



MASTERING EMBEDDED SYSTEM ONLINE DIPLOMA

www.learn-in-depth.com



FIRST TERM - PROJECT 2

ENG. AHMED ESSAM ELDIN ELSHAFIE

<https://www.learn-in-depth.com/online-diploma/ahmedessam020@gmail.com>

Table of Contents

1 PROBLEM STATEMENT	2
2 APPROACH	2
3 IMPLEMENTATION.....	3
3.1 main.c.....	3
3.2 queue.h	4
3.3 queue.c.....	5
4 OUTPUT.....	13
4.1 addStudentManually()	13
4.2 addStudentFromFile()	14
4.3 findByRoll()	15
4.4 findByFirstName()	16
4.5 findByCourseId().....	17
4.6 totalStudentCount()	18
4.7 deleteStudent()	19
4.8 updateStudent()	20
4.9 showAll()	21
4.10 Other Features	22

STUDENT INFO MANAGEMENT SYSTEM

1 PROBLEM STATEMENT

This project is mainly about implementing a software system to manage the students' information regarding the following:

- First Name.
- Last Name.
- GPA .
- Unique Roll Number.
- Current Enrolled Courses.

2 APPROACH

The idea is to form an individual function for each operation. All the functions are unified to form a software system. The functions needed to be implemented are expected to be as following:

1. Add Student Details from File.
2. Add Student Details manually.
3. Find Student by given Roll Number.
4. Find Student by given First Name.
5. Find Students enrolled in a course.
6. Count of students.
7. Delete a student.
8. Update a student.
9. View all info.

We will start implementing each of the following points as a callable functions to be called by user within entry.

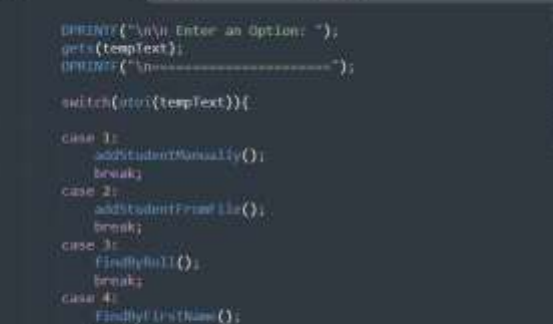
3.1 main.c

```

1  /*
2   * main.c for Student Info Management System
3   * Eng. Ahmed Essam
4   */
5
6  #include "queue.h"
7
8  int main()
9  {
10
11     char tempText[40];
12
13     if(queue_init() == queueError){
14         printf("\n*****Student Management System*****");
15     }
16     else{
17         printf("\nFailed to initialize Queue");
18         return 0;
19     }
20
21     while(1){
22
23         printf("\n=====");
24         printf("\nList of Available Options:\n");
25         printf("\n 1: Add New Student Manually");
26         printf("\n 2: Add New Student/s from Text File");
27         printf("\n 3: Find Student by Roll Number");
28         printf("\n 4: Find Student/s by First Name");
29         printf("\n 5: Find Student/s by Course ID");
30         printf("\n 6: Total Number of Students");
31         printf("\n 7: Delete Student by Roll Number");
32         printf("\n 8: Update Student by Roll Number");
33         printf("\n 9: View All Students Info");
34         printf("\n 10: Exit");
35
36         printf("\n\nEnter an Option: ");
37         gets(tempText);
38         printf("\n-----");
39
40         switch(atoi(tempText)){

```

Figure 1 - main.c 1/2



The screenshot shows a C++ IDE with a dark theme. The main window displays a C++ program with a switch statement. The code is as follows:

```

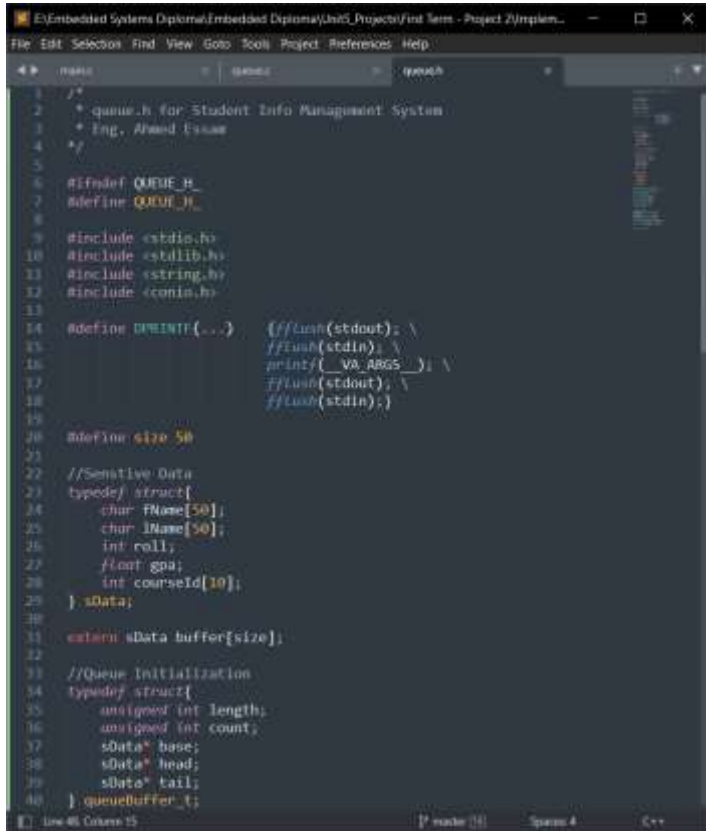
24
25     PRINTF("\n\n Enter an Option: ");
26     gets(tempText);
27     PRINTF("\n\n-----");
28
29     switch(atoi(tempText)){
30
31     case 1:
32         addStudentManually();
33         break;
34     case 2:
35         addStudentFromFile();
36         break;
37     case 3:
38         findById();
39         break;
40     case 4:
41         findByName();
42         break;
43     case 5:
44         findByIdAndName();
45         break;
46     case 6:
47         totalStudentCount();
48         break;
49     case 7:
50         deleteStudent();
51         break;
52     case 8:
53         updateStudent();
54         break;
55     case 9:
56         showAll();
57         break;
58     case 10:
59         return 0;
60     }
61     return 0;
62 }

```

The IDE interface includes a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help), a toolbar, and a status bar at the bottom showing "Line 16, Column 6", "main.c", "Spaces: 4", and a C++ icon.

