Student Management System

Michael Maged William

Learn in Depth

—

Mastering Embedded System Diploma

—

Eng/Keroloes Shenouda

Project Overview

The **Student Management System** is designed to efficiently manage student records using a FIFO (First In, First Out) buffer structure. The system allows users to perform various operations, such as adding, searching, updating, deleting, and displaying student data. The project is implemented in C, and it handles operations both manually and from a file input

System Structure

Heading 3 main files

`Student_Management_System.c`

This file contains the core logic for handling student management operations. It defines and manages all functions related to the FIFO buffer, such as adding a new student, searching, updating, deleting, and displaying student records.

`main.c`

This file serves as the main entry point of the system. It handles the user interface logic and interacts with the user to receive commands. It presents multiple options such as adding students,

`Student_Management_System.h`

This header file contains the essential definitions and structure for the system. It includes structure definitions for student data and the FIFO buffer, as well as

searching for them, updating or deleting student records, and displaying records. The commands are then forwarded to the appropriate functions for execution.

prototypes for all functions used in the system.

Detailed Report on Functions in Each File

main.c

This file contains the main control logic of the program and manages user input, presenting options to interact with the student management system.

1. `int main()`

- **Purpose:** Acts as the entry point of the program, initializing the FIFO buffer, displaying the menu options, and handling user input.
- **Detailed Explanation:**
  - Initializes the FIFO buffer using `FIFO_init` with a size capable of holding 50 students.
  - The function displays a menu of operations using a `while(1)` loop, allowing continuous interaction.
  - A `switch` statement handles different menu choices:
    - **Case 1:** Adds a student manually via `Add_Student_Manually`.
    - **Case 2:** Adds students from a file via `Add_Student_From_File`.
    - **Case 3:** Finds a student by roll number using `Find_Student_By_ID`.
    - **Case 4:** Finds a student by name using `Find_Student_By_First_Name`.

- **Case 5:** Shows students in a specific course using `Find_NO_Of_Student_IN_Course`.
- **Case 6:** Deletes a student by roll number via `Delete_Student`.
- **Case 7:** Updates student details via `Update_Student_Info`.
- **Case 8:** Displays all students via `Show_Student`.
- **Case 9:** Displays the total number of students via `Find_Total_Number`.

  - The loop continues until the user opts to exit.

2. `void display_menu()`

- **Purpose:** Provides the list of available commands for the user to choose from.
- **Detailed Explanation:**
  - It prints out the various options for adding, finding, deleting, and updating students, among others. This makes it easier for the user to interact with the system

```c
/*
 * main.c
 *
 *  Created on: Sep 18, 2024
 *      Author: Michael Naged
 */
#include "Student_Managment_System.h"
#include "stdio.h"
#include "string.h"

Sdata_t  buf[50];
int main()
{
FIFO_buf_t Fifo_data;
int choice;
FIFO_init(&Fifo_data,buf,50);
Dprintf("Welcome to the Student management System \n");
    printf("\n****************************************\n");
    while(1)
    {
        Dprintf("Choose the task that you want perform \n");
        Dprintf("\t1. Add the Student Details Manually \n");
        Dprintf("\t2. Add the Student Details From Text File \n");
        Dprintf("\t3. Find the Student Details by Roll Number \n");
        Dprintf("\t4. Find the Student Details by First Name \n");
        Dprintf("\t5. Find the Student Details by Course ID \n");
        Dprintf("\t6. Find the Total Number Of Students \n");
        Dprintf("\t7. Delete the Students Details by Roll Number \n");
        Dprintf("\t8. Update the Students Details by Roll Number \n");
        Dprintf("\t9. Show All Information \n");
        Dprintf("\t10. To Exit \n");
        Dprintf("\nEnter your choice to perform the task :");
        Dscanf(&choice,"%d");
        switch(choice)
        {
        case 1:
            Dprintf("\n****************************************\n");
```

```c
switch(choice)
{
case 1:
    Dprintf("\n****************************************\n");
    Add_Student_Manually(&Fifo_data);
    Dprintf("\n****************************************\n");
    break;
case 2:
    Dprintf("\n****************************************\n");
    Add_Student_From_File(&Fifo_data);
    Dprintf("\n****************************************\n");
    break;
case 3:
    Dprintf("\n****************************************\n");
    Find_Student_By_ID(&Fifo_data);
    Dprintf("\n****************************************\n");
    break;
case 4:
    Dprintf("\n****************************************\n");
    Find_Student_By_First_Name(&Fifo_data);
    Dprintf("\n****************************************\n");
    break;
case 5:
    Dprintf("\n****************************************\n");
    Find_NO_Of_Student_IN_Course(&Fifo_data);
    Dprintf("\n****************************************\n");
    break;
case 6:
    Find_Total_Number(&Fifo_data);
    Dprintf("\n****************************************\n");
    break;
case 7:
    Dprintf("\n****************************************\n");
    Delete_Student(&Fifo_data);
    Dprintf("\n****************************************\n");
    break;
case 8:
```

