Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project 2)

Eng. Habeba Ahmed Elbaghdady

Table of Contents

Problem Statement

A Simple Software for Student Management System Which Can perform the following operations:

a. Store the First Name of the Student.
b. Store the Last Name of the Student.
c. Store the Unique Roll Number for every Student.
d. Store the GPA of each Student.
e. Store the Courses registered by the student.

Approach:

The idea is to form an individual functions for every operation.

All the functions are unified to form the software:

- Add Student Details from File.
- Add student Details Manually.
- Find the Student by the given Roll Number
- Find the Student by the given First Name.
- Find the Student by the given Last Name.
- Find the Student registered in a course.
- Count of Students.
- Delete a Student.
- Update Student.
- Checking Roll Number in the Data
- Exit

Idea:

The software consists of 5 Files Main.c Queue.h Queue.c App.c App.h.

The main contains the calling functions.

The Queue.h contains the declarations of queue functions.

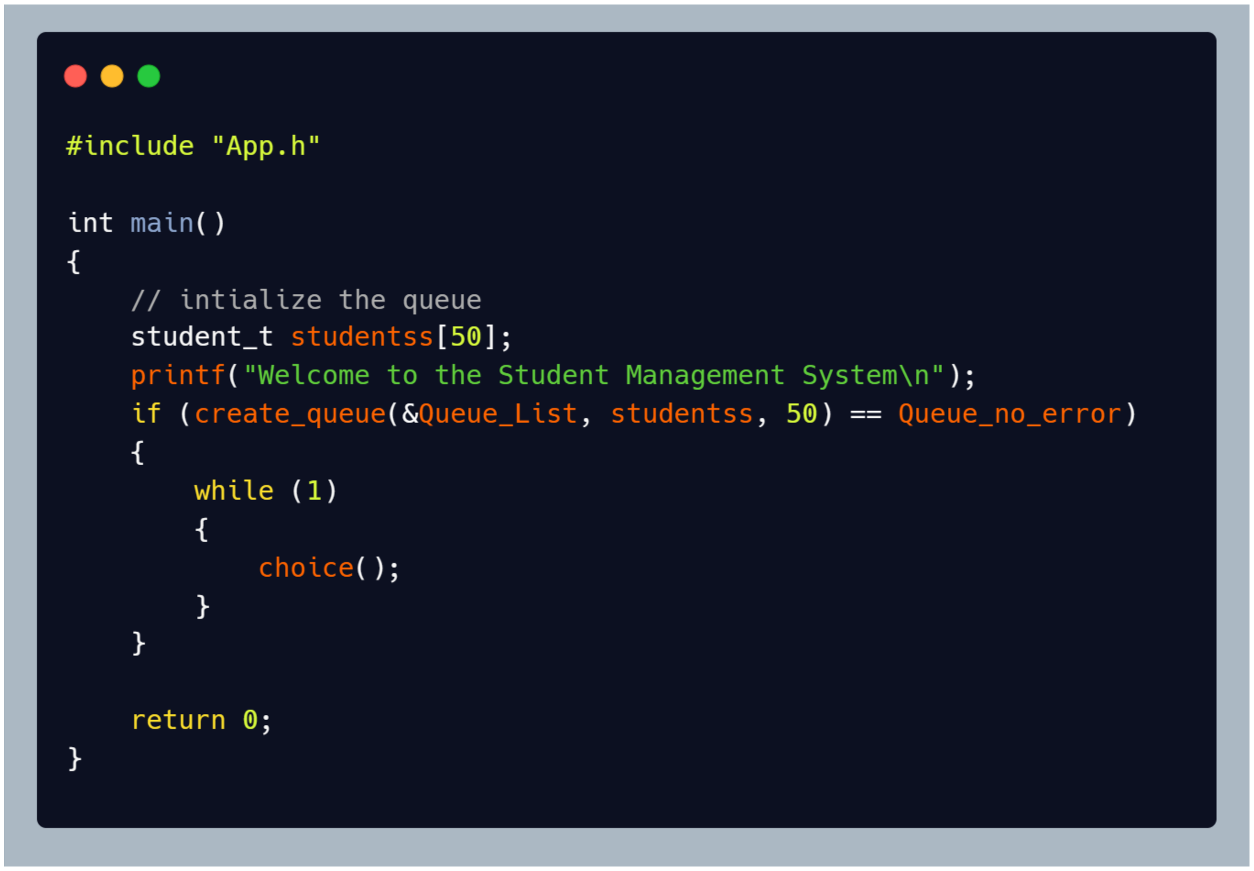
The Queue.c contains the implementations of queue functions.