Figure 2 - main.c 2/2

3.2 queue.h



```
1  /**
2   * queue.h for Student Info Management System
3   * Eng. Ahmed Essam
4   */
5
6  #ifndef QUEUE_H
7  #define QUEUE_H
8
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <string.h>
12 #include <conio.h>
13
14 #define DEBUG(...) { fflush(stdout); \
15                     fflush(stdin); \
16                     printf(_VA_ARGS_); \
17                     fflush(stdout); \
18                     fflush(stdin); }
19
20 #define size 50
21
22 //Sensitive Data
23 typedef struct{
24     char FName[50];
25     char LName[50];
26     int roll;
27     float gpa;
28     int courseId[10];
29 } sData;
30
31 extern sData buffer[size];
32
33 //Queue Initialization
34 typedef struct{
35     unsigned int length;
36     unsigned int count;
37     sData* base;
38     sData* head;
39     sData* tail;
40 } queueBuffer_t;
```

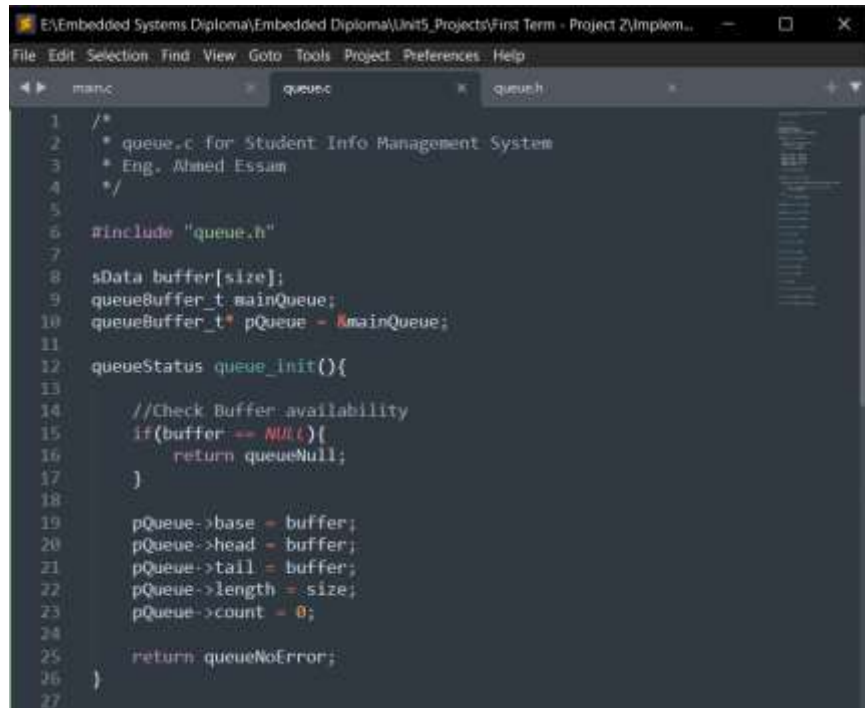
Figure 3 - queue.h 1/2



```
41 //Queue Initialization
42 typedef struct{
43     unsigned int length;
44     unsigned int count;
45     sData* base;
46     sData* head;
47     sData* tail;
48 } queueBuffer_t;
49
50 //Queue Status
51 typedef enum{
52     queueNoError,
53     queueFull,
54     queueEmpty,
55     queueNull
56 } queueStatus;
57
58 //Main APIs to be called by user
59 void addStudentManually();
60 void addStudentFromFile();
61 void findByName();
62 void findByFirstName();
63 void findByCourseId();
64 void totalStudentCount();
65 void deleteStudent();
66 void updateStudent();
67 void showAll();
68
69 //Internal APIs
70 queueStatus queue_init();
71 queueStatus queue_isValid();
72 queueStatus queue_isEmpty();
73 queueStatus queue_isFull();
74 int roll_isUnique(int rollcheck);
75 void printInfo(sData* student);
76 void updateInfo(sData* student);
77
78 #endif /* QUEUE_H */
```

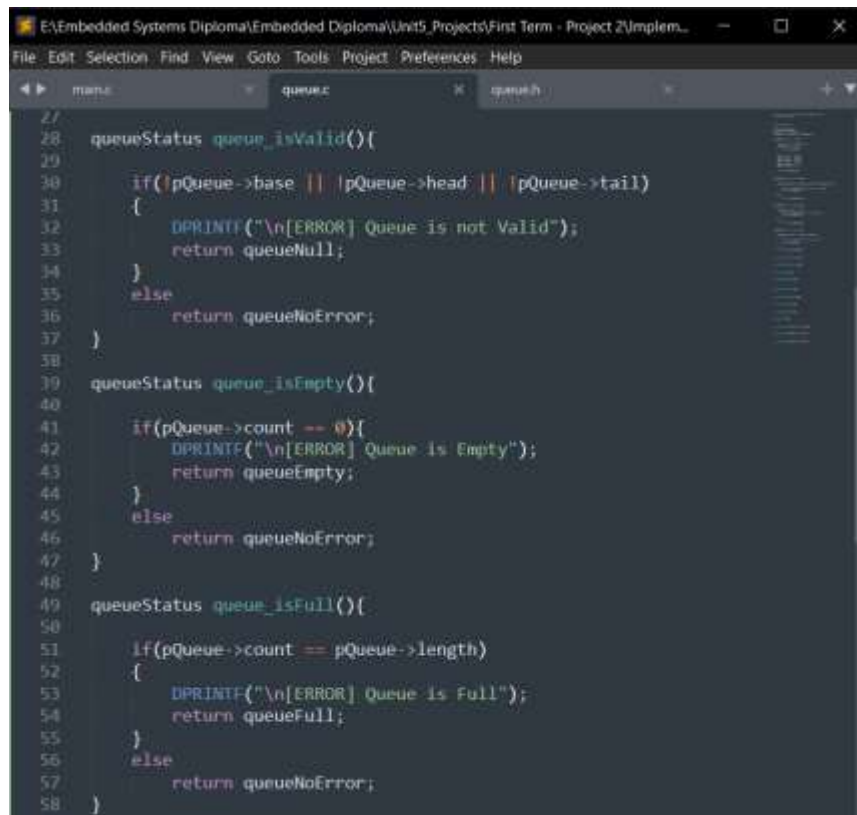
Figure 4 - queue.h 2/2

3.3 queue.c



```
1  /*
2  * queue.c for Student Info Management System
3  * Eng. Ahmed Essam
4  */
5
6  #include "queue.h"
7
8  sData buffer[size];
9  queueBuffer_t mainQueue;
10 queueBuffer_t* pQueue = &mainQueue;
11
12 queueStatus queue_init(){
13
14     //Check Buffer availability
15     if(buffer == NULL){
16         return queueNull;
17     }
18
19     pQueue->base = buffer;
20     pQueue->head = buffer;
21     pQueue->tail = buffer;
22     pQueue->length = size;
23     pQueue->count = 0;
24
25     return queueNoError;
26 }
27
```