Student_Managment_System.h

Overview:

This header file contains the declarations, macros, and structure definitions necessary for the implementation of the Student Management System. It ensures that the functions defined in `Student_Managment_System.c` are accessible to the rest of the program.

Key Components:

1. **Structures:**
   - **`Sdata_t`:** Defines the structure for storing student data, including fields for the student's first and last name, ID, GPA, and an array for the course IDs.
   - **`FIFO_buf_t`:** Defines the structure for the FIFO buffer, which holds the student data and tracks the buffer's state (head, tail, base, length, and counter).
2. **Macros:**
   - Defines useful macros for printing (`Dprintf`) and scanning (`Dscanf`) data, likely acting as wrappers around `printf` and `scanf` for debugging or formatted output.
3. **Function Prototypes:**
   - Provides the declarations for all functions used in the system, such as `FIFO_init`, `Add_Student_From_File`, `Add_Student_Manually`, `Find_Student_By_ID`, and others. These prototypes ensure that the main program can call these functions even though their implementation is in a different file.
4. **Enum `FIFO_buf_status`:**
   - An enumeration defining possible states for the FIFO buffer, such as `FIFO_no_error`, `FIFO_full`, and `FIFO_null`. This enum is used throughout the program to manage the status of buffer operations.

```
 * Student_Managment_System.h
 *
 *  Created on: Sep 18, 2024
 *      Author: Michael Maged
 */

#ifndef STUDENT_MANAGMENT_SYSTEM_H_
#define STUDENT_MANAGMENT_SYSTEM_H_
#include "stdio.h"
#include "stdlib.h"
#include "stdint.h"

#define Dprintf(...) {fflush(stdin);fflush(stdout);\
        printf(__VA_ARGS__);\
        fflush(stdin);fflush(stdout);}

#define Dscanf(val,...) {fflush(stdin);fflush(stdout);\
        scanf(__VA_ARGS__,val);\
        fflush(stdin);fflush(stdout);}
// Defination of struct contains student information
/* student information will be 1- first name
 *                              2- last name
 *                              3- unique roll number (ID)
 *                              4- courses ID
 *                              5- GPA
 */
typedef struct
{
    char First_name[10];
    char last_name[10];
    int ID;
    int Course[5];
    float GPA;
}Sdata_t;
//FIFO data types
typedef struct {
```

```
//API to store the data in fifo
FIFO_buf_status FIFO_Enqueue(FIFO_buf_t* fifo , Sdata_t item);

// API to add student info manually to our FIFO : we will recive the FIFO
void Add_Student_Manually( FIFO_buf_t* fifo_buf );

// API to search and find the student information by roll number
void Find_Student_By_ID( FIFO_buf_t* fifo_buf );

// API to search and find the student information by first name
void Find_Student_By_First_Name( FIFO_buf_t* fifo_buf );

//API to find the number of student incolled in course using course ID
void Find_NO_Of_Student_IN_Course(FIFO_buf_t* fifo_buf );

// API to count the total number of student
void Find_Total_Number( FIFO_buf_t* fifo_buf );

// API to delete student information from the system
void Delete_Student( FIFO_buf_t* fifo_buf );

// API to update student information in the system the system
void Update_Student_Info( FIFO_buf_t* fifo_buf );

// API to show all students information in the system
void Show_Student( FIFO_buf_t* fifo_buf );

// API to check if FIFO is full
FIFO_buf_status FIFO_IS_FULL( FIFO_buf_t *fifo_buf );

// API to check if FIFO is empty
FIFO_buf_status FIFO_EMPTY( FIFO_buf_t* fifo_buf );

// API to check if ID is Repeated or no
FIFO_buf_status Check_ID(FIFO_buf_t* fifo_buf,Sdata_t item);
#endif /* STUDENT_MANAGMENT_SYSTEM_H_ */
```

```c
    float GPA;
}Sdata_t;
//FIFO data types
typedef struct {
    uint32_t length; //the size of buffer
    uint32_t counter;//to increment the number of value it assign in buffer
    Sdata_t* head ;//to increment the location in buffer to assign the next value
    Sdata_t* base ;//the statrt address of buffer
    Sdata_t* tail ;//to pop value and empty location
}FIFO_buf_t;

//create enum to return the status of buffer
typedef enum{
    FIFO_no_error,//0
    FIFO_full,//1
    FIFO_empty,//2
    FIFO_null,//3
    FIFO_error//4
}FIFO_buf_status;

// Prototypes of system API's

// API to initialize items to our FIFO : we will recive the FIFO, the array which will store in , and the lenth
void FIFO_Init_Items( FIFO_buf_t* fifo_buf , Sdata_t* buf , int length );

// API to add students info from file to our FIFO : we will recive the FIFO
void Add_Student_From_File( FIFO_buf_t* fifo_buf);

//API to store the data in fifo
FIFO_buf_status FIFO_Enqueue(FIFO_buf_t* fifo , Sdata_t item);

// API to add student info manually to our FIFO : we will recive the FIFO
void Add_Student_Manually( FIFO_buf_t* fifo_buf );

// API to search and find the student information by roll number
void Find_Student_By_ID( FIFO_buf_t* fifo_buf );
```

