STUDENT INFORMATION MANGEMENT PROJECT(2)

NAME : THOMAS ASHRAF ZAKI

PROFILE : CLICK HERE

GITHUB : CLICK HERE

MAIN.C

```c
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "student_information.h"
//the length of buffer
 vuint32_t length_buffer=1;
//global num to select state
 int check_while=1;
 //pointer in buffer
 element_type* buf_ptr;
void main (){
    element_type student;
    fifo_buffer_t buffer;
     buf_ptr =(element_type*)malloc(length_buffer *sizeof(element_type));
    //to check if fifo return null and slove it
    while(FIFO_INIT( &buffer , buf_ptr,length_buffer )==fifo_null)
    {
        printf("\n[ERROR] buffer return NULL-->i will try again to slove it");
        buf_ptr =(element_type*)malloc(length_buffer *sizeof(element_type));
    }
    printf("\n[INFO] buffer is init");//sucssesful init buffer
    while(check_while){
        int state_num;
        print_states();
        fflush(stdin);fflush(stdout);
        scanf("%d",&state_num);
        printf("\n========================================== ");
        switch(state_num){
        case 1:{
            Add_Student_from_File(&buffer);
            break;
        }
        case 2:{
            Add_Student_Manualy(&buffer);
            break;
        }
        case 3:{
            Find_by_id(&buffer);
            break;
        }
        case 4:{
            find_by_FirstName(&buffer);
            break;
        }
        case 5:{
            find_by_LastName(&buffer);
            break;
        }
        case 6:{
            find_by_GPA(&buffer);
            break;
        }
        case 7:{
            find_by_Registered_Courses(&buffer);
            break;
        }
        case 8:{
            Count_Students(&buffer);
            break;
        }
        case 9:{
            Delete_student(&buffer);
            break;
        }
        case 10:{
            Delete_All_Student(&buffer);
            break;
        }
        case 11:{
            Show_students(&buffer);
            break;
        }
        case 12:{
            Updata_Data_of_student(&buffer);
            break;
        }
        case 13:{
            Exit();
            break;
        }
        default:{
            printf("\n The State NOT FOUND!!");
        }
        }
    }
}
```

STUDENT.TXT

```
1 1 thomas ashraf 3.5 1 22 33 44 9
2 1 marco magdy 3.55 9 44 73 26 5
3 2 kerolos ashraf 4 66 7 1 22 9
4 3 barce ashraf 3.12 67 98 44 24 7
5
```

FIFO.C →INIT

```
fifo_status FIFO_INIT(fifo_buffer_t* fifo_buffer , element_type* buffer ,vuint32_t length )
{
    //check if fifo point null
    if(buffer==NULL)
        return fifo_null;
    //init of buffer
    fifo_buffer->base=buffer;
    fifo_buffer->head=buffer;
    fifo_buffer->tail=buffer;
    fifo_buffer->length=length;
    fifo_buffer->size=length*sizeof(element_type);
    fifo_buffer->count=0;
}
```

FIFO.C →ENQUEUE

```
25  fifo_status FIFO_ENQUEUE(fifo_buffer_t* fifo_buffer ,element_type item )
26  {
27      //check if fifo point null
28      if(!fifo_buffer->head||!fifo_buffer->base||!fifo_buffer->tail)
29          return fifo_null;
30      //check if lifo is full
31      if(FIFO_FULL(fifo_buffer)==fifo_full)
32          return fifo_full;
33      fifo_buffer->count++;//increment count
34      //check if head point on the last of the buffer
35      if (fifo_buffer->head==(fifo_buffer->base+fifo_buffer->size)){
36          fifo_buffer->head=fifo_buffer->base;
37          //add item
38          *(fifo_buffer->head)=item;
39          fifo_buffer->head++;
40      }
41      else{
42          *(fifo_buffer->head)=item;
43          //add item
44          fifo_buffer->head++;
45      }
46
47  }
```