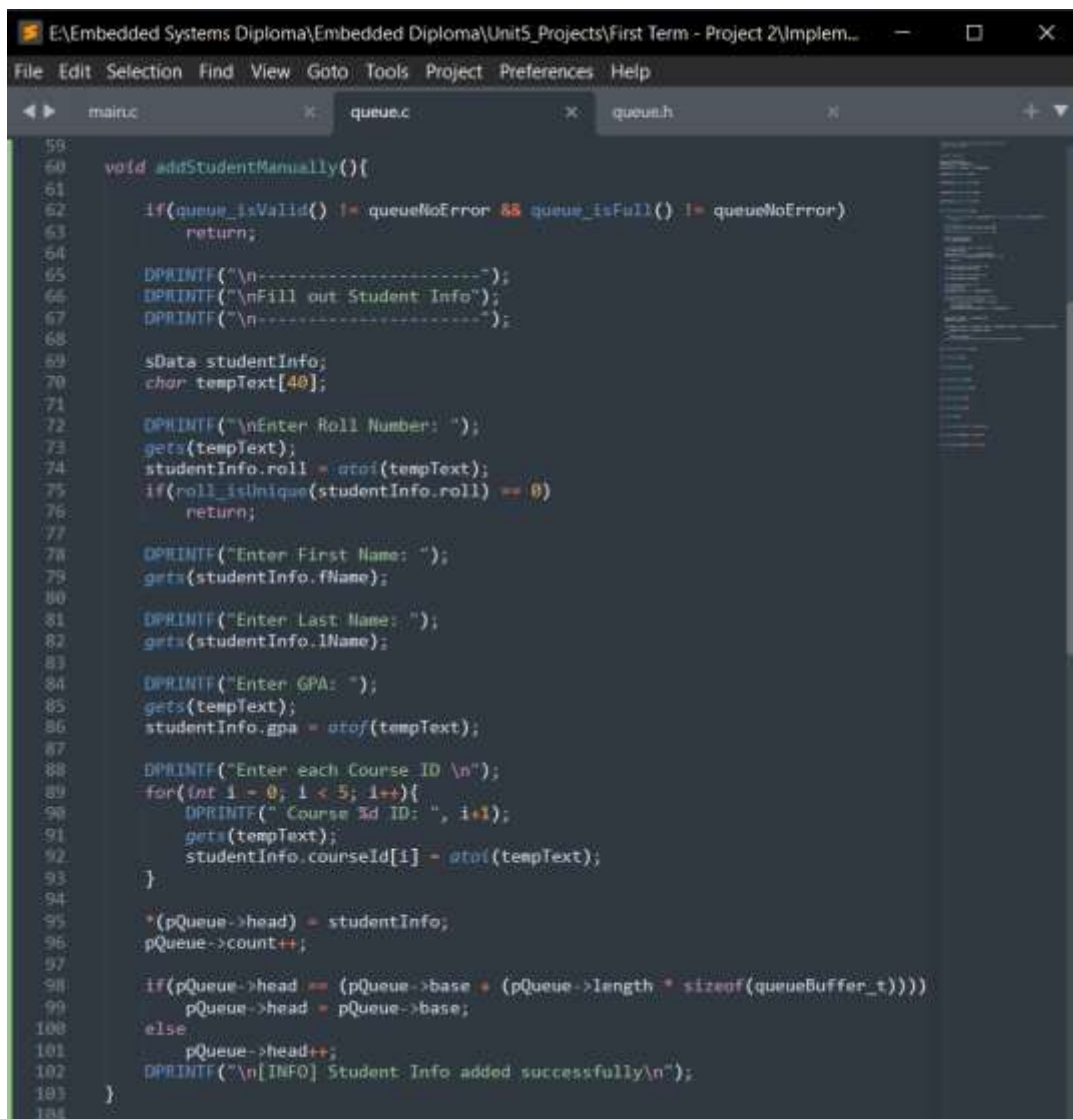
Figure 5 - Global vars and queue_init()



```
28 queueStatus queue_isValid(){
29
30     if(!pQueue->base || !pQueue->head || !pQueue->tail)
31     {
32         DPRINTF("\n[ERROR] Queue is not Valid");
33         return queueNull;
34     }
35     else
36         return queueNoError;
37 }
38
39 queueStatus queue_isEmpty(){
40
41     if(pQueue->count == 0){
42         DPRINTF("\n[ERROR] Queue is Empty");
43         return queueEmpty;
44     }
45     else
46         return queueNoError;
47 }
48
49 queueStatus queue_isFull(){
50
51     if(pQueue->count == pQueue->length)
52     {
53         DPRINTF("\n[ERROR] Queue is Full");
54         return queueFull;
55     }
56     else
57         return queueNoError;
58 }

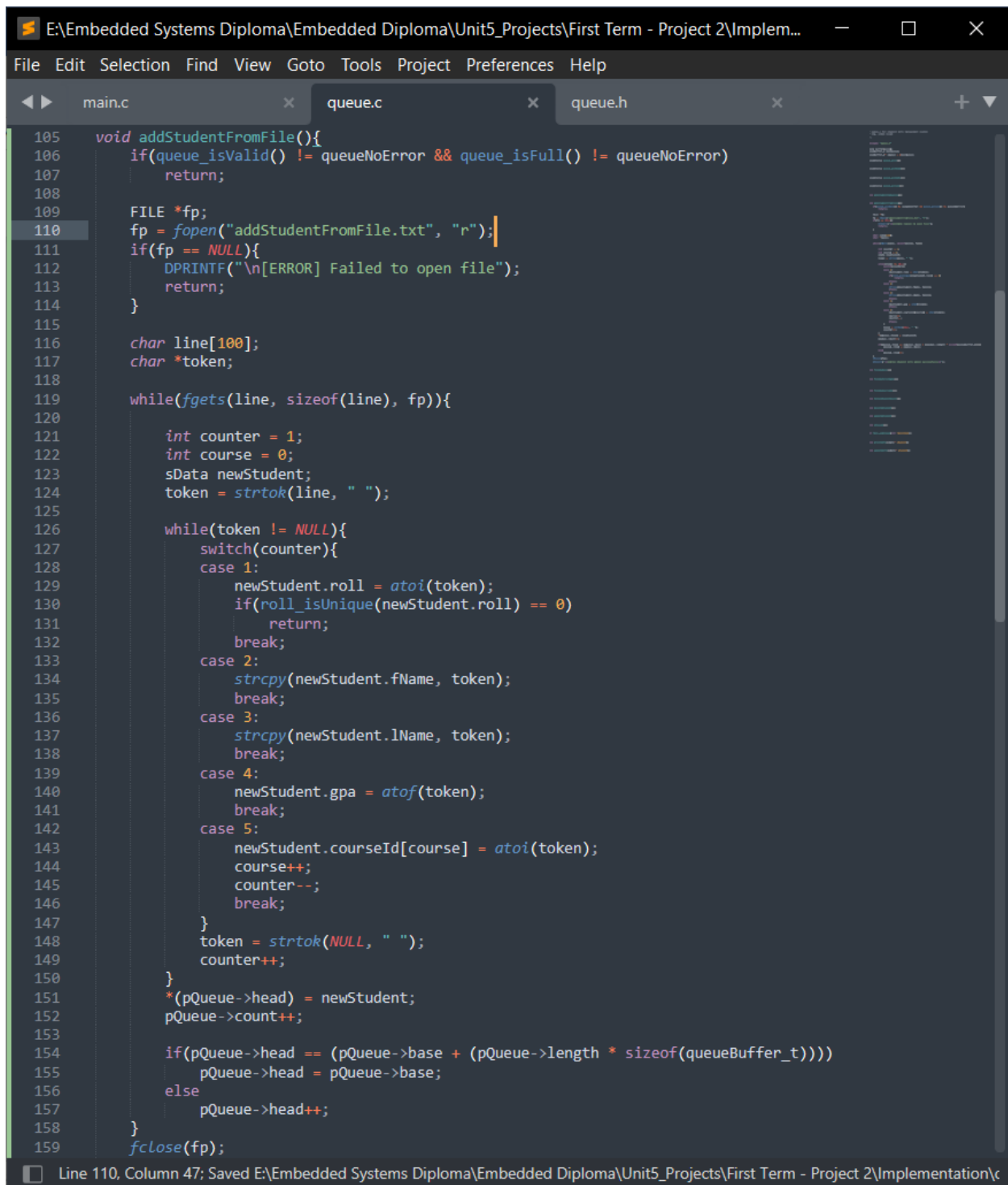
```