Student_Managment_System.c

This file implements the core functionalities for managing student records, including adding, finding, and deleting students, among other tasks.

1. `FIFO_buf_status FIFO_init(FIFO_buf_t* fifo, Sdata_t* buffer, unsigned int length)`

- **Purpose:** Initializes the FIFO buffer by setting pointers and length.
- **Detailed Explanation:**
  - Sets the `base`, `head`, and `tail` pointers of the FIFO buffer to point to the provided `buffer`.
  - Sets the `length` to the number of student records the buffer can hold and initializes `count` to 0.
  - Returns `FIFO_no_error` if initialization is successful.

2. `FIFO_buf_status Add_Student_From_File(FIFO_buf_t* fifo, const char* filename)`

- **Purpose:** Reads student data from a text file and adds them to the FIFO buffer.
- **Detailed Explanation:**
  - Opens the file in read mode and checks for file existence.
  - Loops through the file and reads student details (first name, last name, roll number, GPA, and course IDs).
  - For each student, it checks if the buffer is full using `FIFO_full`, and if not, adds the student to the FIFO using `FIFO_enqueue`.
  - Handles errors like incomplete data and buffer overflow.
  - Returns success or failure status based on the number of students added.

3. `FIFO_buf_status Add_Student_Manually(FIFO_buf_t* fifo)`

- **Purpose:** Adds a student's details to the FIFO buffer manually.
- **Detailed Explanation:**
  - Asks the user to input student details (first name, last name, roll number, GPA, and course IDs).
  - Verifies if the roll number is unique by checking existing entries in the buffer.
  - Adds the student to the FIFO buffer using `FIFO_enqueue`.
  - Returns a status indicating success or failure based on buffer capacity and uniqueness of the roll number.

