Master Embedded System

Learn-in-depth.

Student Management System Project 2

By: Ibrahim Abo Elhassan

Agenda

- Problem Statement
- Approach
- Main.c
- Std_mangement.h
- Std_mangement.c

    1. System_init
    2. Add_Data_From_File
    3. Add_Students_Manually
    4. Search_By_first_name
    5. Search_about_course_id
    6. Search_By_ID
    7. Total_number_of_students
    8. Delete_student_By_id.
    9. Update_student_info
    10. Top_students
    11. Show_all_number

- Run program.

Problem Statement

A simple software for student information management system which can perform the following operations:

1. Store first name of the student.
2. Store last name of the student.
3. Store unique id number for every student.
4. Store GPA for every student.
5. Store courses registered by the student.

Approach

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

1. Add student info from file.
2. Add student info manually.
3. Find the student by the given id number.
4. Find the student by the given first name.
5. Find the student registered in a course.
6. Total number of students.
7. Delete a student by the given id number.
8. Update a student by the given id number.
9. Show all student’s data.
10. Exit the program.

Main.c

```c
#include "std_mangement.h"

    FILE*File_data = NULL;
    char read_date[81];

int main()
{
    Item student_list[MAX_STUDENTS];
    Fifo_buf students;
    Fifo_status init_status  = System_init(&students, student_list, MAX_STUDENTS);

    int result;
    char read_date[81];


    if (init_status == FIFO_No_error) {
        int choice;

        while (1) {
            printf("  == Welcome in our system management ==\n");
            printf("\t\t## Our Menu ## \n");
            printf("\t1.  Add Student from file\n");
            printf("\t2.  Add Student Manually\n");
            printf("\t3.  Find student info from his first name\n");
            printf("\t4.  Find student info from his id\n");
            printf("\t5.  Find student registered in specific course id\n");
            printf("\t6.  Delete student by his id\n");
            printf("\t7.  Update student info \n");
            printf("\t8.  Show all number\n");
            printf("\t9.  Top students \n");
            printf("\t10. Total number of students\n");
            printf("\t11. Exit\n");
            printf("\n");
            printf("Enter your choice: ");
            scanf("%d", &choice);
            printf("-----------------------\n");
            printf("\n");

            switch (choice) {
                case 1:
                    if (students.counter < students.length) {
                        FILE* File_data = fopen("info_data.txt", "r");
                        if (File_data) {
                            char read_date[81];
                            while (fgets(read_date, sizeof(read_date), File_data) != NULL) {
                                read_date[sizeof(read_date) - 1] = '\0';
                                Add_Data_From_File(&students, read_date);
                            }

                            int result = fclose(File_data);
                            if (result == 0) {
                            printf(" Data added from file successfully \n");
                            Total_number_of_students(&students);
                            printf("\n");
                            } else {
                            printf("Error closing the file.\n");
                            }
                        } else {
                        printf("File not opened successfully.\n");
                        }
                    } else {
                        printf("Cannot add more students, the list is full.\n");
                    }
                    break;
                case 2:
                    Add_Students_Manually(&students);
                    Total_number_of_students(&students);
                    break;
                case 3:
                    Search_By_first_name(&students);
                    break;
                case 4:
                    Search_By_ID(&students);
                    break;
                case 5:
                    Search_about_course_id(&students);
                    break;
                case 6:
                    Delete_student_By_id(&students);
                    break;
```

```
                case 6:
                    Delete_student_By_id(&students);
                    break;
                case 7:
                    Update_student_info(&students);
                    break;
                case 8:
                    Show_all_number(&students);
                    break;
                case 9:
                    Top_students(&students);
                    break;
                case 10:
                    Total_number_of_students(&students);
                    break;

                case 11:
                    char cch;
                    printf("Are you sure you want to exit our system? (y/n): ");
                    getchar();
                    scanf("%c", &cch);
                    if (cch == 'y' || cch == 'Y') {
                        printf("== Goodbye ==\n");
                        return 0;
                    } else {
                        break;
                    }

                default:
                    printf("Wrong choice. Please choose again.\n");
                    printf("\n");
            }
        }
    } else {
        printf("Initialization failed.\n");
    }

    return 0;
}
```