Figure 6 – isValid(), isEmpty(), isFull()



```
59
60 void addStudentManually(){
61
62     if(queue_isValid() != queueNoError && queue_isFull() != queueNoError)
63         return;
64
65     DPRINTF("\n-----");
66     DPRINTF("\nFill out Student Info");
67     DPRINTF("\n-----");
68
69     sData studentInfo;
70     char tempText[40];
71
72     DPRINTF("\nEnter Roll Number: ");
73     gets(tempText);
74     studentInfo.roll = atoi(tempText);
75     if(roll_isUnique(studentInfo.roll) == 0)
76         return;
77
78     DPRINTF("Enter First Name: ");
79     gets(studentInfo.fName);
80
81     DPRINTF("Enter Last Name: ");
82     gets(studentInfo.lName);
83
84     DPRINTF("Enter GPA: ");
85     gets(tempText);
86     studentInfo.gpa = atof(tempText);
87
88     DPRINTF("Enter each Course ID \n");
89     for(int i = 0; i < 5; i++){
90         DPRINTF(" Course Id ID: ", i+1);
91         gets(tempText);
92         studentInfo.courseId[i] = atoi(tempText);
93     }
94
95     *(pQueue->head) = studentInfo;
96     pQueue->count++;
97
98     if(pQueue->head == (pQueue->base + (pQueue->length * sizeof(queueBuffer_t))))
99         pQueue->head = pQueue->base;
100     else
101         pQueue->head++;
102     DPRINTF("\n[INFO] Student Info added successfully\n");
103 }
104
```

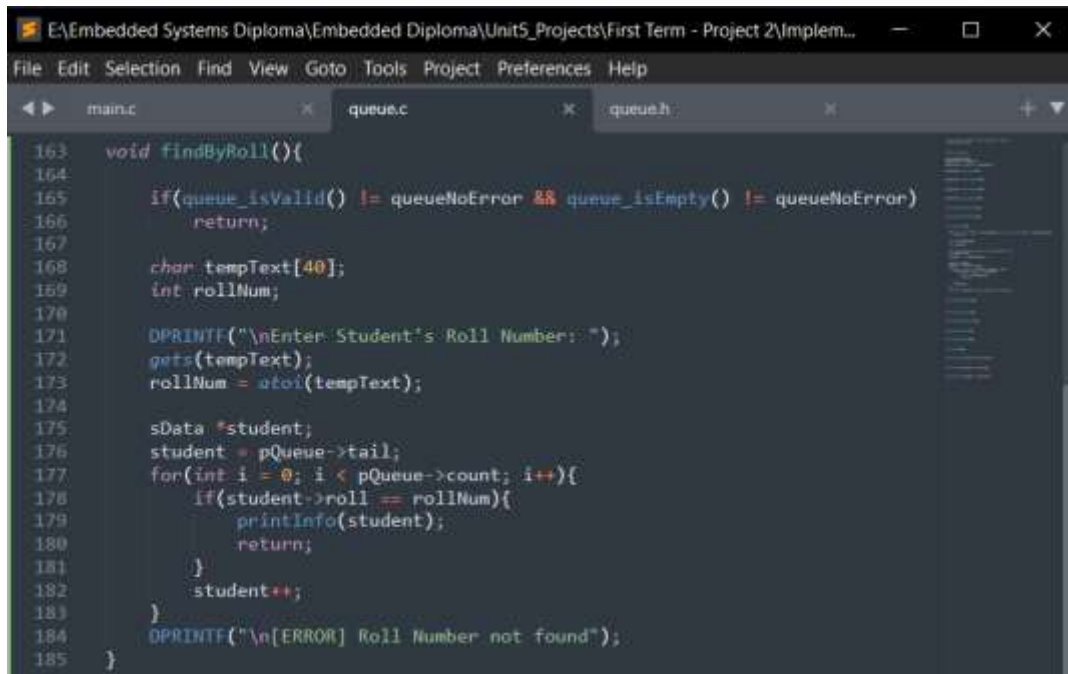
Figure 7 – addStudentManually()



```
105 void addStudentFromFile(){
106     if(queue_isValid() != queueNoError && queue_isFull() != queueNoError)
107         return;
108
109     FILE *fp;
110     fp = fopen("addStudentFromFile.txt", "r");
111     if(fp == NULL){
112         DPRINTF("\n[ERROR] Failed to open file");
113         return;
114     }
115
116     char line[100];
117     char *token;
118
119     while(fgets(line, sizeof(line), fp)){
120
121         int counter = 1;
122         int course = 0;
123         sData newStudent;
124         token = strtok(line, " ");
125
126         while(token != NULL){
127             switch(counter){
128                 case 1:
129                     newStudent.roll = atoi(token);
130                     if(roll_isUnique(newStudent.roll) == 0)
131                         return;
132                     break;
133                 case 2:
134                     strcpy(newStudent.fName, token);
135                     break;
136                 case 3:
137                     strcpy(newStudent.lName, token);
138                     break;
139                 case 4:
140                     newStudent.gpa = atof(token);
141                     break;
142                 case 5:
143                     newStudent.courseId[course] = atoi(token);
144                     course++;
145                     counter--;
146                     break;
147             }
148             token = strtok(NULL, " ");
149             counter++;
150         }
151         *(pQueue->head) = newStudent;
152         pQueue->count++;
153
154         if(pQueue->head == (pQueue->base + (pQueue->length * sizeof(queueBuffer_t))))
155             pQueue->head = pQueue->base;
156         else
157             pQueue->head++;
158     }
159     fclose(fp);
160 }
```

