**Mastering Embedded System Online Diploma**

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

**First Term (Final Project 2 )**

**Eng. Mostafa AboSalama**

**My Profile:** [My Progress Page](https://www.learn-in-depth-store.com/certificate/mostafaabosalama661%40gmail.com)

## ****1. Introduction****

### ****1.1 Project Overview****

The Student Database Management System is designed to manage student records efficiently. It allows operations such as adding, updating, deleting, and viewing student information. The system can handle student data including names, roll numbers, GPA, and course enrollments.

### ****1.2 Objectives****

* To develop a system that maintains student records.
* To enable operations like adding, updating, viewing, and deleting student information.
* To handle student data using both array-based and linked list-based queue implementations.

## ****2. System Design****

### ****2.1 System Architecture****

The system comprises:

* A queue data structure to store student records.
* A command-line interface for user interaction.
* Functions for managing student records including addition, deletion, and updates.

### ****2.2 Components****

* **Queue Implementation**: Supports operations for adding and removing student records. Two implementations are provided: array-based and linked list-based.
* **Student Information**: Includes fields for first name, last name, roll number, GPA, and course IDs.
* **File Handling**: Functions to load student data from a file into the queue.

### ****2.3 Data Structures****

* **Queue**: Manages student records with functionalities for enqueueing and dequeueing.
* **StudentInfo\_t**: Structure holding student details such as name, roll number, GPA, and course IDs.

## ****3. Implementation Details****

### ****3.1 Code Structure****

* **Queue.c**: Implements queue operations including initialization, enqueue, dequeue, and checks for full or empty queues.
* **main.c**: Provides the user interface for interacting with the student database system. It handles user choices and invokes appropriate functions.
* **Queue.h**: Header file declaring functions and data structures used in Queue.c.
* **FileHandling.h**: Includes functions for reading from and writing to files.

### ****3.2 Key Functions****

* **Queue\_enumQueueErrorInit**: Initializes the queue.
* **Queue\_enumQueueErrorEnqueue**: Adds a student record to the queue.
* **Queue\_enumQueueErrorDequeue**: Removes a student record from the queue.
* **AddStudent**: Adds a student to the queue if not already present.
* **AddFromFile**: Reads student data from a file and adds it to the queue.
* **ViewStudentWithRoll**: Displays student information based on roll number.
* **ViewStudentWithFname**: Displays student information based on first name.
* **ViewStudentsInCourse**: Lists students enrolled in a specific course.
* **DeleteStudent**: Removes a student from the queue based on roll number.
* **UpdateStudent**: Updates student information.

## ****4. User Guide****

### ****4.1 How to Run****

1. **Compile the Code**: Open the CMD from the main directory , copy this line and paste it.

**gcc 4\_App/main.c 3\_DataStructure/Queue.c 2\_FileHandling/FileHandling.c -o main.exe**

1. **Execute the Program**: Run the executable to interact with the student database system.

**main.exe**

### ****4.2 Available Options****

* **Add Student From File**: Load student data from a file and add to the system.
* **Add Student Manually**: Enter student details manually.
* **View Student Using Roll Number**: Retrieve student information based on roll number.
* **View Student Using First Name**: Retrieve student information based on first name.
* **View Students in A Course**: List students enrolled in a specific course.
* **Count Number Of Students**: Display the total number of students.
* **Delete Student**: Remove a student record from the system.
* **Update Student**: Modify existing student information.
* **View All Students**: Display all student records.