Std_mangement.h

```h
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_STUDENTS 50
#define NAME_LENGTH 50
#define COURSES_NUMBERS 5

typedef struct Sinfo {
    char student_first_name[NAME_LENGTH];
    char student_last_name[NAME_LENGTH];
    int student_ID;
    float student_GPA;
    int student_courses_ids[COURSES_NUMBERS];
}Item;

typedef struct {
    Item* base;
    Item* tail;
    Item* head;
    int length;
    int counter;
} Fifo_buf;

typedef enum {
    FIFO_No_error,
    FIFO_Empty,
    FIFO_Full,
    FIFO_Null,
    FIFO_NOT_Full,
    FIFO_NOT_Empty
}Fifo_status;

Fifo_status List_is_full(Fifo_buf* student);
Fifo_status List_is_empty(Fifo_buf* student);
void Add_Data_From_File(Fifo_buf* student, char* read_date);
Fifo_status System_init(Fifo_buf* student, Item* list, int length);
Fifo_status Add_Students_Manually(Fifo_buf* student);
void Search_By_first_name(Fifo_buf* student);
void Search_about_course_id(Fifo_buf* student);
void Search_By_ID(Fifo_buf* student);
void Total_number_of_students(Fifo_buf* student);
void Delete_student_By_id(Fifo_buf* student);
void Update_student_info(Fifo_buf* student);
void Top_students(Fifo_buf* student);
void Show_all_number(Fifo_buf* student);

#endif // STD_MANAGEMENT_H
```

Std_mangement.c

1- System_init

```c
#include "std_mangement.h"

Fifo_status List_is_full(Fifo_buf* student)
{
    if(!student || !student->base || !student->head || ! student->tail)
    {
        printf("List is NULL \n");
        return FIFO_Null;
    }


    if(student->counter == student->length)
    {
        return FIFO_Full;
    }

    return FIFO_NOT_Full;
}

Fifo_status System_init(Fifo_buf* student, Item* list, int length)
{
    if (!student || !list || length <= 0) {
        printf("Failed init \n");
        return FIFO_Null;
    }

    student->base = list;
    student->head = student->base;
    student->tail = student->base;
    student->length = length;
    student->counter = 0;

    return FIFO_No_error;
}
```

2- Add data from file

```c
void Add_Data_From_File(Fifo_buf* student, char* read_date)
{

    if (List_is_full(student) == FIFO_Full) {
        printf("[ERROR] Cannot add new students The list is full.\n");
    }

    Item new_std;
    sscanf(read_date, "%d %49s %49s %f %d %d %d %d %d",
           &new_std.student_ID, new_std.student_first_name, new_std.student_last_name, &new_std.student_GPA,
           &new_std.student_courses_ids[0], &new_std.student_courses_ids[1], &new_std.student_courses_ids[2],
           &new_std.student_courses_ids[3], &new_std.student_courses_ids[4]);

    for (int i = 0; i < student->counter; i++) {
        if (new_std.student_ID == student->base[i].student_ID) {
            return;
        }
    }

    if (student->counter < student->length) {
        student->base[student->counter] = new_std;
        student->counter++;
    }
}
```

3- Add data Manually.

```
Fifo_status Add_Students_Manually(Fifo_buf* student)
{
    Item* std = student->base;
    Item new_std;

    if (List_is_full(student) == FIFO_Full)
        {
            printf("[ERROR] No need enough students The list is full.\n");
            return FIFO_No_error;
        }

    printf("Enter your ID: ");
    scanf("%d", &new_std.student_ID);

    for (int i = 0; i < student->counter; i++) {
        if (new_std.student_ID == std->student_ID)
        {
            printf("This id is already taken.\n");
            return FIFO_No_error;
        }
        std++;
    }

    printf("Enter your First Name: ");
    scanf("%s", new_std.student_first_name);
    printf("Enter your Last Name: ");
    scanf("%s", new_std.student_last_name);

    printf("Enter your course id:\n");
    for (int i = 0; i < 5; i++) {
        int temp;
        int already_registered = 0;

          printf("Course %d : ",i+1);
          scanf("%d", &temp);

          for (int j = 0; j < i; j++) {
              if (new_std.student_courses_ids[j] == temp) {
                  printf("You are already registered in this course.\n");
                  already_registered = 1;
                  i--;
                  break;
              }
          }

          if (!already_registered) {
              new_std.student_courses_ids[i] = temp;
          }
    }

    float test = 0.0;
    do {
        printf("Enter your GPA (out of 4): ");
        scanf("%f", &test);
        if (test > 4.00 || test < 0.00) {
            printf("Please, try again !!\n");
        }
    } while (test > 4.00 || test < 0.00);

    printf("\n");
    printf("=== Thanks for your registration ===\n");

    new_std.student_GPA = test;

    student->base[student->counter] = new_std;
    student->counter++;

    return FIFO_No_error;
}
```

