Mastering Embedded system online diploma

https://www.learn-in-depth.com/

Student(s)

Eng: BOULMANE Taha Omar

Profile learn-in-depth: https://www.learn-in-depth.com/online-diploma/boulmane818%40gmail.com

First term final project II

Date of Submission

Monday, August 16, 2021

Lecturer/Tutor

Keroles Shenoda

ABSTRACT

In this report we try to evaluate our competencies on the term of C and data structures for that we design a simple system that can add to a data structure of students both manually or from a text file.

PROBLEM STATEMENT

the system shall

1. Store the first name of the student
2. Store the last name of the student
3. Store the unique Roll number for each student
4. Store the GPA (Grade Point Average) for every student
5. Store the courses registered for the student

APPROACH

1. Add student information from file
2. Add student information manually
3. Find the student by given roll number
4. Find students by a given first name
5. Find student registered in a course
6. Count students.
7. Delete a student
8. Update students

To implement this requirement, we need a Queue of structure called student. For doing so in our header file we define student structure as :

```
typedef struct {      // student information structure
       char fname[50];
       char lname[50];
       int roll;
       float GPA;
       int cid[10];
} Sstudent;
```

Then we defined a buffer (an array) that will contains students' information

```
Sstudent buffer[100];
```

For Queuing we define the queue structure x :

```
typedef struct {
       Sstudent *head;
       Sstudent *tail;
       Sstudent *base;
       int counter;
       int length;
} x;
```

For the queue status we define an enumeration:

```
typedef enum {
	fifo_no_error, fifo_full, fifo_empty, fifo_null, fifo_error

} fifo_buffer_state;
```

The functions which will provide the communication with the system are :

```
// functions
int check_roll(x *fifo, int x); // to check if roll number already exists before or not
fifo_buffer_state fifo_init(x *fifo, Sstudent *buf, int lenght); // initialize the buffer
fifo_buffer_state add_student_file();       // add students info from a text file
fifo_buffer_state add_student_manually(x *fifo);         // add student manually
fifo_buffer_state find_r1(x *fifo);       // find student data using Roll number
fifo_buffer_state find_fn(x *fifo);        // find student data using first name
fifo_buffer_state find_c(x *fifo); // display students info registered by course id
fifo_buffer_state tot_s(x *fifo);                    // total number of student
fifo_buffer_state del_s(x *fifo);                 // delete student by roll num
fifo_buffer_state  shift_buffer(int  index,  x  *fifo);  //  to  shift  buffer  and override on location wanted to be deleted
fifo_buffer_state up_s(x *fifo);                  // update student by roll num
fifo_buffer_state show_s(x *fifo);                // show all information
```

to implement those functions in our c file we prototype them as:

1. Buffer initialization

```
fifo_buffer_state fifo_init(x *fifo, Sstudent *buf, int lenght) {
	if (!fifo || !buf) {
		return fifo_null;
	}
	fifo->base = buf;
	fifo->head = buf;
	fifo->tail = buf;
	fifo->length = lenght;
	fifo->counter = 0;
	return fifo_no_error;
}
```

2. Checking the roll number

```
int check_roll(x *fifo, int x)  // to check if roll number already exists or not
{
	int y;
	Sstudent *ptr = fifo->base;

	for (y = 0; y < (fifo->counter); y++) {
		if (ptr->roll == x) {
			return 0;
		}
		ptr++;
	}
	return 1;
}
```