FIFO.C →DEQUEUE

```
48  fifo_status FIFO_DEQUEUE(fifo_buffer_t* fifo_buffer ,element_type* item )
49  {
50      //check if fifo point null
51      if(!fifo_buffer->head||!fifo_buffer->base)
52          return fifo_null;
53      //check if fifo is full
54      if(FIFO_EMPTY(fifo_buffer)==fifo_empty)
55          return fifo_empty;
56      fifo_buffer->count--;//decrement count
57      //check if tail point on the last of the buffer
58      if (fifo_buffer->tail==(fifo_buffer->base+fifo_buffer->size)){
59          fifo_buffer->tail=fifo_buffer->base;
60          //add item
61          *item=*(fifo_buffer->tail);
62      }
63      else{
64          *item=*(fifo_buffer->tail);
65          //add item
66          fifo_buffer->tail++;
67      }
68
69  }
```

FIFO.C →FULL

```
79  fifo_status FIFO_FULL(fifo_buffer_t* fifo_buffer)
80  {
81      //check if fifo point null
82      if(!fifo_buffer->head||!fifo_buffer->base||!fifo_buffer->tail)
83          return fifo_null;
84      if(fifo_buffer->count==fifo_buffer->length)
85          return fifo_full;
86      return fifo_no_error;
87  }
```

FIFO.C →EMPTY

```
70  fifo_status FIFO_EMPTY(fifo_buffer_t* fifo_buffer)
71  {
72      //check if fifo point null
73      if(!fifo_buffer->head||!fifo_buffer->base||!fifo_buffer->tail)
74          return fifo_null;
75      if(fifo_buffer->count==0)
76          return fifo_empty;
77      return fifo_no_error;
78  }
```

STUDENT.H

```c
#ifndef STUDENT_INFORMATION_H_
#define STUDENT_INFORMATION_H_

#include "platform_t.h"
//typedefs

#define element_type SData_Student_t
typedef enum {    //status of error
    fifo_null,
    fifo_no_error,
    fifo_full,
    fifo_empty
}fifo_status;
//data of each student
typedef struct {
    char first_name [30];
    char last_name [30];
    unsigned int unique_roll;
    float GPA;
    unsigned int ID_Courses[5];
}SData_Student_t;

typedef struct {
    element_type* base;
    element_type* head;
    element_type* tail;
    vuint32_t length;
    vuint32_t size;
    vuint32_t count;
}fifo_buffer_t;
```

```c
//APIS
void Add_Student_from_File(fifo_buffer_t* fifo_buffer );
void Add_Student_Manualy( fifo_buffer_t* fifo_buffer );
void Find_by_id(fifo_buffer_t* fifo_buffer);
void find_by_FirstName(fifo_buffer_t* fifo_buffer);
void find_by_LastName(fifo_buffer_t* fifo_buffer);
void find_by_GPA(fifo_buffer_t* fifo_buffer);
void find_by_Registered_Courses(fifo_buffer_t* fifo_buffer);
void Count_Students(fifo_buffer_t* fifo_buffer);
void Delete_student(fifo_buffer_t* fifo_buffer);
void Delete_All_Student(fifo_buffer_t* fifo_buffer);
void Updata_Data_of_student(fifo_buffer_t* fifo_buffer);
void Show_students(fifo_buffer_t* fifo_buffer);
void increment_length_buf(fifo_buffer_t* fifo_buffer);
void print_states();
void enter_student(char* line, element_type* student);
void Exit();

//APIS of fifo
fifo_status FIFO_ENQUEUE(fifo_buffer_t* fifo_buffer ,element_type item );
fifo_status FIFO_DEQUEUE(fifo_buffer_t* fifo_buffer ,element_type* item );
fifo_status FIFO_INIT(fifo_buffer_t* fifo_buffer , element_type* buffer ,vuint32_t length );
fifo_status FIFO_EMPTY(fifo_buffer_t* fifo_buffer);
fifo_status FIFO_FULL(fifo_buffer_t* fifo_buffer);
void FIFO_PRINT(fifo_buffer_t* fifo_buffer);
#endif /* STUDENT_INFORMATION_H_ */
```

STUDENT.C →#INCLUDE

```c
#include "stdio.h"
#include "student_information.h"
#include "platform_t.h"
#include "stdbool.h"

//global num to select state
extern int check_while;
extern element_type* buf_ptr;
int check file=0;
```