4- Search_By_first_name

```c
void Search_By_first_name(Fifo_buf* student)
{
    char name[50];
    int nameFound = 0;

    if (List_is_empty(student) == FIFO_Empty) {
        printf("[ERROR] The student list is empty and your request can't achieve \n");
        printf("\n");
        return;
    }
    if (List_is_full(student) == FIFO_Full) {
        printf("[ERROR] The student list is full and your request can't achieve \n");
        printf("\n");
        return;
    }

    printf("Enter a first name: ");
    scanf("%s", name);

    for (int i = 0; i < student->counter; i++) {
        if (strcmp(name, student->base[i].student_first_name) == 0) {
            printf("Detailed info related to name [%s]:\n", name);
            printf("\tID: %d\n", student->base[i].student_ID);
            printf("\tFirst Name: %s\n", student->base[i].student_first_name);
            printf("\tLast Name: %s\n", student->base[i].student_last_name);
            printf("\tGPA: %.2f\n", student->base[i].student_GPA);
            printf("\tCourses IDS: \n");
            for (int j = 0; j < 5; j++) {
                printf("\tCourse %d : %d\n", j+1 , student->base[i].student_courses_ids[j]);
            }
            printf("================================\n");
            printf("\n");

            nameFound = 1; // Set the flag since the name is found in at least one student's record
        }
    }

    if (!nameFound) {
        printf("This name [%s] is not found \n", name);
        printf("\n");
    }
}
```

5- Search_about_course_id

```c
void Search_about_course_id(Fifo_buf* student)
{
    int c_id;

    if (List_is_empty(student) == FIFO_Empty) {
        printf("[ERROR] The student list is empty and your request can't achieve \n");
        printf("\n");
        return;
    }

    printf("Enter course id: ");
    scanf("%d", &c_id);

    int courseFound = 0;
    int studentsInCourse = 0;

    printf("Students registered in this course: \n");
    for (int i = 0; i < student->counter; i++) {
        Item* Current_student = &student->base[i];

        for (int j = 0; j < 5; j++) {
            if (Current_student->student_courses_ids[j] == c_id) {
                printf("\tFirst Name: %s\n", Current_student->student_first_name);
                printf("\tLast Name: %s\n", Current_student->student_last_name);
                printf("\tID: %d\n", Current_student->student_ID);
                printf("\tGPA: %0.2f\n", Current_student->student_GPA);

                courseFound = 1;
                studentsInCourse++;
                break;
            }
        }
    }

    if (!courseFound) {
        printf("This course id [%d] is NOT Found \n", c_id);
    }
    else{
        printf("Total number in this course %d\n", studentsInCourse); // Display the correct count
    }

    printf("================================\n");
    printf("\n");
}
```