3. Adding students manually

```
fifo_buffer_state add_student_manually(x *fifo) {
	int temp_int, y, x;
	char temp_str[30];
	if (!fifo->base || !fifo->head || !fifo->tail) // check if queue already
exists or not
				{
		DPRINTF("the database do not exist  \n");
		return fifo_null;
	}
	if (fifo->counter == fifo->length)   // check if full
				{
		DPRINTF("[ERROR] data base is full\n");
		return fifo_full;
	}
	DPRINTF("--------------------------------------------\n");
	DPRINTF("Add Student Details \n");
	DPRINTF("--------------------------------------------\n");
	DPRINTF("Enter the Roll Number\n");
	gets(temp_str);
	temp_int = atoi(temp_str);
	if (check_roll(fifo, temp_int) == 0) {
		DPRINTF("[ERROR] Roll Number is already taken before \n");
		return fifo_error;
	}
	fifo->head->roll = atoi(temp_str);
	DPRINTF("Enter First name of the student:\n");
	gets(fifo->head->fname);
	DPRINTF("Enter Last name of the student:\n");
	gets(fifo->head->lname);
	DPRINTF("Enter the GPA you obtained\n");
	gets(temp_str);
	fifo->head->GPA = atof(temp_str);
	DPRINTF("Enter the course id of each course\n");
	for (x = 0; x < 5; x++) {
		DPRINTF("course %d id :\n", x + 1);
		gets(temp_str);
		y = atoi(temp_str);
		if (y > 0 && y < 30)              // check if course id is available
id
					{
			fifo->head->cid[x] = y;
			continue;
		}
		DPRINTF("[ERROR] course id is not correct\n");
		x--;
	}
	fifo->head++;
	fifo->counter++;
	DPRINTF("[INFO] Student Details are added successfully \n");
	DPRINTF("--------------------------------------------\n");
	DPRINTF("[INFO] the total number of students is : %d\n", fifo->counter);
	DPRINTF("[INFO] you can add up to %d students \n", fifo->length);
	DPRINTF("[INFO] you can add more about %d students \n",
			fifo->length - fifo->counter);
	DPRINTF("--------------------------------------------\n");
	return fifo_no_error;}
```

4. Showing all data

```
fifo_buffer_state show_s(x *fifo)  // show all students information
{
	Sstudent *current_stuednt = fifo->base;
	int x, y;
	if (!fifo->base || !fifo->head || !fifo->tail) // check if the queue exists
already or not
			{
		DPRINTF("database does not exist  \n");
		return fifo_null;
	}
	if (fifo->counter == 0)		// check if is empty
			{
		DPRINTF("--------------------------------------------\n");
		DPRINTF("[ERROR] database is empty \n");

		return fifo_empty;
	}
	for (x = 0; x < fifo->counter; x++)	// show students data
			{
		DPRINTF("--------------------------------------------\n");
		DPRINTF("Student Roll number : %d\n", current_stuednt->roll);
		DPRINTF("Student first name : %s\n", current_stuednt->fname);
		DPRINTF("Student last name : %s\n", current_stuednt->lname);
		DPRINTF("Student GPA : %.2f\n", current_stuednt->GPA);
		for (y = 0; y < 5; y++) {
			DPRINTF("course %d id : %d \n", y + 1, current_stuednt-
>cid[y]);
		}
		current_stuednt++;
	}
	DPRINTF("--------------------------------------------\n");
	DPRINTF("total number of students : %d\n", fifo->counter);
	return fifo_no_error;
}
```

5. Find students by their roll number

```
fifo_buffer_state find_r1(x *fifo)  // find student data using Roll number
{
	char temp_str[30];
	int temp_roll;
	Sstudent *current_stuednt = fifo->base;
	int x, y;
	if (!fifo->base || !fifo->head || !fifo->tail) // check queue is exist or
not
			{
		DPRINTF("database not exist  \n");
		return fifo_null;
	}
	if (fifo->counter == 0)		// check if is empty
			{
		DPRINTF("[ERROR] database is empty \n");
		DPRINTF("--------------------------------------------\n");
		return fifo_empty;
	}
	DPRINTF("Enter student roll number  \n");
	gets(temp_str);
	temp_roll = atoi(temp_str);

	for (x = 0; x < fifo->counter; x++)    // loop to get the roll number
				{
		if (current_stuednt->roll == temp_roll) {
			DPRINTF("--------------------------------------------\n");
			DPRINTF("Student Roll number : %d\n", current_stuednt->roll);
			DPRINTF("Student first name : %s\n", current_stuednt->fname);
			DPRINTF("Student last name : %s\n", current_stuednt->lname);
			DPRINTF("Student GPA : %.2f\n", current_stuednt->GPA);
			for (y = 0; y < 5; y++) {
				DPRINTF("course %d id : %d \n", y + 1, current_stuednt-
>cid[y]);
			}
			return fifo_no_error;
		}
		current_stuednt++;
	}
	DPRINTF("--------------------------------------------\n");
	DPRINTF("[ERROR] Roll number is not found\n"); // loop finished and roll
not found
	DPRINTF("--------------------------------------------\n");
	return fifo_error;
}
```