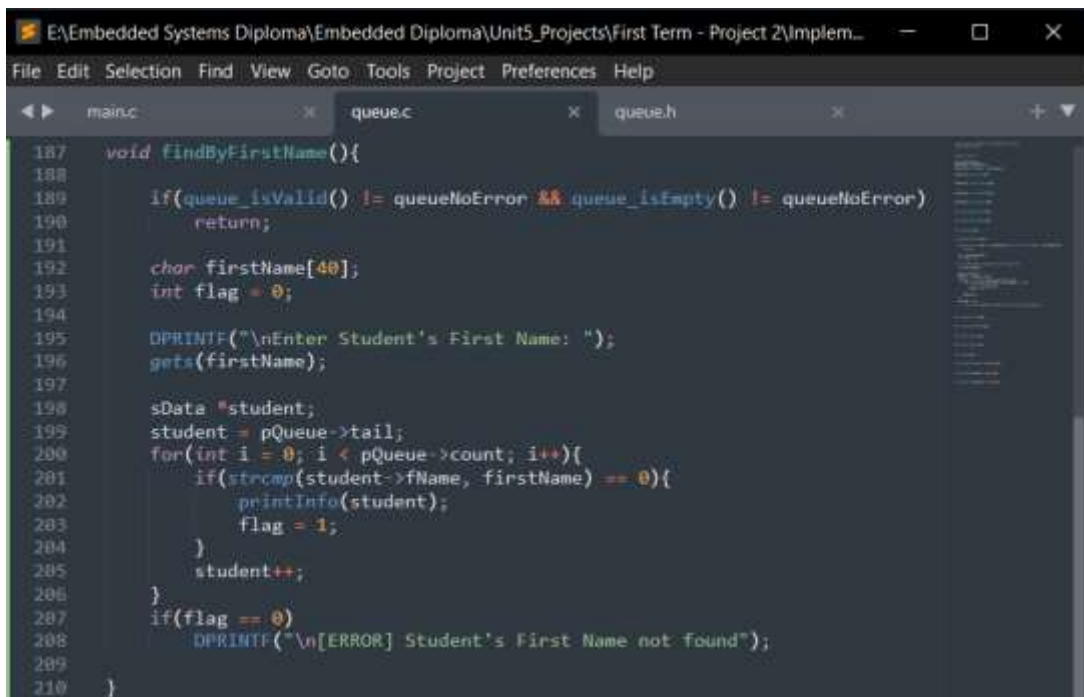
Line 110, Column 47; Saved E:\Embedded Systems Diploma\Embedded Diploma\Unit5_Projects\First Term - Project 2\Implementation\c

Figure 8 - addStudentFromFile()



```
163 void findByRoll(){
164
165     if(queue_isValid() != queueNoError && queue_isEmpty() != queueNoError)
166         return;
167
168     char tempText[40];
169     int rollNum;
170
171     DPRINTF("\nEnter Student's Roll Number: ");
172     gets(tempText);
173     rollNum = atoi(tempText);
174
175     sData *student;
176     student = pQueue->tail;
177     for(int i = 0; i < pQueue->count; i++){
178         if(student->roll == rollNum){
179             printInfo(student);
180             return;
181         }
182         student++;
183     }
184     DPRINTF("\n[ERROR] Roll Number not found");
185 }
```

Figure 9 - findByRoll()



```
187 void findByFirstName(){
188
189     if(queue_isValid() != queueNoError && queue_isEmpty() != queueNoError)
190         return;
191
192     char firstName[40];
193     int flag = 0;
194
195     DPRINTF("\nEnter Student's First Name: ");
196     gets(firstName);
197
198     sData *student;
199     student = pQueue->tail;
200     for(int i = 0; i < pQueue->count; i++){
201         if(strcmp(student->fName, firstName) == 0){
202             printInfo(student);
203             flag = 1;
204         }
205         student++;
206     }
207     if(flag == 0)
208         DPRINTF("\n[ERROR] Student's First Name not found");
209 }
210 }
```