STUDENT.C →INCREMENT_LENGTH_BUF()

- This fun when call it ,it increment length and stop until 50 ,you can add normally by call it when add new student
- Why?? To save memory from allocate 50 element and not used all it

```
522  void increment_length_buf(fifo_buffer_t* fifo_buffer){//to add anther element if you add again by limitting with 50 student
523      if(fifo_buffer->count<50){
524          fifo_buffer->length++;
525          fifo_buffer->size=fifo_buffer->length*sizeof(element_type);
526      }
527  }
```

STUDENT.C →ADD_STUDENT_FROM_FILE()

```
//APIS
void Add_Student_from_File(fifo_buffer_t* fifo_buffer) {
    if (check_file) {
        printf("\n[ERROR] You have added this data before.");
        return;
    }

    FILE *file = fopen("students.txt", "r");
    if (file == NULL) {
        printf("\n[ERROR] The file could not be opened or is empty.");
        return;
    }

    char line[256];
    element_type item;

    while (fgets(line, sizeof(line), file)) {
        enter_student(line, &item);

        // Check if the student ID already exists in the FIFO buffer
        int check = 0;
        for (int i = 0; i < fifo_buffer->count; i++) {
            element_type existing_student = fifo_buffer->tail[i]; // Use the tail to access the elements
            if (item.unique_roll == existing_student.unique_roll) {
                printf("\n[ERROR] NOT ADDING %s %s due to duplicate ID with %s %s",
                       item.first_name, item.last_name,
                       existing_student.first_name, existing_student.last_name);
                printf("\n==========================");
                check = 1; // Set flag to not enqueue this student
                break; // Exit the loop early
            }
        }

        if (!check) {
            // Attempt to enqueue the new student
            fifo_status status = FIFO_ENQUEUE(fifo_buffer, item);
            if (status == fifo_full) {
                printf("\n[ERROR] The list is full; please delete a student.");
            } else {
                // Increment length only after successful addition
                increment_length_buf(fifo_buffer);
                printf("\n[INFO] ADDING %s %s to the list", item.first_name, item.last_name);
                printf("\n==========================");
            }
        }
    }

    check_file = 1; // Mark that data has been added
    Count_Students(fifo_buffer); // Count and print the remaining students
    fclose(file); // Close the file after reading
}
```

STUDENT.C →ENTER_STUDENT()

```
529  void enter_student(char* line, element_type* student) {
530      // Assuming the file format is: roll first_name last_name GPA id1 id2 id3 id4 id5
531      sscanf(line, "%u %s %s %f %u %u %u %u %u",
532              &student->unique_roll,
533              student->first_name,
534              student->last_name,
535              &student->GPA,
536              &student->ID_Courses[0],
537              &student->ID_Courses[1],
538              &student->ID_Courses[2],
539              &student->ID_Courses[3],
540              &student->ID_Courses[4]);
541  }
```