6. Find student by first name

```
fifo_buffer_state find_fn(x *fifo)         // find student data using first name
{
	char temp_str[30];
	int flag = 0;
	Sstudent *current_stuednt = fifo->base;
	int x, y;
	if (!fifo->base || !fifo->head || !fifo->tail) // check queue is exist or
not
				{
		DPRINTF("database not exist  \n");
		return fifo_null;
	}
	if (fifo->counter == 0)         // check if is empty
				{
		DPRINTF("[ERROR] database is empty \n");
		DPRINTF("--------------------------------------------\n");
		return fifo_empty;
	}
	DPRINTF("Enter student first name   \n");
	gets(temp_str);
	for (x = 0; x < fifo->counter; x++)    // loop to get the roll number
				{
		if (strcmpi(current_stuednt->fname, temp_str) == 0) //compare
strings without case sensitive
					{
			DPRINTF("--------------------------------------------\n");
			DPRINTF("Student Roll number : %d\n", current_stuednt->roll);
			DPRINTF("Student first name : %s\n", current_stuednt->fname);
			DPRINTF("Student last name : %s\n", current_stuednt->lname);
			DPRINTF("Student GPA : %.2f\n", current_stuednt->GPA);
```

```
                    for (y = 0; y < 5; y++) {
                          DPRINTF("course %d id : %d \n", y + 1, current_stuednt-
>cid[y]);
                    }
                    DPRINTF("--------------------------------------------\n");
                    flag = 1;       // flag to know the first name found at least
1 time
              }
              current_stuednt++;
       }
       if (flag == 0) {
              DPRINTF("--------------------------------------------\n");
              DPRINTF("[ERROR] No first name matched this name\n"); // loop
finished and roll not found
              DPRINTF("--------------------------------------------\n");
              return fifo_error;
       }
       return fifo_no_error;
}
```

7. Display students by course they are registered for

```
fifo_buffer_state find_c(x *fifo) // display students info registered by course id
{
       char temp_str[30];
       int temp_course_id;
       Sstudent *current_stuednt = fifo->base;
       int x, y, z, flag = 0;
       if (!fifo->base || !fifo->head || !fifo->tail) // check queue is exist or
not
                    {
              DPRINTF("database not exist  \n");
              return fifo_null;
       }
       if (fifo->counter == 0)         // check if is empty
                    {
              DPRINTF("[ERROR] database is empty \n");
              DPRINTF("--------------------------------------------\n");
              return fifo_empty;
       }
       DPRINTF("Enter course id number  \n");
       gets(temp_str);
       temp_course_id = atoi(temp_str);
       for (x = 0; x < fifo->counter; x++)   // loop to get the course id

                    {
              for (z = 0; z < 5; z++) {
                    if (current_stuednt->cid[z] == temp_course_id) // if course id
matches
                                 {
                          DPRINTF("--------------------------------------------
\n");

                          DPRINTF("Student Roll number : %d\n", current_stuednt-
>roll);

                          DPRINTF("Student first name : %s\n", current_stuednt-
>fname);

                          DPRINTF("Student last name : %s\n", current_stuednt-
>lname);
```

```
                    DPRINTF("Student GPA : %.2f\n", current_stuednt->GPA);
                    for (y = 0; y < 5; y++) {
                        DPRINTF("course %d id : %d \n", y + 1,
                                current_stuednt->cid[y]);
                    }
                    flag = 1; // found at least one student
                }
            }
            current_stuednt++;
        }
        if (flag == 0) {
            DPRINTF("--------------------------------------------\n");
            DPRINTF("[ERROR] no student registered \n"); // loop finished and
students not found
            DPRINTF("--------------------------------------------\n");
            return fifo_error;
        }
        return fifo_no_error;

}
```

8. Counting the total number of students

```
fifo_buffer_state tot_s(x *fifo)                    // total number of student
{

        if (!fifo->base || !fifo->head || !fifo->tail) // check queue is exist or
not
                {
            DPRINTF("database not exist  \n");
            return fifo_null;
        }
        if (fifo->counter == 0)          // check if is empty
                {
            DPRINTF("[ERROR] database is empty \n");
            DPRINTF("--------------------------------------------\n");
            return fifo_empty;
        }
        DPRINTF("--------------------------------------------\n");
        DPRINTF("[INFO] the total number of students is : %d\n", fifo->counter);
        DPRINTF("[INFO] you can add up to %d students \n", fifo->length);
        DPRINTF("[INFO] you can add more about %d students \n",
                fifo->length - fifo->counter);
        DPRINTF("--------------------------------------------\n");
        return fifo_no_error;
}
```