The App.h contains the declarations of Student Management System functions.

The App.c contains the implementations of Student Management System functions.

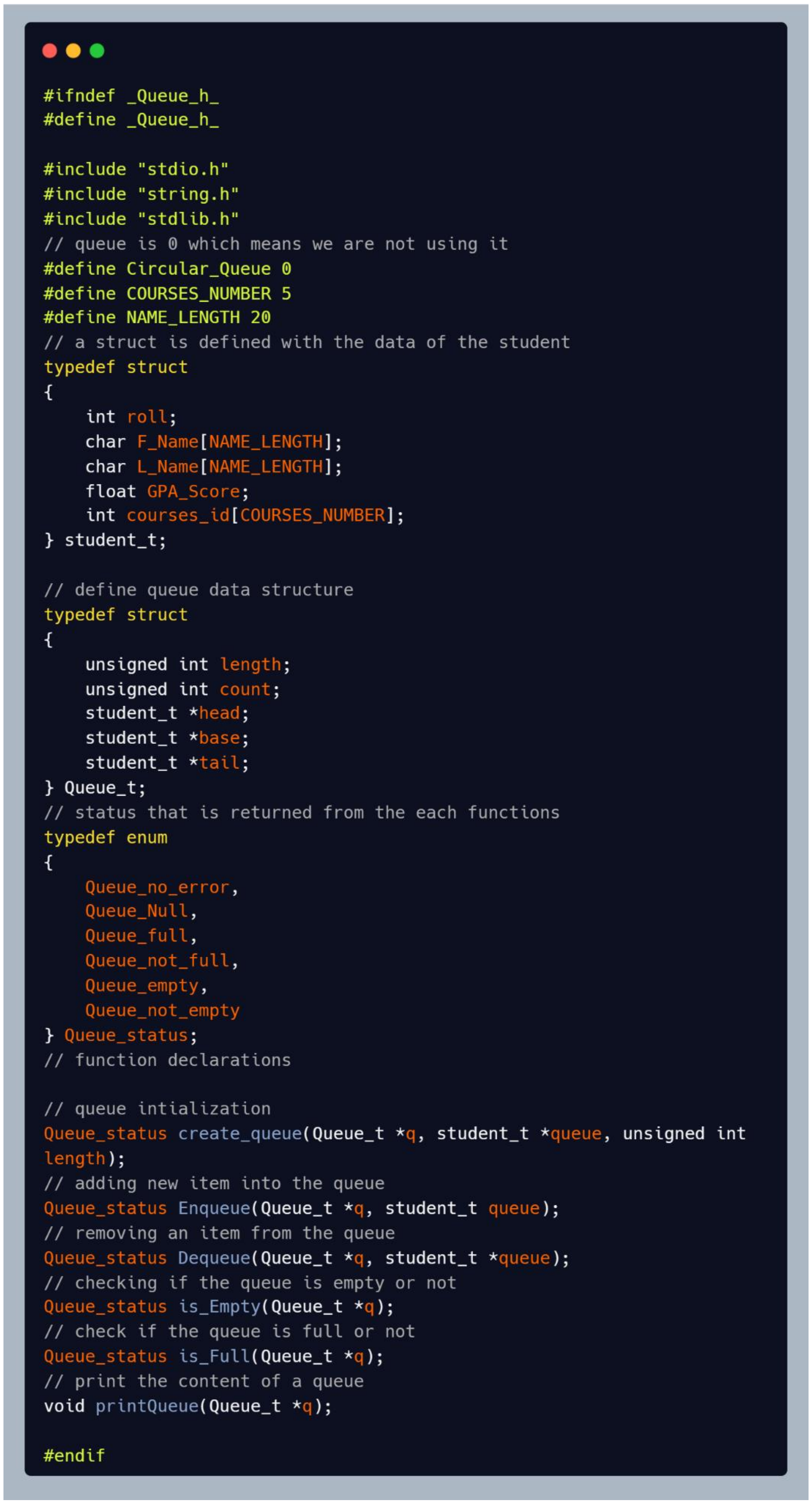
Main.c

- The queue is initialized using create_queue function.
- A welcoming message is shown to the user.
- The choice function is called infinitely.


```
#include "App.h"

int main()
{
    // intialize the queue
    student_t studentss[50];
    printf("Welcome to the Student Management System\n");
    if (create_queue(&Queue_List, studentss, 50) == Queue_no_error)
    {
        while (1)
        {
            choice();
        }
    }

    return 0;
}
```

Queue.h

- It includes the libraries needed for the program to run.