4. `FIFO_buf_status Find_Student_By_ID(FIFO_buf_t* fifo, int id)`

- **Purpose:** Finds a student by their roll number (ID).
- **Detailed Explanation:**
  - Iterates through the FIFO buffer looking for a matching roll number.
  - If found, it displays the student's details (first name, last name, GPA, and course IDs).
  - Returns a status indicating whether the student was found or not.

5. `FIFO_buf_status Find_Student_By_First_Name(FIFO_buf_t* fifo, const char* name)`

- **Purpose:** Searches for students by their first name.
- **Detailed Explanation:**
  - Performs a case-insensitive comparison between the input name and the first name of each student in the buffer.
  - If a match is found, displays the student's details.
  - Returns a status indicating whether any students with the given name were found.

6. `FIFO_buf_status Find_NO_Of_Student_IN_Course(FIFO_buf_t* fifo, int course_id)`

- **Purpose:** Finds and displays the number of students enrolled in a specific course.
- **Detailed Explanation:**
  - Iterates through all student records in the buffer and checks if they are enrolled in the course with the given `course_id`.
  - Displays the details of all students taking the course and counts how many are enrolled.
  - Returns a status indicating the success of the operation.

7. `FIFO_buf_status Delete_Student(FIFO_buf_t* fifo, int id)`

- **Purpose:** Deletes a student from the FIFO buffer based on their roll number (ID).
- **Detailed Explanation:**
  - Searches for the student with the given roll number.
  - If found, it removes the student by shifting all subsequent entries forward to maintain buffer order.
  - Returns a status indicating success or failure.

8. `FIFO_buf_status Update_Student_Info(FIFO_buf_t* fifo, int id)`

- **Purpose:** Updates the details of an existing student.
- **Detailed Explanation:**
  - Searches for the student with the given roll number and allows the user to modify details like name, GPA, and course IDs.
  - Ensures that no other student has the same roll number after the update.
  - Returns a status indicating success or failure.

9. `FIFO_buf_status Show_Student(FIFO_buf_t* fifo)`

- **Purpose:** Displays the details of all students in the FIFO buffer.
- **Detailed Explanation:**
  - Iterates through the buffer and prints each student's details, including their first name, last name, roll number, GPA, and courses.
  - Returns a status indicating the success of the operation.

10. `FIFO_buf_status Find_Total_Number(FIFO_buf_t* fifo)`

- **Purpose:** Displays the total number of students in the buffer and the number of available spots left.
- **Detailed Explanation:**