9. Deleting students

```
fifo_buffer_state del_s(x *fifo)                  // delete student by roll num
{
        char temp_str[30];
        int x, y, temp_roll, index = 0;
        Sstudent *current_stuednt = fifo->base;
        if (!fifo->base || !fifo->head || !fifo->tail) // check queue is exist or
not
                {
            DPRINTF("database not exist  \n");
            return fifo_null;
        }
        if (fifo->counter == 0)          // check if is empty
                {
            DPRINTF("[ERROR] database is empty \n");
            DPRINTF("--------------------------------------------\n");
            return fifo_empty;
        }
        DPRINTF("Enter student roll number  \n");
        gets(temp_str);
        temp_roll = atoi(temp_str);

        for (x = 0; x < fifo->counter; x++)    // loop to get the roll number
                {
            if (current_stuednt->roll == temp_roll) {
                DPRINTF("--------------------------------------------\n");
                DPRINTF("Student Roll number : %d\n", current_stuednt->roll);
                DPRINTF("Student first name : %s\n", current_stuednt->fname);
                DPRINTF("Student last name : %s\n", current_stuednt->lname);
                DPRINTF("Student GPA : %.2f\n", current_stuednt->GPA);
                for (y = 0; y < 5; y++) {
                    DPRINTF("course %d id : %d \n", y + 1, current_stuednt-
>cid[y]);
                }
                DPRINTF("--------------------------------------------\n");
                DPRINTF("Delete student 1-yes 2-No \n");
                DPRINTF("--------------------------------------------\n");
                gets(temp_str);
                temp_roll = atoi(temp_str);
                if (temp_roll == 1) {
                    shift_buffer(index, fifo); // to shift buffer and
override on location wanted to be deleted
                    fifo->head--;
                    fifo->counter--;
                    return fifo_no_error;
                } else if (temp_roll == 0) {
                    DPRINTF("--------------------------------------------
\n");
                    DPRINTF("--------Process canceled-------\n");
                    return fifo_no_error;
                } else {
                    DPRINTF("--------------------------------------------
\n");
                    DPRINTF(
                            "[ERROR]wrong choice ..\n Uncompleted
process...\n back to main menu .....  \n");
                    return fifo_no_error;
                }
            }
            current_stuednt++;
            index++;               //to find location in the buffer
        }
        DPRINTF("--------------------------------------------\n");
        DPRINTF("[ERROR] Roll number is not found\n"); // loop finished and roll
not found
        DPRINTF("--------------------------------------------\n");
        return fifo_error;
}
```

10. Shifting buffer

```
fifo_buffer_state shift_buffer(int index, x *fifo) // to shift buffer and override
the location to be deleted
{
	int x;
	for (x = index; x < fifo->counter; x++) {
		buffer[x] = buffer[x + 1];
	}
	DPRINTF("--------------------------------------------\n");
	DPRINTF("student deleted successfully\n");
	DPRINTF("--------------------------------------------\n");
	return fifo_no_error;
}
```