STUDENT.C →ADD_STUDENT_MANUALY()

```c
void Add_Student_Manualy(fifo_buffer_t* fifo_buffer){
    //init the parameters
    element_type item ;
    //enter First Name
    printf("\n Enter First Name : ");
    fflush(stdin);fflush(stdout);
    scanf("%s",item.first_name);
    //enter last Name
    printf("\n Enter last Name : ");
    fflush(stdin);fflush(stdout);
    scanf("%s",item.last_name);
    //enter unique roll
    printf("\n Enter unique roll : ");
    fflush(stdin);fflush(stdout);
    scanf("%u",&item.unique_roll);
    //========check if he enter same id============
    fifo_buffer_t temp=*fifo_buffer;//this cannot be pointer becouase it reach to reference(*-->becouse i use fifo_buffer_t temp not fifo_buffer_t*temp)

    while(temp.count)
    {
        if(item.unique_roll==(temp.tail->unique_roll)){
            printf("\n [ERROR]NOT ADD %s %s that have same id of %s %s",item.first_name,item.last_name,temp.tail->first_name,temp.tail->last_name);
            return;
        }
        else{
            temp.count--;
            temp.tail++;//to loop in buffer checking the unique_roll
        }
    }
    //enter GPA
    printf("\n Enter GPA : ");
    fflush(stdin);fflush(stdout);
    scanf("%f",&item.GPA);
    //enter ID Courses
    printf("\n Enter ID Courses");
    //loop to enter 5 courses
    for(int i=0;i<5;i++)
    {
        printf("\n \t Enter ID Course(%d) : ",i+1);
        fflush(stdin);fflush(stdout);
        scanf("%u",&item.ID_Courses[i]);
        //to check if this student add the same id course twice
        for(int j=0;j<i;j++)
        {
            if(item.ID_Courses[i]==item.ID_Courses[j])// to check that this not the same element in the array(i want to check with the anthors )
            {
                printf("\n-------------------------");
                printf("\n[ERROR] this id course added before-->enter the true id course");
                printf("\n-------------------------");
                i--;//to return enter this element
                continue;// go to enter again
            }
        }
    }
    printf("\n========================");
    if(FIFO_ENQUEUE(fifo_buffer,item)==fifo_full)//if buffer full
    {
        printf("\n[ERROR] the list is full please delete student");
    }
    else{//i want to increase length every time when add (to donot take space in memory the user donnot use it now)
        increment_length_buf(fifo_buffer);//call
        printf("\n[INFO]Student Details is added ");
        printf("\n========================");
        //to count and print the remain
        Count_Students(fifo_buffer);//call
    }
}
```

STUDENT.C →FIND_BY_ID()

```
126 void Find_by_id(fifo_buffer_t* fifo_buffer){
127     //check empty
128     //-----------------------
129     if(fifo_buffer->count){
130         //variable of id
131         unsigned int id;
132         //enter id to find student
133         printf("\n [INFO] Enter unique roll to find student: ");
134         fflush(stdin);fflush(stdout);
135         scanf("%u",&id);
136         //========check if this ID is found=============
137         fifo_buffer_t temp=*fifo_buffer;//this cannot be pointer becouase it reach to reference(*-->becouse i use fifo_buffer_t temp not fifo_buffer_t*temp)
138         while(temp.count){
139             if(id==(temp.tail->unique_roll)){
140                 printf("\n ============================");
141                 printf("\n This is student details :");
142                 printf("\n The first name : %s",temp.tail->first_name);
143                 printf("\n-------------------------------------------");
144                 printf("\n The last name : %s",temp.tail->last_name);
145                 printf("\n-------------------------------------------");
146                 printf("\n The unique roll : %u",temp.tail->unique_roll);
147                 printf("\n-------------------------------------------");
148                 printf("\n The GPA : %f",temp.tail->GPA);
149                 printf("\n-------------------------------------------");
150                 printf("\n ID Courses is");
151                 //loop to enter 5 courses
152                 for(int i=0;i<5;i++)
153                 {
154                     printf("\n \t the Course(%d) id : %u ",i+1,temp.tail->ID_Courses[i]);}
155                 return;}
156             else{
157                 temp.count--;
158                 temp.tail++;//to loop in buffer checking the unique_roll
159             }}
160         printf("\n[ERROR]This ID = (%d) NOT FOUND",id);
161     }
162     else{
163         printf("\n[ERROR] NO Student added");    }
164 }
```