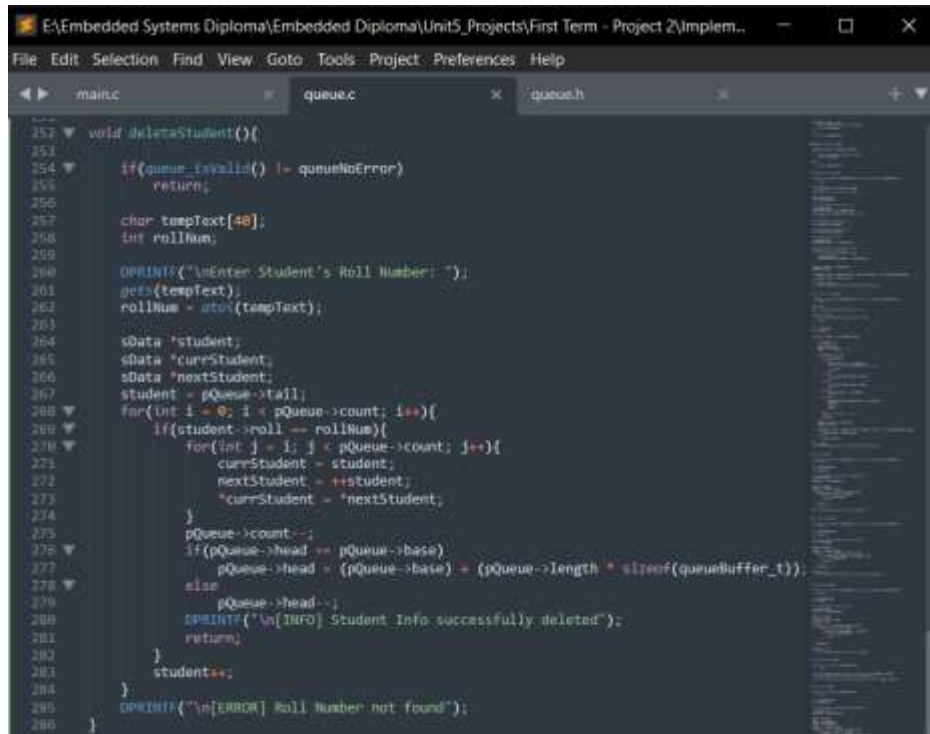
Figure 10 - findByFirstName()

```
212 void findByCourseId(){
213
214     if(queue_isValid() != queueNoError && queue_isEmpty() != queueNoError)
215         return;
216
217     char tempText[40];
218     int course;
219
220     DPRINTF("\nEnter Course ID: ");
221     gets(tempText);
222     course = atoi(tempText);
223
224     sData *student;
225     student = pQueue->tail;
226     int flag = 0;
227     for(int i = 0; i < pQueue->count; i++){
228         for(int j = 0; j < 5; j++){
229             if(student->courseId[j] == course){
230                 printInfo(student);
231                 flag = 1;
232             }
233         }
234         student++;
235     }
236     if(flag == 0)
237         DPRINTF("\n[ERROR] Course ID not found");
238 }
239
```

Figure 11 – findByCourseId()

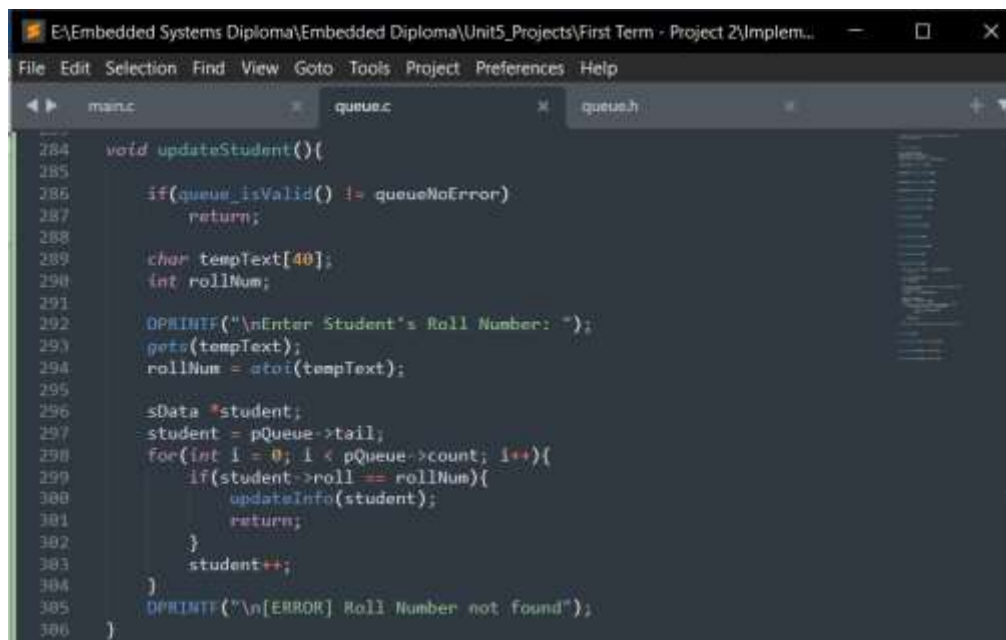
```
240 void totalStudentCount(){
241
242     if(queue_isValid() != queueNoError)
243         return;
244
245     DPRINTF("\n[INFO] Total Number of Students = %d", pQueue->count);
246     DPRINTF("\n[INFO] You can add up to 50 Students");
247     DPRINTF("\n[INFO] %d more slots are available", 50 - pQueue->count);
248 }
```

Figure 12 - totalStudentCount()



```
E:\Embedded Systems Diploma\Embedded Diploma\Unit5_Projects\First Term - Project 2\Implem...
File Edit Selection Find View Goto Tools Project Preferences Help
main.c queue.c queue.h
252 void deleteStudent(){
253
254     if(queue_isValid() != queueNoError)
255         return;
256
257     char tempText[40];
258     int rollNum;
259
260     DPRINTF("\nEnter Student's Roll Number: ");
261     gets(tempText);
262     rollNum = atoi(tempText);
263
264     sData *student;
265     sData *currStudent;
266     sData *nextStudent;
267     student = pQueue->tail;
268     for(int i = 0; i < pQueue->count; i++){
269         if(student->roll == rollNum){
270             for(int j = i; j < pQueue->count; j++){
271                 currStudent = student;
272                 nextStudent = ++student;
273                 *currStudent = *nextStudent;
274             }
275             pQueue->count--;
276             if(pQueue->head == pQueue->base)
277                 pQueue->head = (pQueue->base) + (pQueue->length * sizeof(queueBuffer_t));
278             else
279                 pQueue->head--;
280             DPRINTF("\n[INFO] Student Info successfully deleted");
281             return;
282         }
283         student++;
284     }
285     DPRINTF("\n[ERROR] Roll Number not found");
286 }
```