6- Search_By_ID

```
253  void Search_By_ID(Fifo_buf* student)
254  {
255      int id;
256
257      if (List_is_empty(student) == FIFO_Empty)
258          {
259              printf("[NOTE] The student list is empty and your request can't achieve.\n");
260              printf("\n");
261              return;
262          }
263
264      printf("Enter an id: ");
265      scanf("%d", &id);
266
267      if (List_is_empty(student) == FIFO_Empty)
268          {
269              printf("[NOTE] The student list is empty and your request can't achieve.\n");
270              printf("\n");
271              return;
272          }
273
274      for (int i = 0; i < student->counter; i++)
275      {
276          if (student->base[i].student_ID== id)
277          {
278              printf("Detailed info related to id %d:\n", id);
279              printf("ID: %d\n", student->base[i].student_ID);
280              printf("First Name: %s\n", student->base[i].student_first_name);
281              printf("Last Name: %s\n", student->base[i].student_last_name);
282              printf("GPA: %.2f\n", student->base[i].student_GPA);
283              for(int j=0 ; j<5 ;j++)
284              {
285                  printf("course : %d\n",student->base[i].student_courses_ids[j]);
286              }
287              printf("================================\n");
288              printf("\n");
289              return;
290          }
291      }
292
293      printf(" This id [%d] is not found.\n", id);
294      printf("\n");
295  }
296
```

7- Total_number_of_students

```
241
242   void Total_number_of_students(Fifo_buf* student)
243   {
244       int count = student->counter;
245       int size = student->length;
246
247       printf("[NOTE] Total students: %d\n", count);
248       printf("[NOTE] You can add %d more students.\n", size - count);
249       printf("====================================\n");
250       printf("\n");
251   }
```

8- Delete_student_By_id.

```
297
298   void Delete_student_By_id(Fifo_buf* student)
299   {
300       int id_delete;
301
302       if (List_is_empty(student) == FIFO_Empty)
303           {
304               printf("[NOTE] The student list is empty and your request can't achieve.\n");
305               printf("\n");
306               return;
307           }
308
309       printf("Enter an id to delete: ");
310       scanf("%d", &id_delete);
311
312       for (int i = 0; i < student->counter; i++) {
313           if (student->base[i].student_ID == id_delete) {
314               // Move the last student to the deleted student's position
315               student->base[i] = student->base[student->counter - 1];
316               student->counter--;
317               printf("Student with ID %d deleted \n", id_delete);
318               printf("============================\n");
319               printf("\n");
320               return;
321           }
322       }
323
324       printf("This ID [%d]  is Not FOUND to delete it \n", id_delete);
325       printf("\n");
326   }
```

9- Update_student_info

```c
void Update_student_info(Fifo_buf* student)
{
    int id, ch, temp;
    Item *new_std = student->base;

    if (List_is_empty(student) == FIFO_Empty)
        {
            printf("[NOTE] The student list is empty and your request can't achieve.\n");
            printf("\n");
            return;
        }

    printf("Enter an id: ");
    scanf("%d", &id);

    for (int i = 0; i < student->counter; i++)
        {
        if (student->base[i].student_ID == id) {
            Item* up_std = &student->base[i];  // Pointer to the student for updating
            printf("1. First name\n");
            printf("2. Last name\n");
            printf("3. ID\n");
            printf("4. GPA\n");
            printf("5. courses id\n");
            printf("Please choose which data to update: ");
            scanf("%d", &ch);

            switch (ch) {
                case 1:
                    printf("Enter a first name: ");
                    scanf("%s", up_std->student_first_name);
                    printf("\tUpdate First Name [%s] is DONE Successfully \n",up_std->student_first_name);
                    printf("\t============================================================\n");
                    break;
                case 2:
                    printf("Enter a last name: ");
                    scanf("%s", up_std->student_last_name);
                    printf("\tUpdate Last Name [%s] is Done Successfully\n",up_std->student_last_name);
                    printf("\t============================================================ \n");
                    break;
                case 3:
                    int new_id;
                    int Is_unique;  //use it as a flag

                    do {
                        Is_unique = 1;
                        printf("Enter a new id: ");
                        scanf("%d", &new_id);

                        // Check if the new ID already exists in the list
                        for (int i = 0; i < student->counter; i++) {
                            if (i != up_std - student->base && student->base[i].student_ID == new_id) {
                                printf("This id is already taken. Please enter a different ID.\n");
                                Is_unique = 0; // Set the flag to indicate the ID is not unique
                    break;} }

                    } while (!Is_unique);

                    up_std->student_ID = new_id;

                    printf("\t Update New Id [%d] is Done Successfully\n", up_std->student_ID);
                    printf("\t============================================================ \n");
                    break;
                case 4:
                    printf("Enter a new GPA: ");
                    scanf("%f", &up_std->student_GPA);
                    printf("\tUpdate New Id [%f] is Done Successfully\n",up_std->student_GPA);
                    printf("\t============================================================ \n");
                    break;
```

```
398                 case 5:
399                     printf("Enter the course ID number to update (1-5): ");
400                     int NO_c;
401                     scanf("%d", &NO_c);
402 
403                     if (NO_c < 1 || NO_c > 5) {
404                         printf("[ERROR] Try Again !! \n");
405                         break;
406                     }
407 
408                     int new_c_id;
409                     printf("Enter the new course id: ");
410                     scanf("%d", &new_c_id);
411 
412                     // Check if the new course ID is already registered
413                     int already_registered = 0;
414                     for (int i = 0; i < 5; i++) {
415                         if (i != NO_c - 1 && up_std->student_courses_ids[i] == new_c_id) {
416                         printf("[NOTE] You are already registered in \n");
417                         already_registered = 1;
418                         break;}
419                     }
420 
421                     if (!already_registered) {
422                         up_std->student_courses_ids[NO_c - 1] = new_c_id;
423                         printf("\tUpdate New Course id [%d] is DONE Successfully\n",new_c_id);
424                         printf("\t========================================================= \n");
425                     }
426 
427 
428                     break;
429 
430                 default:
431                     printf("Wrong choice. update failed.\n");
432             }
433             return;
434         }
435     }
436 
437     printf("Student with id number [%d] not found.\n", id);
438     printf("========================================\n");
439     printf("\n");
440 }
```