STUDENT.C →FIND_BY_FIRSTNAME()

```
166 void find_by_FirstName(fifo_buffer_t* fifo_buffer){
167     if(fifo_buffer->count){//check empty
168         //array of first name
169         unsigned char first_name[30];
170         //enter first name to find student
171         printf("\n [INFO] Enter first name to find student: ");
172         fflush(stdin);fflush(stdout);
173         scanf("%s",first_name);
174         //========check if this first name is found=============
175         fifo_buffer_t temp=*fifo_buffer;//this cannot be pointer becouase it reach to reference(*-->becouse i use fifo_buffer_t temp not fifo_buffer_t*temp)
176         boolean check =FALSE;//to check if never have this
177         while(temp.count){
178             if(strcmp(first_name, temp.tail->first_name)==0){//strcmp compare between two string to check if it return 0 it is equal
179                 check=TRUE;//becouse i find item that have the searched name
180                 printf("\n ============================");
181                 printf("\n This is student details :");
182                 printf("\n The first name : %s",temp.tail->first_name);
183                 printf("\n-------------------------------------------");
184                 printf("\n The last name : %s",temp.tail->last_name);
185                 printf("\n-------------------------------------------");
186                 printf("\n The unique roll : %u",temp.tail->unique_roll);
187                 printf("\n-------------------------------------------");
188                 printf("\n The GPA : %f",temp.tail->GPA);
189                 printf("\n-------------------------------------------");
190                 printf("\n ID Courses is");
191                 //loop to enter 5 courses
192                 for(int i=0;i<5;i++){
193                     printf("\n \t the Course(%d) id : %u ",i+1,temp.tail->ID_Courses[i]);}
194                 printf("\n==============================");}
195             //i made this becouse it will be infinte without those ,they help to enter the next item without go to else
196             temp.count--;
197             temp.tail++;
198             //i donot make return becouse we can find two student that have same first name
199         }
200         if(!check)
201             printf("\n[ERROR]This first name : (%s) NOT FOUND",first_name); }
202     else{
203         printf("\n[ERROR] NO Student added");
204     }}
```

STUDENT.C →FIND_BY_LASTNAME()

```
206 void find_by_LastName(fifo_buffer_t* fifo_buffer){
207     if(fifo_buffer->count){//check empty
208         //array of last name
209         unsigned char last_name[30];
210         //enter last name to find student
211         printf("\n [INFO] Enter last name to find student: ");
212         fflush(stdin);fflush(stdout);
213         scanf("%s",last_name);
214         //========check if this last name is found============
215         fifo_buffer_t temp=*fifo_buffer;//this cannot be pointer becouase it reach to reference(*-->becouse i use fifo_buffer_t temp not fifo_buffer_t*temp)
216         boolean check =FALSE;//to check if never have this
217         while(temp.count){
218             if(strcmp(last_name, temp.tail->last_name)==0){//strcmp compare between two string to check if it return 0 it is equal
219                 check=TRUE;//becouse i find item that have the searched name
220                 printf("\n ============================");
221                 printf("\n This is student details :");
222                 printf("\n The first name : %s",temp.tail->first_name);
223                 printf("\n-------------------------------------------");
224                 printf("\n The last name : %s",temp.tail->last_name);
225                 printf("\n-------------------------------------------");
226                 printf("\n The unique roll : %u",temp.tail->unique_roll);
227                 printf("\n-------------------------------------------");
228                 printf("\n The GPA : %f",temp.tail->GPA);
229                 printf("\n-------------------------------------------");
230                 printf("\n ID Courses is");
231                 //loop to enter 5 courses
232                 for(int i=0;i<5;i++){
233                     printf("\n \t the Course(%d) id : %u ",i+1,temp.tail->ID_Courses[i]);}
234                 printf("\n================================");}
235             //i made this becouse it will be infinte without those ,they help to enter the next item without go to else
236             temp.count--;
237             temp.tail++;
238             //i donot make return becouse we can find two student that have same last name
239         }
240         if(!check)
241             printf("\n[ERROR]This last name : (%s) NOT FOUND",last_name);   }
242     else{
243         printf("\n[ERROR] NO Student added");
244     }}
```