Figure 13 - deleteStudent()



```
E:\Embedded Systems Diploma\Embedded Diploma\Unit5_Projects\First Term - Project 2\Implem...
File Edit Selection Find View Goto Tools Project Preferences Help
main.c queue.c queue.h
284 void updateStudent(){
285
286     if(queue_isValid() != queueNoError)
287         return;
288
289     char tempText[40];
290     int rollNum;
291
292     DPRINTF("\nEnter Student's Roll Number: ");
293     gets(tempText);
294     rollNum = atoi(tempText);
295
296     sData *student;
297     student = pQueue->tail;
298     for(int i = 0; i < pQueue->count; i++){
299         if(student->roll == rollNum){
300             updateInfo(student);
301             return;
302         }
303         student++;
304     }
305     DPRINTF("\n[ERROR] Roll Number not found");
306 }
```

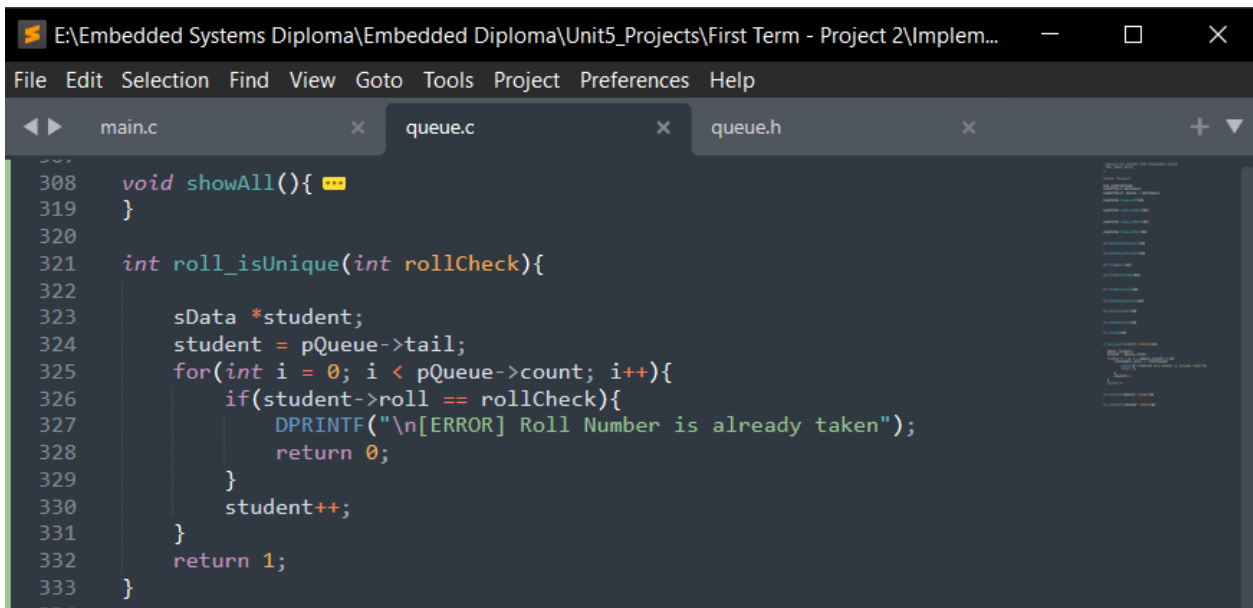
Figure 13 - updateStudent()

```

308 void showAll(){
309
310     if(queue_isValid() != queueNoError)
311         return;
312
313     sData *student;
314     student = pQueue->tail;
315     for(int i = 0; i < pQueue->count; i++){
316         printInfo(student);
317         student++;
318     }
319 }

```

Figure 14 - showAll()



```

308 void showAll(){
309 }
310
320
321 int roll_isUnique(int rollCheck){
322
323     sData *student;
324     student = pQueue->tail;
325     for(int i = 0; i < pQueue->count; i++){
326         if(student->roll == rollCheck){
327             DPRINTF("\n[ERROR] Roll Number is already taken");
328             return 0;
329         }
330         student++;
331     }
332     return 1;
333 }

```

Figure 15 - roll_isUnique()

```

300 void showAll() {
301 }
302
303 int roll_isUnique(int rollCheck) {
304 }
305
306 void printInfo(sData* student) {
307     printf("\nRoll Number: %d", student->roll);
308     printf("\nFirst Name: %s", student->fName);
309     printf("\nLast Name: %s", student->lName);
310     printf("\nGPA: %.2f", student->gpa);
311     printf("\nCourses ID: ");
312     for(int i = 0; i < 5; i++) {
313         printf("%d ", student->courseId[i]);
314     }
315     printf("\n-----");
316 }

```

Figure 16 - printInfo()

```

347 void updateInfo(sData* student) {
348     char tempText[40];
349     printf("\nChoose data to be updated: ");
350     printf("\n 1: Roll Number");
351     printf("\n 2: First Name");
352     printf("\n 3: Last Name");
353     printf("\n 4: GPA");
354     printf("\n 5: Courses ID");
355     printf("\nEnter an option: ");
356     gets(tempText);
357
358     switch(atoi(tempText)) {
359     case 1:
360         printf("\nEnter Roll Number: ");
361         gets(tempText);
362         student->roll = atoi(tempText);
363         if(!roll_isUnique(student->roll))
364             return;
365         else
366             printf("\n[INFO] Data updated successfully");
367         break;
368
369     case 2:
370         printf("\nEnter First Name: ");
371         gets(student->fName);
372         printf("\n[INFO] Data updated successfully");
373         break;
374
375     case 3:
376         printf("\nEnter Last Name: ");
377         gets(student->lName);
378         printf("\n[INFO] Data updated successfully");
379         break;
380
381     case 4:
382         printf("\nEnter GPA: ");
383         gets(tempText);
384         student->gpa = atof(tempText);
385         printf("\n[INFO] Data updated successfully");
386         break;
387
388     case 5:
389         printf("\nEnter each Course ID \n");
390         for(int i = 0; i < 5; i++) {
391             printf(" Course %d ID: ", i+1);
392             gets(tempText);
393             student->courseId[i] = atoi(tempText);
394         }
395         printf("\n[INFO] Data updated successfully");
396         break;
397     }
398 }
399
400 }

```

Figure 17 - updateInfo()

4 OUTPUT

4.1 addStudentManually()

```
*****Student Management System*****
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 1

=====
-----
Fill out Student Info
-----
Enter Roll Number: 5
Enter First Name: Ahmed
Enter Last Name: ElShafie
Enter GPA: 3.5
Enter each Course ID
Course 1 ID: 6
Course 2 ID: 7
Course 3 ID: 8
Course 4 ID: 9
Course 5 ID: 1
[
[INFO] Student Info added successfully
```

Figure 18 - Output addStudentManually()