10- Top_students

```
469 void Top_students(Fifo_buf* student)
470 {
471     int test=0;
472     Item* top_std = student->base;
473 
474     if (List_is_empty(student) == FIFO_Empty)
475         {
476             printf("[NOTE] The student list is empty and your request can't achieve.\n");
477             printf("\n");
478             return;
479         }
480 
481     printf("=== Our Honor Board ===\n");
482     for (int i = 0; i < student->counter; i++)
483     {
484         if(top_std->student_GPA >= 3.5)
485         {
486 
487             printf("**First name %s \n",top_std->student_first_name);
488             printf("  Last name %s \n",top_std->student_last_name);
489             printf("  ID %d \n",top_std->student_ID);
490             printf("  GPA %0.2f \n",top_std->student_GPA);
491 
492             test = 1;
493         }
494         top_std++;
495     }
496     printf("========================================\n");
497     printf("\n");
498     if(!test)
499     {
500         printf("Our students need more effort.\n");
501         printf("\n");
502     }
503 
504 }
```

11- Show_all_number

```
441
442  void Show_all_number(Fifo_buf* student)
443  {
444      int count = 1;
445      Item* travers = student->base;
446
447      if(student->counter == 0)
448      {
449          printf("\tNo one registered yet .. The students list is empty \n");
450      }
451      for (int i = 0; i < student->counter; i++) {
452          printf("==== data of student #%d ====\n",i+1);
453          printf("first name : %s\n", travers->student_first_name);
454          printf("last name : %s\n", travers->student_last_name);
455          printf("id : %d\n", travers->student_ID);
456          printf("gpa: %.2f\n", travers->student_GPA);
457          printf("His courses ids that registered in: \n");
458          for (int j = 0; j < 5; j++) {
459              printf("course No.%d : %d\n", j+1 , travers->student_courses_ids[j]);
460          }
461          printf("_____________________________________________\n");
462          travers++;
463      }
464      printf("\n");
465  }
```

- Extra-functions “ FULL, EMPTY”

```
3    Fifo_status List_is_full(Fifo_buf* student)
4    {
5        if(!student || !student->base || !student->head || ! student->tail)
6        {
7            printf("List is NULL \n");
8            return FIFO_Null;
9        }
10
11
12       if(student->counter == student->length)
13       {
14           return FIFO_Full;
15       }
16
17       return FIFO_NOT_Full;
18   }
19
20
21   Fifo_status List_is_empty(Fifo_buf* student)
22   {
23       if(!student || !student->base || !student->head || ! student->tail)
24       {
25           printf("List is NULL \n");
26           return FIFO_Null;
27       }
28
29       if(student->counter == 0)
30       {
31           return FIFO_Empty;
32       }
33
34       return FIFO_NOT_Empty;
35   }
```