- A struct of type student holds the variables which will have the data of each student in the system.
- A struct of type queue.
- An enum of type status that includes the status of each operation done in the queue.
- Functions declarations.

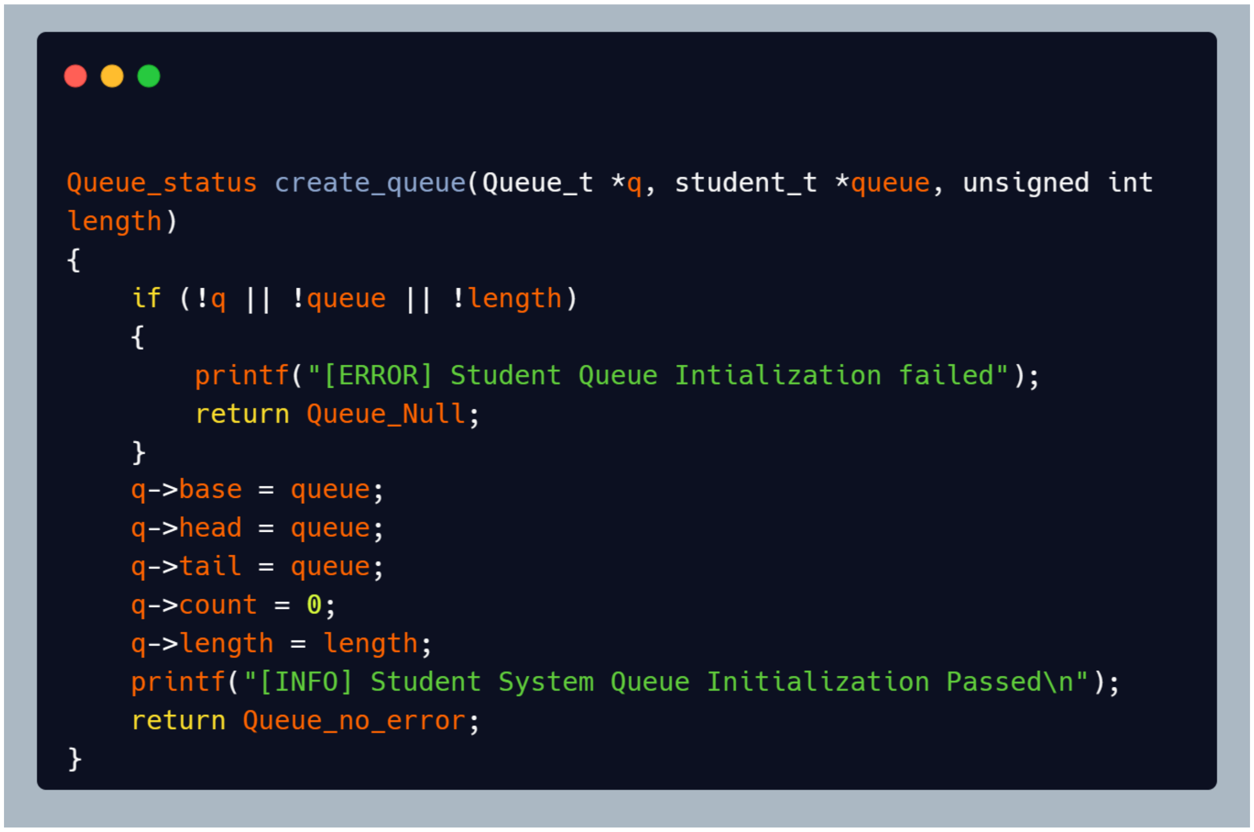


```c
#ifndef _Queue_h_
#define _Queue_h_

#include "stdio.h"
#include "string.h"
#include "stdlib.h"
// queue is 0 which means we are not using it
#define Circular_Queue 0
#define COURSES_NUMBER 5
#define NAME_LENGTH 20
// a struct is defined with the data of the student
typedef struct
{
    int roll;
    char F_Name[NAME_LENGTH];
    char L_Name[NAME_LENGTH];
    float GPA_Score;
    int courses_id[COURSES_NUMBER];
} student_t;

// define queue data structure
typedef struct
{
    unsigned int length;
    unsigned int count;
    student_t *head;
    student_t *base;
    student_t *tail;
} Queue_t;
// status that is returned from the each functions
typedef enum
{
    Queue_no_error,
    Queue_Null,
    Queue_full,
    Queue_not_full,
    Queue_empty,
    Queue_not_empty
} Queue_status;
// function declarations

// queue intialization
Queue_status create_queue(Queue_t *q, student_t *queue, unsigned int
length);
// adding new item into the queue
Queue_status Enqueue(Queue_t *q, student_t queue);
// removing an item from the queue
Queue_status Dequeue(Queue_t *q, student_t *queue);
// checking if the queue is empty or not
Queue_status is_Empty(Queue_t *q);
// check if the queue is full or not
Queue_status is_Full(Queue_t *q);
// print the content of a queue
void printQueue(Queue_t *q);

#endif
```