- Simply prints the total number of students (`fifo->count`) and the available capacity (`fifo->length - fifo->count`)

```c
/*
 * Student_Managment_System.c
 *
 *  Created on: Sep 18, 2024
 *      Author: Michael Maged
 */
#include "Student_Managment_System.h"
#include "stdio.h"
#include "string.h"
//defination the APIs
FIFO_buf_status FIFO_init(FIFO_buf_t* fifo , Sdata_t* buf, uint32_t length)
{
    //to check the FIFO is existed or no
    if(buf==NULL)
        return FIFO_null;
    //to intialize the FIFO
    fifo->base=buf;
    fifo->head=buf;
    fifo->tail=buf;
    fifo->length=length;
    fifo->counter=0;
    return FIFO_no_error;
}
/*============================================================================================*/

void Add_Student_From_File( FIFO_buf_t* fifo_buf)
{
    int i =0;//to count number of student is stored
    FIFO_buf_status check;//to check the state of fifo
    if( FIFO_IS_FULL(fifo_buf)==FIFO_full)
    {
        printf("[ERORR] FIFO is FULL \n");
        return ;
    }
    else if(FIFO_IS_FULL(fifo_buf)==FIFO_null)
    {
        printf("[ERORR] FIFO is NOT Exist \n");
```

```c
        printf("[ERORR] FIFO is NOT Exist \n");
        return ;
    }
    else
    {
        Sdata_t S_temp;//to store data and send it to the enqueue to store in head
        int result=0;//to check the information is complete for each student
        FILE *P_file;   //define the pointer from type file to read data from file

        P_file = fopen("Student_Data.txt","r");
        // Check if file is not opening

        if(P_file==NULL)
        {
            printf("Error! in opening file. \n");
            return ;
        }
        // Check if file is empty
        if (fscanf(P_file,"%d",&S_temp.ID)!=1)//this function is return non_zero if the file is empty "must be Try reading at least one"
        {
            if(feof(P_file)){
                printf("The file is empty.\n");
                fclose(P_file);
                return;
            }
        }

        while((result=fscanf(P_file,"%s %s %d %f %d %d %d %d %d",
                S_temp.First_name,
                S_temp.last_name,
                &S_temp.ID,
                &S_temp.GPA,
                &S_temp.Course[0],
                &S_temp.Course[1],
                &S_temp.Course[2],
                &S_temp.Course[3],
                &S_temp.Course[4]))==9)
```

```
            check=FIFO_Enqueue(fifo_buf, S_Temp);
            if(check==FIFO_no_error)
            {
                printf("Student[%d] Details is added Successfully.\n",i);
            }
            else if(check==FIFO_full)//if between store the list is full before the data of all file is stored
            {
                printf("[Error] the list is Full\n");
                return ;
            }
        }
        if (result != EOF) {
            printf("Warning: Incomplete data found for a student.\n");
            return ;
        }
        fclose(P_file);
    }

    Find_Total_Number(fifo_buf);

}
/*=============================================================================================*/
void Add_Student_Manually( FIFO_buf_t* fifo_buf )
{
    FIFO_buf_status check_M;//to check the state of fifo
    Sdata_t S_Student;

    Dprintf("\nEnter the Roll Number: ");
    scanf("%d",&S_Student.ID);
    Dprintf("\nEnter the First Name: ");
    scanf("%s",S_Student.First_name);
    Dprintf("\nEnter the Last Name: ");
    scanf("%s",S_Student.last_name);
    Dprintf("\nEnter the GPA: ");
    scanf("%f",&S_Student.GPA);
    for(int i=0; i<5;i++)
    {
```

```
    {
        Dprintf("\nCourse %d id: ",i+1);
