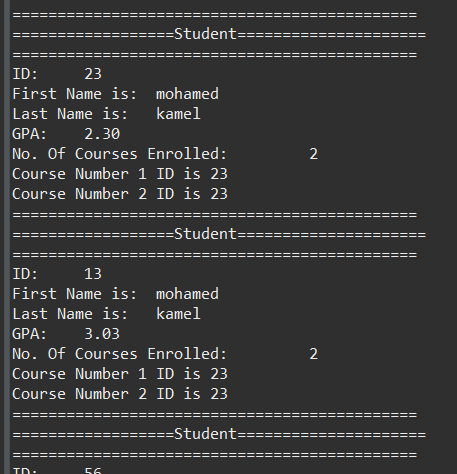
The students taking course No 33 are all printed as intended

Entering a non-existing course will print the following message:

## Delete Student By ID

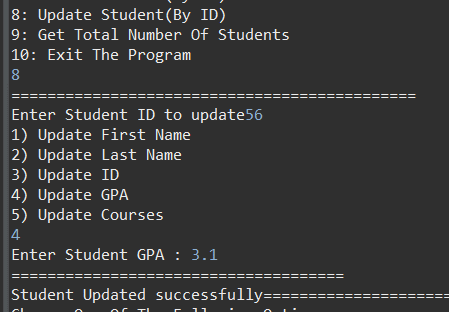
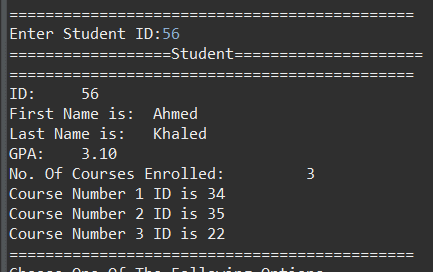
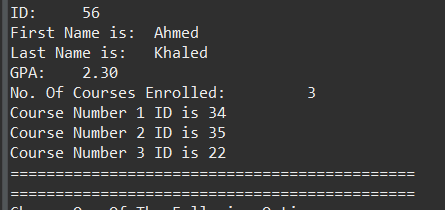
In this test, we delete the student with the ID 132, the lower screenshots show the DB before and after the deletion



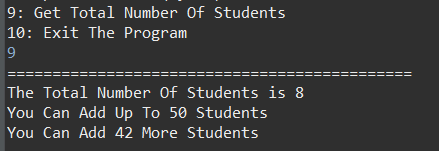


## Update Student By ID

The left-most Screen shows a student we will update, upon entering student ID, we choose what we want to update, the right-most screen shows that the student has been updated successfully



## Get Total Number Of Students

This functionality shows the number of students in our database