11. Updating students by roll number

```
fifo_buffer_state up_s(x *fifo)                   // update student by roll num
{
	char temp_str[30];
	int x, y, i, j, temp_option, temp_roll;
	Sstudent *current_stuednt = fifo->base;
	if (!fifo->base || !fifo->head || !fifo->tail) // check queue is exist or
not
			{
		DPRINTF("database not exist  \n");
		return fifo_null;
	}
	if (fifo->counter == 0)          // check if is empty
			{
		DPRINTF("[ERROR] database is empty \n");
		DPRINTF("--------------------------------------------\n");
		return fifo_empty;
	}
	DPRINTF("Enter student roll number  \n");
	gets(temp_str);
	temp_roll = atoi(temp_str);

	for (x = 0; x < fifo->counter; x++)   // loop to get the roll number
			{
		if (current_stuednt->roll == temp_roll) {
			DPRINTF("--------------------------------------------\n");
			DPRINTF("Student Roll number : %d\n", current_stuednt->roll);
			DPRINTF("Student first name : %s\n", current_stuednt->fname);
			DPRINTF("Student last name : %s\n", current_stuednt->lname);
			DPRINTF("Student GPA : %.2f\n", current_stuednt->GPA);
			for (y = 0; y < 5; y++) {
				DPRINTF("course %d id : %d \n", y + 1, current_stuednt-
>cid[y]);
			}
			DPRINTF("--------------------------------------------\n");
			DPRINTF("Enter option to update data\n");
			DPRINTF("1- first name\n");
			DPRINTF("2- last name \n");
			DPRINTF("3- GPA\n");
			DPRINTF("4- courses \n");
			DPRINTF("--------------------------------------------\n");
			gets(temp_str);
```

```
                temp_option = atoi(temp_str);
                switch (temp_option) {
                case 1: {
                        DPRINTF("Enter New first name\n");
                        gets(current_stuednt->fname);
                        break;
                }
                case 2: {
                        DPRINTF("Enter New second name\n");
                        gets(current_stuednt->lname);
                        break;

                }
                case 3: {
                        DPRINTF("Enter New GPA \n");
                        gets(temp_str);
                        current_stuednt->GPA = atof(temp_str);
                        break;

                }
                case 4: {
                        DPRINTF("Enter the course id of each course\n");
                        for (i = 0; i < 5; i++) {
                                DPRINTF("course %d id :\n", i + 1);
                                gets(temp_str);
                                j = atoi(temp_str);
                                if (j > 0 && j < 30)    // check if course id is
available
                                                {
                                        current_stuednt->cid[i] = j;
                                        continue;
                                }
                                DPRINTF("[ERROR] course id is not correct\n");
                                i--;
                        }
                        break;

                }
                default: {
                        DPRINTF("[ERROR] wrong choice \n");
                        return fifo_error;
                }
                }
                // print student information after update
                DPRINTF("-------------------------------------------\n");
                DPRINTF("information updated successfully \n");
                DPRINTF("Student information after update \n");
                DPRINTF("-------------------------------------------\n");
                DPRINTF("Student Roll number : %d\n", current_stuednt->roll);
                DPRINTF("Student first name : %s\n", current_stuednt->fname);
                DPRINTF("Student last name : %s\n", current_stuednt->lname);
                DPRINTF("Student GPA : %.2f\n", current_stuednt->GPA);
                for (y = 0; y < 5; y++) {
                        DPRINTF("course %d id : %d \n", y + 1, current_stuednt-
>cid[y]);
                }
                return fifo_no_error;
        }
```

```
                current_stuednt++;
        }
        DPRINTF("--------------------------------------------\n");
        DPRINTF("[ERROR] Roll number is not found\n"); // loop finished and roll
not found
        DPRINTF("--------------------------------------------\n");
        return fifo_error;
}
```

12. Adding student from a text file

```
fifo_buffer_state add_student_file(x *fifo) // add students info using text file
{
        char f_name[50];
        char l_name[50];
        int roll_num, cid[5], x, file_count = 0, flag = 0;
        float GPA;
        int line = 0;
        if (!fifo->base || !fifo->head || !fifo->tail) // check queue is exist or
not
                        {
                DPRINTF("database not exist  \n");
                return fifo_null;
        }
        if (fifo->counter == fifo->length)   // check if full
                        {
                DPRINTF("[ERROR] data base is full\n");
                return fifo_full;
        }
        FILE *p_file = fopen("text.txt", "r");
        if (p_file == NULL) {
                DPRINTF("------------------------------ \n");
                DPRINTF("[ERROR] File not found \n");
                return fifo_error;
        }

        // reading from file

        while (fscanf(p_file, "%d %s %s %f %d %d %d %d %d [^\n]", &roll_num,
f_name,
                        l_name, &GPA, &cid[0], &cid[1], &cid[2], &cid[3], &cid[4]) !=
EOF) {
                if (fifo->counter == fifo->length) {
                        DPRINTF("----------------------------\n");
                        DPRINTF("[ERROR] data base is full\n");
                        DPRINTF("[INFO] students added : %d\n", file_count);
                        DPRINTF("[INFO] remaining students due to size or errors are
:%d\n",
                                        line - file_count);

                        return fifo_full;
                }
                if (check_roll(fifo, roll_num) == 0) {
                        DPRINTF(
                                        "[ERROR] IN line %d : Roll Number is already
taken before \n",
                                        line);
                        line++;
```

```
				continue; // to skip this student
			}
			fifo->head->roll = roll_num;
			fifo->head->GPA = GPA;
			strcpy(fifo->head->fname, f_name);
			strcpy(fifo->head->lname, l_name);
			for (x = 0; x < 5; x++) {
				flag = 0;
				if (cid[x] < 0 || cid[x] > 30) {
					flag = 1;	// that there is non-valid course id
					break;
				}
				fifo->head->cid[x] = cid[x];
			}
			if (flag == 1)  // non valid course id
					{
				DPRINTF(
						"[ERROR] IN line %d : non valid course id we will
skip this student \n",
						line);
				line++;
				continue; // to skip this student
			}
			fifo->head++;
			fifo->counter++;
			line++;
			file_count++;	// to record successful records

	}
	DPRINTF("\nEnd of file.\n");

	// close connection
	fclose(p_file);
	DPRINTF("[INFO] students added : %d\n", file_count);
	DPRINTF("[INFO] remaining students due to errors are :%d\n",
				line - file_count);
	return fifo_error;
}
```