        scanf("%d",&S_Student.Course[i]);
    }
    check_M=FIFO_Enqueue(fifo_buf, S_Student);
    if(check_M==FIFO_no_error)
    {
        printf("Student Details is added Successfully. \n");
    }
    else if(check_M==FIFO_full)//if between store the list is full before the data of all file is stored
    {
        printf("[Error] the list is Full\n");
        return ;
    }

    Find_Total_Number(fifo_buf);
}

/*=============================================================================================*/
void Find_Student_By_ID( FIFO_buf_t* fifo_buf )
{   int ID;
Sdata_t* check_Student = fifo_buf->tail;
Dprintf("\nEnter the Roll Number of the Student: ");
scanf("%d",&ID);
for (int i = 0; i < fifo_buf->counter; ++i)
{
    if (check_Student->ID == ID)
    {
        Dprintf("\nThe First Name: %s",check_Student->First_name);
        Dprintf("\nThe Last Name: %s",check_Student->last_name);
        Dprintf("\nThe GPA: %.2f",check_Student->GPA);
        Dprintf("\nThe Course ID are: %d",check_Student->Course[0]);
        Dprintf("\nThe Course ID are: %d",check_Student->Course[1]);
        Dprintf("\nThe Course ID are: %d",check_Student->Course[2]);
        Dprintf("\nThe Course ID are: %d",check_Student->Course[3]);
        Dprintf("\nThe Course ID are: %d",check_Student->Course[4]);
        return ;
```

```
        return ;
    }
    check_Student = (check_Student + 1); // move to next element
    if (check_Student == (fifo_buf->base + fifo_buf->length * sizeof(Sdata_t)))
        check_Student = fifo_buf->base; //if fifo is reach to top fifo
}

Dprintf("[Error] Roll Number %d not found \n",ID);
}
/*=============================================================================================*/
void Find_Student_By_First_Name( FIFO_buf_t* fifo_buf )
{
    int found_student=0;//If the student finds it, he will let the program not print "[Error] the First Name not found"
    char check_name[10];
    Sdata_t* check_Student = fifo_buf->tail;
    Dprintf("\nEnter the First Number of the Student: ");
    scanf("%s",check_name);
    for (int i = 0; i < fifo_buf->counter; ++i)
    {
        if (stricmp(check_Student->First_name,check_name)==0)//using stricmp Because it is possible to enter the first name small letter
        {
            Dprintf("\nThe ID: %d",check_Student->ID);
            Dprintf("\nThe First Name: %s",check_Student->First_name);
            Dprintf("\nThe Last Name: %s",check_Student->last_name);
            Dprintf("\nThe GPA: %.2f",check_Student->GPA);
            Dprintf("\nThe Course ID are: %d",check_Student->Course[0]);
            Dprintf("\nThe Course ID are: %d",check_Student->Course[1]);
            Dprintf("\nThe Course ID are: %d",check_Student->Course[2]);
            Dprintf("\nThe Course ID are: %d",check_Student->Course[3]);
            Dprintf("\nThe Course ID are: %d",check_Student->Course[4]);
            Dprintf("\n================================================");
            found_student=1;
        }
        check_Student = (check_Student + 1); // move to next element
        if (check_Student == (fifo_buf->base + fifo_buf->length * sizeof(Sdata_t)))
            check_Student = fifo_buf->base; //if fifo is reach to top fifo
    }
```

```
    }
    //if student not found
    if(found_student==0)
    {
        Dprintf("[Error] the First Name: %s not found \n",check_name);
    }
}
/*=============================================================================================*/
void Find_NO_Of_Student_IN_Course(FIFO_buf_t* fifo_buf )
{
    Sdata_t* check_no_Student = fifo_buf->tail;
    int check_course_id=0;