STUDENT.C →FIND_BY_GPA()

```
246  void find_by_GPA(fifo_buffer_t* fifo_buffer){
247      if(fifo_buffer->count){//check empty
248          //variable of GPA
249          float GPA;
250          //enter GPA to find student
251          printf("\n [INFO] Enter GPA to find student: ");
252          fflush(stdin);fflush(stdout);
253          scanf("%f",&GPA);
254          //========check if this GPA is found=============
255          fifo_buffer_t temp=*fifo_buffer;//this cannot be pointer becouase it reach to reference(*-->becouse i use fifo_buffer_t temp not fifo_buffer_t*temp)
256          boolean check =FALSE;//to check if never have this
257          while(temp.count){
258              if(GPA==temp.tail->GPA){
259                  check=TRUE;//becouse i find item that have the searched name
260                  printf("\n ============================");
261                  printf("\n This is student details :");
262                  printf("\n The first name : %s",temp.tail->first_name);
263                  printf("\n-------------------------------------------");
264                  printf("\n The last name : %s",temp.tail->last_name);
265                  printf("\n-------------------------------------------");
266                  printf("\n The unique roll : %u",temp.tail->unique_roll);
267                  printf("\n-------------------------------------------");
268                  printf("\n The GPA : %f",temp.tail->GPA);
269                  printf("\n-------------------------------------------");
270                  printf("\n ID Courses is");
271                  //loop to enter 5 courses
272                  for(int i=0;i<5;i++){
273                      printf("\n \t the Course(%d) id : %u ",i+1,temp.tail->ID_Courses[i]);   }
274                  printf("\n================================");  }
275              //i made this becouse it will be infinte without those ,they help to enter the next item without go to else
276              temp.count--;
277              temp.tail++;
278              //i donot make return becouse we can find two student that have same last name
279          }
280          if(!check)
281              printf("\n[ERROR]This GPA : (%f) NOT FOUND",GPA);}
282      else{
283          printf("\n[ERROR] NO Student added");
284      }}
```

STUDENT.C →FIND_BY_REGISTERED_COURSES()

```
286  void find_by_Registered_Courses(fifo_buffer_t* fifo_buffer){
287          if(fifo_buffer->count){ //check empty
288          //array of id course
289          unsigned int id_course;
290          //enter id course to find student
291          printf("\n [INFO] Enter id course to find student: ");
292          fflush(stdin);fflush(stdout);
293          scanf("%u",&id_course);
294          //========check if this id course is found=============
295          fifo_buffer_t temp=*fifo_buffer;//this cannot be pointer becouase it reach to reference(*-->becouse i use fifo_buffer_t temp not fifo_buffer_t*temp)
296          boolean check =FALSE;//to check if never have this
297          while(temp.count){
298              for(int i=0;i<5;i++){//to check the id from the 5 courses
299                  if(id_course==temp.tail->ID_Courses[i]){
300                      check=TRUE;//becouse i find item that have the searched name
301                      printf("\n ==============================");
302                      printf("\n This is student details :");
303                      printf("\n The first name : %s",temp.tail->first_name);
304                      printf("\n-----------------------------------------------");
305                      printf("\n The last name : %s",temp.tail->last_name);
306                      printf("\n-----------------------------------------------");
307                      printf("\n The unique roll : %u",temp.tail->unique_roll);
308                      printf("\n-----------------------------------------------");
309                      printf("\n The GPA : %f",temp.tail->GPA);
310                      printf("\n-----------------------------------------------");
311                      printf("\n ID Courses is");
312                      //loop to enter 5 courses
313                      for(int j=0;j<5;j++){// becouse there ara i in outter loop
314                          printf("\n \t the Course(%d) id : %u ",j+1,temp.tail->ID_Courses[j]);}
315                      printf("\n================================");
316                  }   }
317              //i made this becouse it will be infinte without those ,they help to enter the next item without go to else
318              temp.count--;
319              temp.tail++;
320              //i donot make return becouse we can find two student that have same id course      }
321          if(!check)
322              printf("\n[ERROR]This id course : (%u) NOT FOUND",id_course);   }
323      else{
324          printf("\n[ERROR] NO Student added");
```

STUDENT.C →COUNT_STUDENT()

```
327  void Count_Students(fifo_buffer_t* fifo_buffer){
328      printf("\n[INFO]the total number of student =  %d",fifo_buffer->count);
329      printf("\n[INFO]you can up to 50 student");
330      printf("\n[INFO]Now you add %d student ",50-fifo_buffer->count);
331  }
```