- Run Program

```
  == Welcome in our system management ==
                ## Our Menu ##
        1.  Add Student from file
        2.  Add Student Manually
        3.  Find student info from his first name
        4.  Find student info from his id
        5.  Find student registered in specific course id
        6.  Delete student by his id
        7.  Update student info
        8.  Show all number
        9.  Top students
        10. Total number of students
        11. Exit

Enter your choice: 1
----------------------

 Data added from file successfully
[NOTE] Total students: 3
[NOTE] You can add 47 more students.
================================
```

Students that added.

```
==== data of student #1 ====
first name : ibrahim
last name : mohamed
id : 22
gpa: 3.70
His courses ids that registered in:
course No.1 : 1
course No.2 : 2
course No.3 : 3
course No.4 : 4
course No.5 : 5
_____________________________________
==== data of student #2 ====
first name : khaled
last name : ali
id : 44
gpa: 2.90
His courses ids that registered in:
course No.1 : 3
course No.2 : 2
course No.3 : 4
course No.4 : 5
course No.5 : 2
_____________________________________
==== data of student #3 ====
first name : samy
last name : ahmed
id : 55
gpa: 4.00
His courses ids that registered in:
course No.1 : 4
course No.2 : 2
course No.3 : 77
course No.4 : 6
course No.5 : 4
_____________________________________
```

Data in file

```
22 ibrahim mohamed    3.70 1 2 3 4 5
22 mohamed khaled     2.40 5 6 7 8 9
44 khaled ali         2.90 3 2 4 5 2
55 samy ahmed         4.00 4 2 77 6 4
```

NOTE: Don’t add Mohamed because he has same id of ibrahim and ibrahim come first in list.

- Add data of student manually

```
Enter your choice: 2
----------------------

Enter your ID: 99
Enter your First Name: omar
Enter your Last Name: ahmed
Enter your course id:
Course 1 : 3
Course 2 : 4
Course 3 : 6
Course 4 : 6
You are already registered in this course.
Course 4 : 7
Course 5 : 8
Enter your GPA (out of 4): 4.7
Please, try again !!
Enter your GPA (out of 4): 2.9

=== Thanks for your registration ===
[NOTE] Total students: 4
[NOTE] You can add 46 more students.
================================
```

NOTE: Don't repeat register in same course id.

```
==== data of student #1 ====
first name : ibrahim
last name : mohamed
id : 22
gpa: 3.70
His courses ids that registered in:
course No.1 : 1
course No.2 : 2
course No.3 : 3
course No.4 : 4
course No.5 : 5
_____________________________________
==== data of student #2 ====
first name : khaled
last name : ali
id : 44
gpa: 2.90
His courses ids that registered in:
course No.1 : 3
course No.2 : 2
course No.3 : 4
course No.4 : 5
course No.5 : 2
_____________________________________
==== data of student #3 ====
first name : samy
last name : ahmed
id : 55
gpa: 4.00
His courses ids that registered in:
course No.1 : 4
course No.2 : 2
course No.3 : 77
course No.4 : 6
course No.5 : 4
_____________________________________
==== data of student #4 ====
first name : omar
last name : ahmed
id : 99
gpa: 2.90
His courses ids that registered in:
course No.1 : 3
course No.2 : 4
course No.3 : 6
course No.4 : 7
course No.5 : 8
_____________________________________
```

New student come after the students that are registered.