    int number_students=0;
    Dprintf("\nEnter the Course ID: ");
    scanf("%d",&check_course_id);
    for(int i=0; i<fifo_buf->counter;i++)
    {
        for(int j=0;j<5;j++)
        {
            if ( check_course_id == check_no_Student->Course[j] )
            {
                Dprintf("\nThe Student Details are: ");
                Dprintf("\nThe First Name: %s",check_no_Student->First_name);
                Dprintf("\nThe Last Name: %s",check_no_Student->last_name);
                Dprintf("\nThe Roll Number is %d",check_no_Student->ID);
                Dprintf("\nThe GPA: %.2f",check_no_Student->GPA);
                Dprintf("\n================================================");
                number_students+=1;
            }
        }
        check_no_Student = (check_no_Student + 1); // move to next element
        if (check_no_Student == (fifo_buf->base + fifo_buf->length * sizeof(Sdata_t)))
            check_no_Student = fifo_buf->base; //if fifo is reach to top fifo
    }
    Dprintf("\nTotal Number of Student Enrolled: %d",number_students);
}
/*=============================================================================================*/
```

```
    Dprintf("\nTotal Number of Student Enrolled: %d",number_students);
}
/*=============================================================================================*/
void Find_Total_Number( FIFO_buf_t* fifo_buf )
{

    Dprintf("\n*****************************************\n");
    Dprintf("[INFO] The total number of student is %d\n",fifo_buf->counter);
    Dprintf("[INFO] You can add up to %d students\n",50);
    Dprintf("[INFO] You can add %d more students\n",50-(fifo_buf->counter));

}
/*=============================================================================================*/
void Delete_Student( FIFO_buf_t* fifo_buf )
{
    if(!fifo_buf->base || !fifo_buf->head || !fifo_buf->tail)
    {
        Dprintf("[Error]!The list is not exist");
        return;
    }
    //check FIFO is empty ?
    if(fifo_buf->counter==0)
    {
        Dprintf("[Error]!The list is empty");
        return;
    }

    Sdata_t* current_student = fifo_buf->tail;
    int delete_student_id;
    Dprintf("\nEnter the Roll Number which you want to delete: ");
    scanf("%d", &delete_student_id);

    int found = 0; // to track if student is found
    for (int i = 0; i < fifo_buf->counter; ++i)
    {
        if (current_student->ID == delete_student_id)
```

```
        if (current_student->ID == delete_student_id)
        {
            found = 1;
            break;
        }
        current_student++;
        if (current_student == (fifo_buf->base + fifo_buf->length * sizeof(Sdata_t)))
        {
            current_student = fifo_buf->base; // Circular FIFO
        }
    }

    if (!found)
    {
        Dprintf("[Error] Roll Number %d not found\n", delete_student_id);
        return;
    }

    // Shift all elements after the deleted one
    Sdata_t* next_student = current_student + 1;
    if (next_student == (fifo_buf->base + fifo_buf->length))
    {
        next_student = fifo_buf->base; // Circular FIFO
    }

    //to rearrange the list of student after delete id
    while (next_student != fifo_buf->head)
    {
        *current_student = *next_student; // shift the student
        current_student = next_student;//update adress of current_student to point the next element
        next_student++; //move to next element
        if (next_student == (fifo_buf->base + fifo_buf->length * sizeof(Sdata_t)))
        {
            next_student = fifo_buf->base; // Circular FIFO
        }
    }