STUDENT.C →DELETE_STUDENT()

```
void Delete_student(fifo_buffer_t* fifo_buffer){
    //check empty
    if(fifo_buffer->count){
        //variable of id
        unsigned int id;
        //enter id to delete student
        printf("\n [INFO] Enter unique roll to delete student: ");
        fflush(stdin);fflush(stdout);
        scanf("%u",&id);
        //========check if this ID is found=============
        //you can fifo_buffer_t temp=*fifo_buffer buuuut i prefere to use that to protect stack from alot of variables i donnot use
        vuint32_t count=fifo_buffer->count;
        element_type* temp=fifo_buffer->tail;
        while(count)
        {
            if(id==(temp->unique_roll)){//condition of check id
                //printf("\n %p       %p",temp,fifo_buffer->head);-->to know where is temp and head
                while((fifo_buffer->head-1)!=temp){//now i detect the space i want to delete .i will not use free becouse i cannot reach to this space again
                    //i slove that by take the next value temp+1 to temp until reach to head-1 that can i touch the end value
                    *(temp)=*(temp+1);
                    temp++;
                }
                fifo_buffer->head--;//after that condition is fail and make head-- to return one step becouse the last two element is equal
                fifo_buffer->length--;//i decrement the length becouse i delete one element
                printf("\n [INFO] The student with this ID : %u is deleted",id);
                fifo_buffer->count--;
                //printf("\n %p     %p",temp,fifo_buffer->head);here temp==head
                //i want to fill this free area with
                return;
            }
            else{
                count--;//i make it value that take the value of fifo_buffer->count becouse i donot need to
                temp++;//to loop in buffer checking the unique_roll
            }       }
        printf("\n[ERROR]This ID = (%d) NOT FOUND",id);
    }
    else{
        printf("\n[ERROR] NO Student added");
    }}
```

STUDENT.C →DELETE_ALL_STUDENT()

```
373  void Delete_All_Student(fifo_buffer_t* fifo_buffer){
374      //check empty
375      //-----------------------
376      if(fifo_buffer->count){
377          fifo_buffer_t* temp=fifo_buffer;
378          while(temp->count){
379              temp->tail=fifo_buffer->tail;
380              fifo_buffer->tail++;
381              free(temp->tail);
382              temp->count--;
383          }
384          printf("\n [INFO] ALL Student deleted.");
385      }
386      else{
387          printf("\n[ERROR] NO Student added");}
388  }
```

STUDENT.C →SHOW_STUDENTS()

```
487  void Show_students(fifo_buffer_t* fifo_buffer){
488      //check empty
489      //----------------------
490      if(fifo_buffer->count){
491          fifo_buffer_t temp=*fifo_buffer;//this cannot be pointer becouase it reach to reference(*-->becouse i use fifo_buffer_t temp not fifo_buffer_t*temp)
492          while(temp.count)
493          {
494              printf("\n ============================");
495              printf("\n This is student details :");
496              printf("\n The first name : %s",temp.tail->first_name);
497              printf("\n-------------------------------------------");
498              printf("\n The last name : %s",temp.tail->last_name);
499              printf("\n-------------------------------------------");
500              printf("\n The unique roll : %u",temp.tail->unique_roll);
501              printf("\n-------------------------------------------");
502              printf("\n The GPA : %f",temp.tail->GPA);
503              printf("\n-------------------------------------------");
504              printf("\n ID Courses is");
505              //loop to enter 5 courses
506              for(int i=0;i<5;i++){
507                  printf("\n \t the Course(%d) id : %u ",i+1,temp.tail->ID_Courses[i]);
508              }
509              printf("\n================================");
510              //i made this becouse it will be infinte without those ,they help to enter the next item without go to else
511              temp.count--;
512              temp.tail++;
513              //i donot make return becouse we can find two student that have same last name
514          }
515      }
516      //=================
517      else{
518          printf("\n[ERROR] NO Student added");
519      }
520  }
```