- Search by first name

```
Enter your choice: 3
----------------------

Enter a first name: khaled
Detailed info related to name [khaled]:
        ID: 44
        First Name: khaled
        Last Name: ali
        GPA: 2.90
        Courses IDS:
        Course 1 : 3
        Course 2 : 2
        Course 3 : 4
        Course 4 : 5
        Course 5 : 2
================================
```

```
Enter your choice: 3
----------------------

Enter a first name: yasser
This name [yasser] is not found
```

- Search by ID

```
Enter your choice: 4
-----------------------

Enter an id: 22
Detailed info related to id 22:
ID: 22
First Name: ibrahim
Last Name: mohamed
GPA: 3.70
course : 1
course : 2
course : 3
course : 4
course : 5
================================
```

```
Enter your choice: 4
----------------------

Enter an id: 34
 This id [34] is not found.
```

- Search about specific course id

```
Enter your choice: 5
----------------------

Enter course id: 5
Students registered in this course:
        First Name: ibrahim
        Last Name: mohamed
        ID: 22
        GPA: 3.70
        First Name: khaled
        Last Name: ali
        ID: 44
        GPA: 2.90
Total number in this course 2
================================
```

```
Enter your choice: 5
----------------------

Enter course id: 88
Students registered in this course:
This course id [88] is NOT Found
================================
```

- Top students

```
Enter your choice: 9
----------------------

=== Our Honor Board ===
**First name ibrahim
  Last name mohamed
  ID 22
  GPA 3.70
========================================
```

- Delete student by ID.

```
Enter your choice: 6
----------------------

Enter an id to delete: 55
Student with ID 55 deleted
============================
```

```
Enter your choice: 8
----------------------

==== data of student #1 ====
first name : ibrahim
last name : mohamed
id : 22
gpa: 3.70
His courses ids that registered in:
course No.1 : 1
course No.2 : 2
course No.3 : 3
course No.4 : 4
course No.5 : 5
_____________________________________
==== data of student #2 ====
first name : khaled
last name : ali
id : 44
gpa: 2.90
His courses ids that registered in:
course No.1 : 3
course No.2 : 2
course No.3 : 4
course No.4 : 5
course No.5 : 2
_____________________________________
==== data of student #3 ====
first name : omar
last name : ahmed
id : 99
gpa: 2.90
His courses ids that registered in:
course No.1 : 3
course No.2 : 4
course No.3 : 6
course No.4 : 7
course No.5 : 8
_____________________________________
```

- Update data of student (first name)

```
Enter your choice: 7
----------------------

Enter an id: 22
1. First name
2. Last name
3. ID
4. GPA
5. courses id
Please choose which data to update: 1
Enter a first name: Hima
       Update First Name [Hima] is DONE Successfully
       ======================================================
```

```
Enter your choice: 4
----------------------

Enter an id: 22
Detailed info related to id 22:
ID: 22
First Name: Hima
Last Name: mohamed
GPA: 3.70
course : 1
course : 2
course : 3
course : 4
course : 5
================================
```

```
Enter your choice: 7
-----------------------

Enter an id: 55
Student with id number [55] not found.
=========================================
```

- Update data of student (one of his courses id)

```
Enter your choice: 7
----------------------

Enter an id: 44
1. First name
2. Last name
3. ID
4. GPA
5. courses id
Please choose which data to update: 5
Enter the course ID number to update (1-5): 1
Enter the new course id: 88
        Update New Course id [88] is DONE Successfully
        ================================================
```

```
Enter your choice: 4
----------------------

Enter an id: 44
Detailed info related to id 44:
ID: 44
First Name: khaled
Last Name: ali
GPA: 2.90
course : 88
course : 2
course : 4
course : 5
course : 2
================================
```

- Print all students.

```
Enter your choice: 8
----------------------

==== data of student #1 ====
first name : Hima
last name : mohamed
id : 22
gpa: 3.70
His courses ids that registered in:
course No.1 : 1
course No.2 : 2
course No.3 : 3
course No.4 : 4
course No.5 : 5
_____________________________________________
==== data of student #2 ====
first name : khaled
last name : ali
id : 44
gpa: 2.90
His courses ids that registered in:
course No.1 : 88
course No.2 : 2
course No.3 : 4
course No.4 : 5
course No.5 : 2
_____________________________________________
==== data of student #3 ====
first name : omar
last name : ahmed
id : 99
gpa: 2.90
His courses ids that registered in:
course No.1 : 3
course No.2 : 4
course No.3 : 6
course No.4 : 7
course No.5 : 8
_____________________________________________
```

- Exit from program.

```
Enter your choice: 11
----------------------

Are you sure you want to exit our system? (y/n): y
== Goodbye ==
```