```

```
    // Adjust head and counter
    if (fifo_buf->head == fifo_buf->base)
    {
        fifo_buf->head = fifo_buf->base + fifo_buf->length-1 ;
    }
    else
    {
        fifo_buf->head--;
    }

    fifo_buf->counter--;
    Dprintf("The Roll number is deleted successfully.\n");
    Find_Total_Number(fifo_buf);
}
/*=============================================================================================*/
void Update_Student_Info( FIFO_buf_t* fifo_buf )
{
    int ID;
    int choice;//to choice the task is will update
    int updated=0;//to check is update or no
    Sdata_t* check_Student = fifo_buf->tail;
    Dprintf("\nEnter the Roll Number to Update the Entry: ");
    scanf("%d",&ID);
    for (int i = 0; i < fifo_buf->counter; ++i)
    {
        if (check_Student->ID == ID)
        {
            int RESULT;//to store state of check
            Sdata_t temp_ID;//to store ID before check
            Dprintf("\n1. First Name.");
            Dprintf("\n2. Second Name.");
            Dprintf("\n3. roll no.");
            Dprintf("\n4. GPA.");
            Dprintf("\n5. Course.");
            Dprintf("\nEnter the task to update: ");
            scanf("%d",&choice);
            scanf("%d",&choice);
            switch(choice)
            {
            case 1:
                Dprintf("\nEnter the First Name: ");
                scanf("%s",check_Student->First_name);
                break;
            case 2:
                Dprintf("\nEnter the Last Name: ");
                scanf("%s",check_Student->last_name);
                break;
            case 3:
                Dprintf("\nEnter the Roll Number: ");
                scanf("%d",&temp_ID.ID);
                RESULT=Check_ID(fifo_buf,temp_ID);//to check the update id is not repeated
                if(RESULT==FIFO_no_error)
                    check_Student->ID=temp_ID.ID;
                break;
            case 4:
                Dprintf("\nEnter the GPA: ");
                scanf("%f",&check_Student->GPA);
                break;
            case 5:
                for(int j=0; j<5;j++)
                {
                    Dprintf("\nCourse %d id: ",j+1);
                    scanf("%d",&check_Student->Course[j]);
                }
                break;
            default:
                Dprintf("\n*****************************************\n");
                Dprintf("\n You Entered a Wrong Option \n");
                Dprintf("\n*****************************************\n");
                break;
            }
            updated=1;
        }
```

```
        }
        check_Student++; // move to next element
        if (check_Student == (fifo_buf->base + fifo_buf->length * sizeof(Sdata_t)))
            check_Student = fifo_buf->base; //if fifo is reach to top fifo
    }
    if(updated!=1)
    {
        Dprintf("[Error] Roll Number %d not found \n",ID);
    }
    Find_Total_Number(fifo_buf);
}
/*=============================================================================================*/
void Show_Student( FIFO_buf_t* fifo_buf )
{
    Sdata_t* Show_student = fifo_buf->tail;

    if(FIFO_EMPTY(fifo_buf) == FIFO_empty)
    {
        Dprintf("[ERORR] The FIFO is empty \n");
        return;
    }
    // looping untill show all students
    for(int counter = 0 ; counter < fifo_buf->counter ; counter++)
    {
        Dprintf("\n*****************************************\n");
        Dprintf("Student first name %s\n",Show_student->First_name);
        Dprintf("Student last name %s\n",Show_student->last_name);
        Dprintf("Student roll number %d\n",Show_student->ID);
        Dprintf("Student GPA %0.2f\n",Show_student->GPA);
        Dprintf("The course ID are %d\n",Show_student->Course[0]);
        Dprintf("The course ID are %d\n",Show_student->Course[1]);
        Dprintf("The course ID are %d\n",Show_student->Course[2]);
        Dprintf("The course ID are %d\n",Show_student->Course[3]);
        Dprintf("The course ID are %d\n",Show_student->Course[4]);
        Show_student++;
    }
    Dprintf("\n*****************************************\n");
```

```
FIFO_buf_status FIFO_Enqueue(FIFO_buf_t* fifo , Sdata_t item)
{
    FIFO_buf_status check_student_id;
    //to check the FIFO is not null and check the base , tail , head is existed
    if(!fifo->base || !fifo->head || !fifo->tail)
        return FIFO_null;

    //to check the FIFO IS FULL or no?
    if(FIFO_IS_FULL(fifo)==FIFO_full)
        return FIFO_full;

    // check the id is not repeated

    check_student_id=Check_ID(fifo,item);

    if(check_student_id!=FIFO_error)
    {
        //start of Enqueue process
        *(fifo->head)=item; //assign the value of the item in head
        fifo->counter++;// increment the counter
        //write code of circular FIFO
        if(fifo->head==(fifo->head+(fifo->length*sizeof(Sdata_t))))//if the head is reach to end of FIFO or no
            fifo->head=fifo->base;
        else
            fifo->head++;
        return FIFO_no_error;//return the status of FIFO
    }
    return FIFO_error;
}
/*=============================================================================================*/
FIFO_buf_status FIFO_IS_FULL( FIFO_buf_t *fifo_buf )
{
    //to check the FIFO is not null and check the base , tail , head is existed
    if(!fifo_buf->base || !fifo_buf->head || !fifo_buf->tail)
        return FIFO_null;
    //to check the FIFO is full or no
```

```
435         return FIFO_null;
436     //to check the FIFO is full or no
437     if(fifo_buf->counter>=fifo_buf->length)
438         return FIFO_full;
439     return FIFO_no_error;
440 }
441 /*============================================================================================*/
442 FIFO_buf_status FIFO_EMPTY( FIFO_buf_t* fifo_buf )
443 {
444     //check if FIFO is null
445     if(!fifo_buf->head||!fifo_buf->base||!fifo_buf->tail)
446         return FIFO_null;
447     //to check FIFO is empty
448     if(fifo_buf->counter==0)
449         return FIFO_empty;
450 
451     return FIFO_no_error;
452 }
453 /*============================================================================================*/
454 FIFO_buf_status Check_ID(FIFO_buf_t* fifo_buf , Sdata_t item)
455 {
456     Sdata_t* check_ID = fifo_buf->tail;//to check the id is not repeated
457 
458     for (int i = 0; i < fifo_buf->counter; ++i)
459     {
460         if (check_ID->ID == item.ID)
461         {
462             printf("[ERROR] The ID is Repeated!!\n");
463             return FIFO_error;
464         }
465         check_ID = (check_ID + 1); // move to next element
466         if (check_ID == (fifo_buf->base + fifo_buf->length * sizeof(Sdata_t)))
467             check_ID = fifo_buf->base; //if fifo is reach to top fifo
468     }
469 
470     return FIFO_no_error;
471 }
```