STUDENT.C →UPDATE_DATA_STUDENT()

```
390  void Updata_Data_of_student(fifo_buffer_t* fifo_buffer){
391      //check empty
392      //-----------------------
393      if(fifo_buffer->count){
394          //variable of id
395          unsigned int id;
396          //enter id to delete student
397          printf("\n [INFO] Enter unique roll to update data of student: ");
398          fflush(stdin);fflush(stdout);
399          scanf("%u",&id);
400          //========check if this ID is found=============
401          //you can fifo_buffer_t temp=*fifo_buffer buuuut i prefere to use that to protect stack from alot of variables i donnot use
402          vuint32_t count=fifo_buffer->count;
403          element_type* temp=fifo_buffer->tail;
404          while(count)
405          {
406              if(id==(temp->unique_roll)){//condition of check id
407                  printf("\n [INFO] Each of this you want to update");
408                  printf("\n \t 1- First Name");
409                  printf("\n \t 2- Second Name");
410                  printf("\n \t 3- ID");
411                  printf("\n \t 4- GPA");
412                  printf("\n \t 5- Registered Courses");
413                  //variable of check
414                  unsigned int check;
415                  //enter num to update
416                  printf("\n [INFO] Enter number of the last list : ");
417                  fflush(stdin);fflush(stdout);
418                  scanf("%u",&check);
419                  //to chose each of them i will update
420                  switch(check){
421                  case 1:{
422                      printf("\n [INFO] update the first name : ");
423                      fflush(stdin);fflush(stdout);
424                      scanf("%s",temp->first_name);
425                      break;
426                  }
```

STUDENT.C →UPDATE_DATA_STUDENT()

```
427         case 2:{
428             printf("\n [INFO] update the last name : ");
429             fflush(stdin);fflush(stdout);
430             scanf("%s",temp->last_name);
431             break;
432         }
433         case 3:{
434             printf("\n [INFO] update the ID : ");
435             fflush(stdin);fflush(stdout);
436             scanf("%u",&temp->unique_roll);
437             break;
438         }
439         case 4:{
440             printf("\n [INFO] update the GPA : ");
441             fflush(stdin);fflush(stdout);
442             scanf("%f",&temp->GPA);
443             break;
444         }
445         case 5:{
446             printf("\n [INFO] update the Registered courses : ");
447             for(int i=0;i<5;i++)
448             {
449                 printf("\n \t Enter ID Course(%d) : ",i+1);
450                 fflush(stdin);fflush(stdout);
451                 scanf("%u",&temp->ID_Courses[i]);
452                 //to check if this student add the same id course twice
453                 for(int j=0;j<i;j++)
454                 {
455                     if(temp->ID_Courses[i]==temp->ID_Courses[j])// to check that this not the same element in the array(i want to check with the anthors )
456                     {
457                         printf("\n-------------------------");
458                         printf("\n[ERROR] this id course added before-->enter the true id course");
459                         printf("\n-------------------------");
460                         i--;//to return enter this element
461                         continue;// go to enter again
462                     }
463                 }
464             }
465             break;
```

```
466                 }
467                 default:{
468                     printf("\n -------");
469                     printf("\n [ERROR] You enter invalid number!!!!!");
470                     printf("\n -------");
471                     continue;//retrn to while again
472                 }}
473                 return;
474             }
475             else{
476                 count--;//i make it value that take the value of fifo_buffer->count becouse i donot need to
477                 temp++;//to loop in buffer checking the unique_roll
478             }
479         }
480         printf("\n[ERROR]This ID = (%d) NOT FOUND",id);//if i donot found the id
481     }
482     else{
483         printf("\n[ERROR] NO Student added");
484     }
485 }
```

STUDENT.C →PRINT_STATES()
STUDENT.C →EXIT()

```
543  void print_states(){
544      printf("\n====================================");
545      printf("\n Chose from this states");
546      printf("\n \t 1- Add Student from file");
547      printf("\n \t 2- Add Student manually");
548      printf("\n \t 3- find student by ID");
549      printf("\n \t 4- find student by first name");
550      printf("\n \t 5- find student by last name");
551      printf("\n \t 6- find student by GPA");
552      printf("\n \t 7- find student by registered courses");
553      printf("\n \t 8- count the students");
554      printf("\n \t 9- delete student ");
555      printf("\n \t 10- delete ALl student ");
556      printf("\n \t 11- view All student ");
557      printf("\n \t 12- update student ");
558      printf("\n \t 13- Exit from program ");
559      printf("\n Enter the state : ");//enter state
560  }
561  void Exit(){
562      check_while=0;
563      printf("\n Thanks..\n program end!!");
564  }
565  //===============END================================
566
```

VIDEO OF TESTING

link : click here

THANKS

- Under the supervision of
- Eng: Keroloes Shenouda Khalil (here)