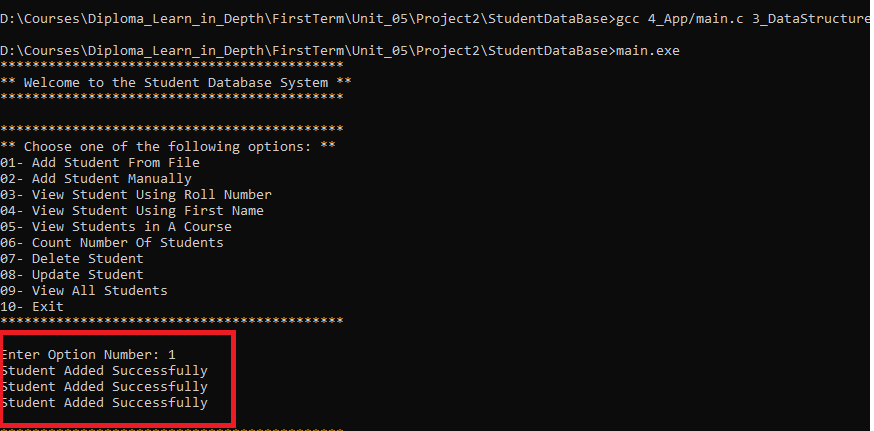
## ****5. Testing and Results****

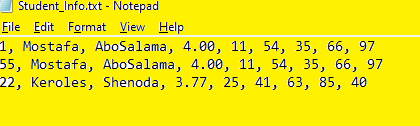
### ****5.1 Test Cases****

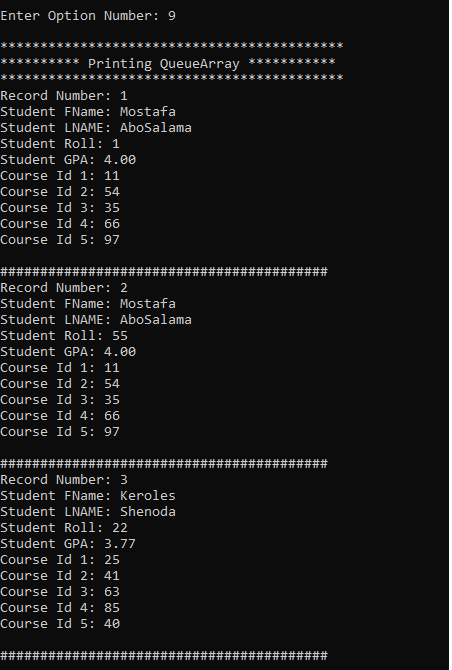
* **Test Case 1**: Add students from file and verify if the records are correctly added.
* **Test Case 2**: Add a student manually and verify if the record is correctly added
* **Test Case 3**: View a student using his roll number and check if details are correct.
* **Test Case 4**: View a student using his first name and check if details are correct.
* **Test Case 5**: View students enrolled in a course and check if details are correct.
* **Test Case 6**: Count the number of the students in the database .
* **Test Case 7**: Delete a student and ensure the record is removed from the queue.
* **Test Case 8**: Update student information and verify changes.
* **Test Case 9**: View all students and ensure correct listing.

### ****5.2 Results****

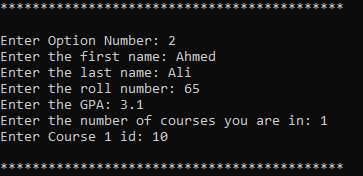
* **Add Student From A File**: Successfully added student records to the queue.

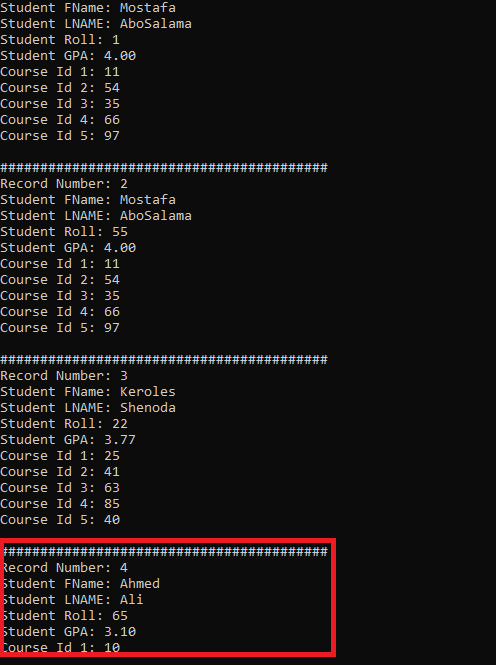




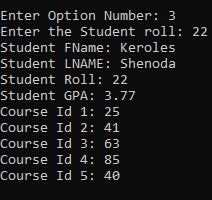


* **Add Student Manually**: Correctly added student records to the queue.

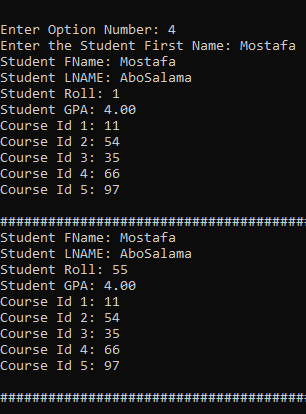




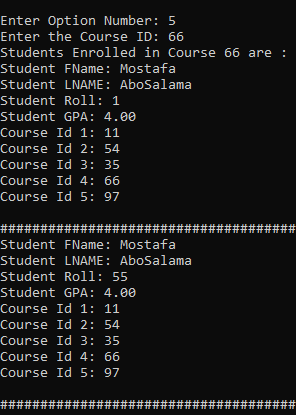
* **View Student By Roll Number**: Correctly view student records.



* **View Student By First Name**: Correctly view student records.



* **View Students In A Specific Course**: Correctly view students records.

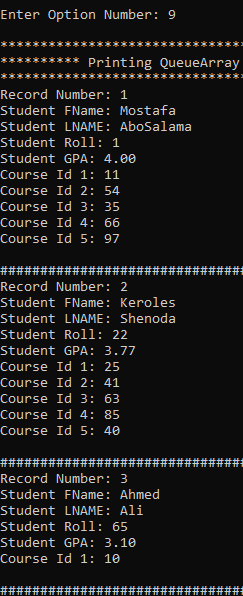


* **Count Number Of The Students**: Correctly count the number.

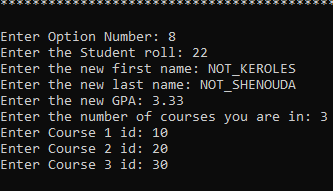


* **Delete Student**: Correctly remove the student from the database.





* **Update Student**: Correctly update the student record in the database.



* **View All Students**: Correctly view all students records.