Queue.c

- ### create_queue

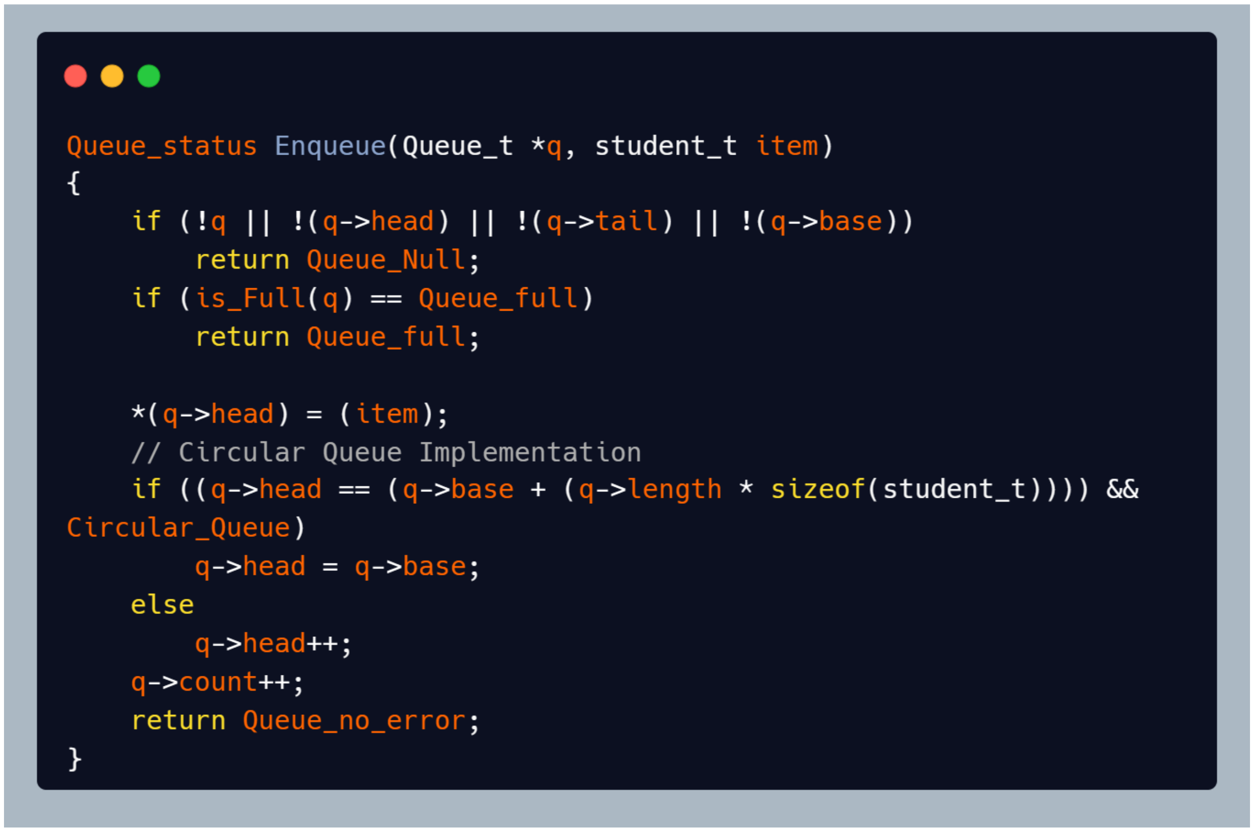
The function is used to initialize the queue.



```c
Queue_status create_queue(Queue_t *q, student_t *queue, unsigned int
length)
{
    if (!q || !queue || !length)
    {
        printf("[ERROR] Student Queue Intialization failed");
        return Queue_Null;
    }
    q->base = queue;
    q->head = queue;
    q->tail = queue;
    q->count = 0;
    q->length = length;
    printf("[INFO] Student System Queue Initialization Passed\n");
    return Queue_no_error;
}
```