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 9

=====
Roll Number: 5
First Name: Ahmed
Last Name: ElShafie
GPA: 3.50
Courses ID: 6 7 8 9 1
-----
=====
```

Figure 19 - showAll() after adding.

4.2 addStudentFromFile()

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 2

=====
[INFO] Student Info added successfully
```

Figure 20 - Output addStudentFromFile()

```
=====
Roll Number: 5
First Name: Ahmed
Last Name: ElShafie
GPA: 3.50
Courses ID: 6 7 8 9 1
-----
Roll Number: 1
First Name: Ahmed
Last Name: Essam
GPA: 3.00
Courses ID: 1 2 3 4 5
-----
Roll Number: 2
First Name: Hagar
Last Name: Karkar
GPA: 2.50
Courses ID: 11 22 3 44 55
-----
Roll Number: 3
First Name: Reem
Last Name: Sadek
GPA: 3.30
Courses ID: 11 66 77 88 99
-----
Roll Number: 4
First Name: Zead
Last Name: Hani
GPA: 2.80
Courses ID: 1 22 77 3 99
-----
```

Figure 21 - showAll() after adding.

4.3 findByRoll()

```
Enter an Option: 3

=====
Enter Student's Roll Number: 1

    Roll Number: 1
    First Name: Ahmed
    Last Name: Essam
    GPA: 3.00
    Courses ID: 1 2 3 4 5
-----

=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 3

=====
Enter Student's Roll Number: 2
[
    Roll Number: 2
    First Name: Hagar
    Last Name: Karkar
    GPA: 2.50
    Courses ID: 11 22 3 44 55
-----
```

Figure 22 - Output findByRoll()

4.4 findByFirstName()

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 4

=====
Enter Student's First Name: Ahmed
[
    Roll Number: 5
    First Name: Ahmed
    Last Name: ElShafie
    GPA: 3.50
    Courses ID: 6 7 8 9 1
-----
    Roll Number: 1
    First Name: Ahmed
    Last Name: Essam
    GPA: 3.00
    Courses ID: 1 2 3 4 5
-----
```

Figure 23 - Output findByFirstName()

4.5 findByCourseId()

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 5

=====
Enter Course ID: 1
[
    Roll Number: 5
    First Name: Ahmed
    Last Name: ElShafie
    GPA: 3.50
    Courses ID: 6 7 8 9 1
-----
    Roll Number: 1
    First Name: Ahmed
    Last Name: Essam
    GPA: 3.00
    Courses ID: 1 2 3 4 5
-----
    Roll Number: 4
    First Name: Zead
    Last Name: Hani
    GPA: 2.80
    Courses ID: 1 22 77 3 99
-----
```

Figure 24 - Output findByCourseId()

4.6 totalStudentCount()

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 6
[
=====
[INFO] Total Number of Students = 5
[INFO] You can add up to 50 Students
[INFO] 45 more slots are available
=====
```

Figure 25 - Output totalStudentCount()

4.7 deleteStudent()

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 7

=====
Enter Student's Roll Number: 5

[INFO] Student Info successfully deleted
=====
```

Figure 26 - Output deleteStudent()

```
Enter an Option: 9
{
=====
    Roll Number: 1
    First Name: Ahmed
    Last Name: Essam
    GPA: 3.00
    Courses ID: 1 2 3 4 5
-----
    Roll Number: 2
    First Name: Hagar
    Last Name: Karkar
    GPA: 2.50
    Courses ID: 11 22 3 44 55
-----
    Roll Number: 3
    First Name: Reem
    Last Name: Sadek
    GPA: 3.30
    Courses ID: 11 66 77 88 99
-----
    Roll Number: 4
    First Name: Zead
    Last Name: Hani
    GPA: 2.80
    Courses ID: 1 22 77 3 99
-----
=====
```

Figure 27 - showAll() after deleting

4.8 updateStudent()

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 8

=====
Enter Student's Roll Number: 1

Choose data to be updated:
1: Roll Number
2: First Name
3: Last Name
4: GPA
5: Courses ID

Enter an option: 4

Enter GPA: 3.7

[INFO] Data updated successfully
=====
```

Figure 28 - Output updateStudent()

```
Enter an Option: 9
[
=====
Roll Number: 1
First Name: Ahmed
Last Name: Essam
GPA: 3.70
Courses ID: 1 2 3 4 5
-----
Roll Number: 2
First Name: Hagar
Last Name: Karkar
GPA: 2.50
Courses ID: 11 22 3 44 55
-----
Roll Number: 3
First Name: Reem
Last Name: Sadek
GPA: 3.30
Courses ID: 11 66 77 88 99
-----
Roll Number: 4
First Name: Zead
Last Name: Hani
GPA: 2.80
Courses ID: 1 22 77 3 99
-----
]
```

Figure 29 - showAll() after updating

4.9 showAll()

```
Enter an Option: 9
[
=====
    Roll Number: 1
    First Name: Ahmed
    Last Name: Essam
    GPA: 3.70
    Courses ID: 1 2 3 4 5
-----
    Roll Number: 2
    First Name: Hagar
    Last Name: Karkar
    GPA: 2.50
    Courses ID: 11 22 3 44 55
-----
    Roll Number: 3
    First Name: Reem
    Last Name: Sadek
    GPA: 3.30
    Courses ID: 11 66 77 88 99
-----
    Roll Number: 4
    First Name: Zead
    Last Name: Hani
    GPA: 2.80
    Courses ID: 1 22 77 3 99
-----
=====
```

Figure 30 - Output showAll()

4.10 Other Features

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 1

=====
-----
Fill out Student Info
-----
Enter Roll Number: 1
[
[ERROR] Roll Number is already taken
=====
```

Figure 30 - Adding duplicate Roll Number

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 3

=====
Enter Student's Roll Number: 7

[ERROR] Roll Number not found
=====
```

Figure 31 - Roll Number not found

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 4

=====
Enter Student's First Name: Omar

[ERROR] Student's First Name not found
=====
```

Figure 32 - First Name not found

```
=====
List of Available Options:

1: Add New Student Manually
2: Add New Student/s From Text File
3: Find Student by Roll Number
4: Find Student/s by First Name
5: Find Student/s by Course ID
6: Total Number of Students
7: Delete Student by Roll Number
8: Update Student by Roll Number
9: View All Students Info
10: Exit

Enter an Option: 5

=====
Enter Course ID: 17

[ERROR] Course ID not found
=====
```

Figure 33 - Course ID not found