After that a main code for testing is done as below:

```
int main(void) {
	int temp;
	x buffer_controller;  // that controls student buffer
	fifo_init(&buffer_controller, buffer, 100);

	DPRINTF("Welcome to the Student Management System\n");
	while (1) {
		DPRINTF("--------------------------------------------\n");
		DPRINTF("Choose The Task that you want to perform\n");
		DPRINTF("1. Add the Student Details Manually\n");
		DPRINTF("2. Add the Student Details From Text File\n");
		DPRINTF("3. Find the Student Details by Roll Number\n");
		DPRINTF("4. Find the Student Details by First Name\n");
		DPRINTF("5. Find the Student Details by Course ID\n");
		DPRINTF("6. Find the Total number of Students\n");
		DPRINTF("7. Delete the Student Details by Roll Number \n");
		DPRINTF("8. Update the Student Details by Roll Number \n");
```

```
            DPRINTF("9. Show all information\n");
            DPRINTF(" Enter your choice to perform the task\n");
            scanf("%d", &temp);
            switch (temp) {
            case 1: {
                   add_student_manually(&buffer_controller);
                   break;
            }
            case 2: {
                   add_student_file(&buffer_controller);
                   break;
            }
            case 3: {
                   find_r1(&buffer_controller);
                   break;
            }
            case 4: {
                   find_fn(&buffer_controller);
                   break;
            }
            case 5: {
                   find_c(&buffer_controller);
                   break;
            }
            case 6: {
                   tot_s(&buffer_controller);
                   break;
            }
            case 7: {
                   del_s(&buffer_controller);
                   break;
            }
            case 8: {
                   up_s(&buffer_controller);
                   break;
            }
            case 9: {
                   show_s(&buffer_controller);
                   break;
            }
            default: {
                   DPRINTF("Wrong choice\n");

            }
            }

      }

}
```

RESULTS

```
Welcome to the Student Management System
-------------------------------------------
Choose The Task that you want to perform
1. Add the Student Details Manually
2. Add the Student Details From Text File
3. Find the Student Details by Roll Number
```

```
4. Find the Student Details by First Name
5. Find the Student Details by Course ID
6. Find the Total number of Students
7. Delete the Student Details by Roll Number
8. Update the Student Details by Roll Number
9. Show all information
 Enter your choice to perform the task
1
--------------------------------------------
Add Student Details
--------------------------------------------
Enter the Roll Number
1245
Enter First name of the student:
taha
Enter Last name of the student:
taha
Enter the GPA you obtained
15
Enter the course id of each course
course 1 id :
01
course 2 id :
02
course 3 id :
03
course 4 id :
04
course 5 id :
05
[INFO] Student Details are added successfully
--------------------------------------------
[INFO] the total number of students is : 1
[INFO] you can add up to 100 students
[INFO] you can add more about 99 students
--------------------------------------------
--------------------------------------------
Choose The Task that you want to perform
1. Add the Student Details Manually
2. Add the Student Details From Text File
3. Find the Student Details by Roll Number
4. Find the Student Details by First Name
5. Find the Student Details by Course ID
6. Find the Total number of Students
7. Delete the Student Details by Roll Number
8. Update the Student Details by Roll Number
9. Show all information
 Enter your choice to perform the task
9
--------------------------------------------
Student Roll number : 1245
Student first name : taha
Student last name : taha
Student GPA : 15.00
course 1 id : 1
course 2 id : 2
course 3 id : 3
course 4 id : 4
course 5 id : 5
--------------------------------------------
total number of students : 1
--------------------------------------------
Choose The Task that you want to perform
1. Add the Student Details Manually
2. Add the Student Details From Text File
3. Find the Student Details by Roll Number
4. Find the Student Details by First Name
5. Find the Student Details by Course ID
6. Find the Total number of Students
7. Delete the Student Details by Roll Number
8. Update the Student Details by Roll Number
9. Show all information
 Enter your choice to perform the task
```