- Enqueue

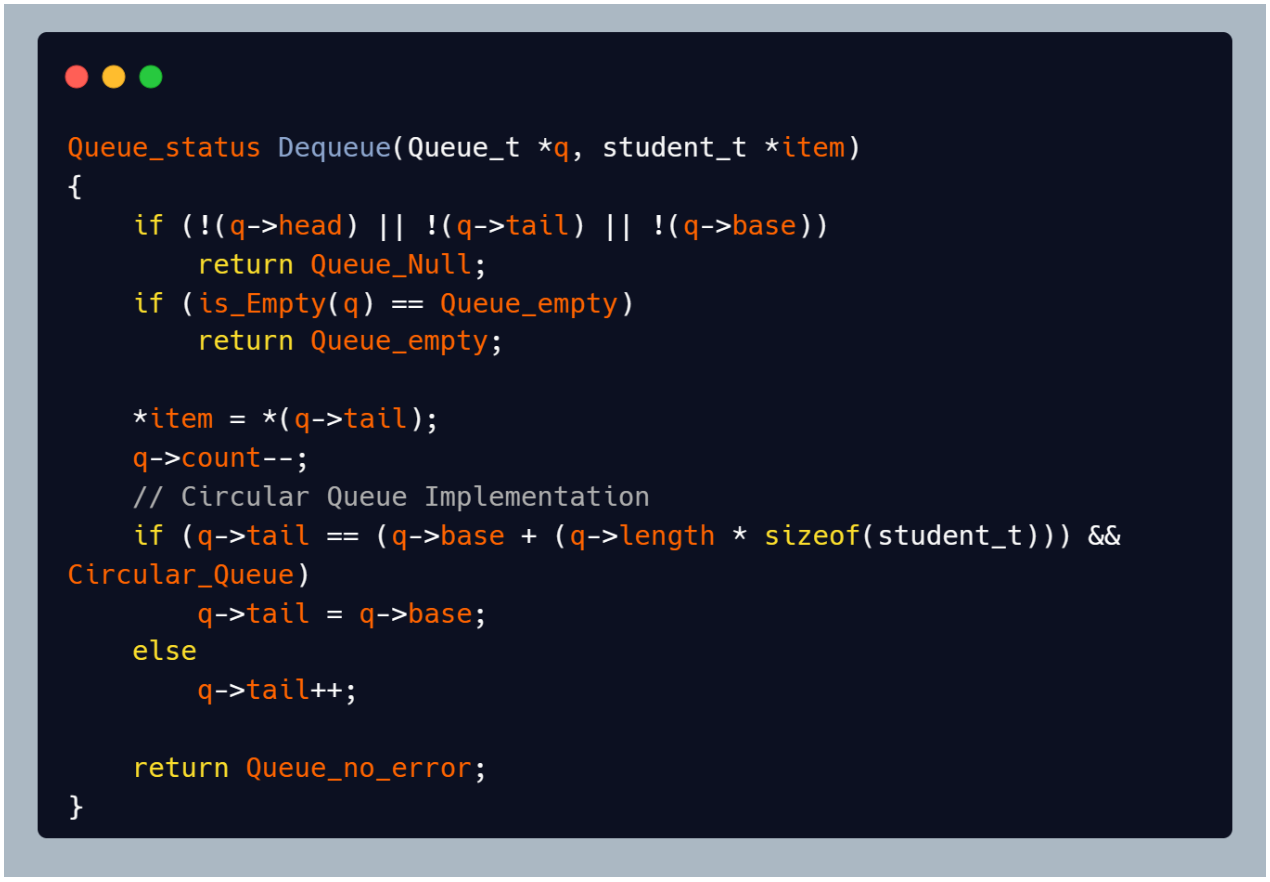
The function is used to add a value in the queue.



```c
Queue_status Enqueue(Queue_t *q, student_t item)
{
    if (!q || !(q->head) || !(q->tail) || !(q->base))
        return Queue_Null;
    if (is_Full(q) == Queue_full)
        return Queue_full;

    *(q->head) = (item);
    // Circular Queue Implementation
    if ((q->head == (q->base + (q->length * sizeof(student_t)))) &&
Circular_Queue)
        q->head = q->base;
    else
        q->head++;
    q->count++;
    return Queue_no_error;
}
```

- Dequeue

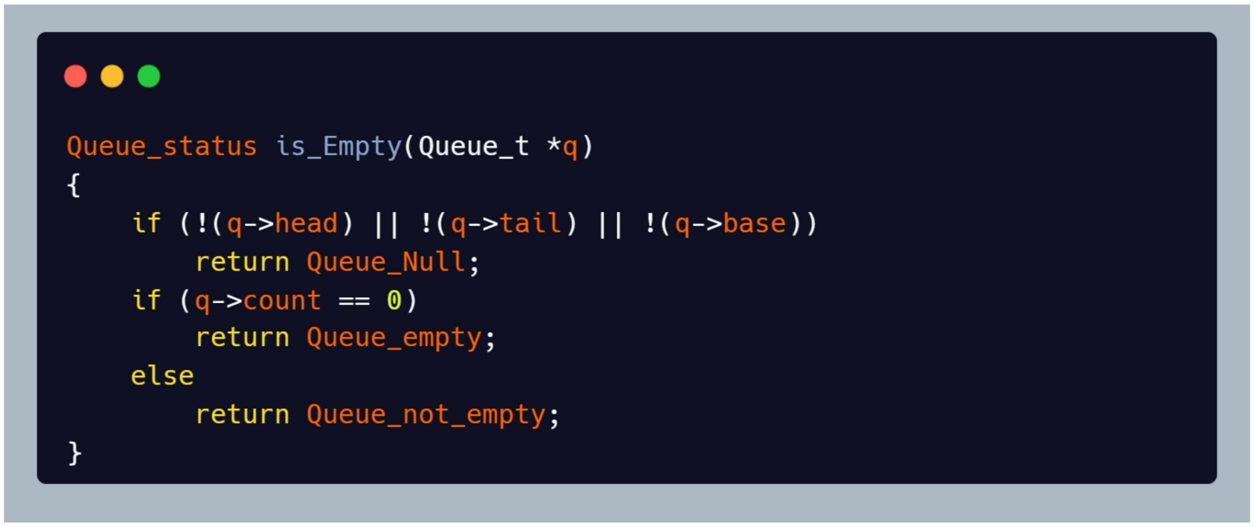
The function is used to remove the value from the queue.



```
Queue_status Dequeue(Queue_t *q, student_t *item)
{
    if (!(q->head) || !(q->tail) || !(q->base))
        return Queue_Null;
    if (is_Empty(q) == Queue_empty)
        return Queue_empty;

    *item = *(q->tail);
    q->count--;
    // Circular Queue Implementation
    if (q->tail == (q->base + (q->length * sizeof(student_t))) &&
Circular_Queue)
        q->tail = q->base;
    else
        q->tail++;

    return Queue_no_error;
}
```

- is_Empty

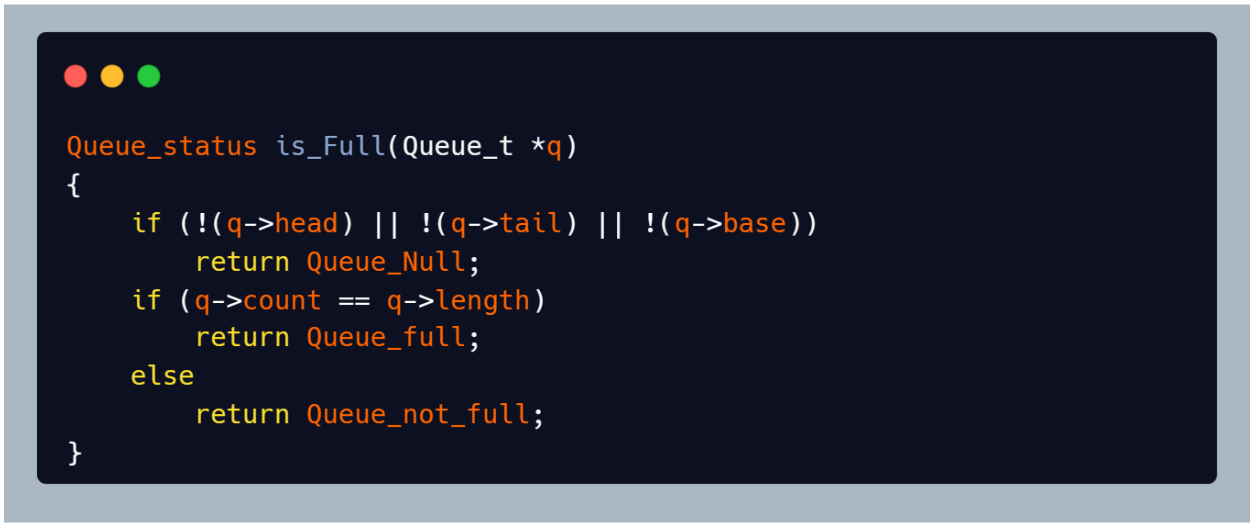
The function is used to check if the queue is empty or not.



```c
Queue_status is_Empty(Queue_t *q)
{
    if (!(q->head) || !(q->tail) || !(q->base))
        return Queue_Null;
    if (q->count == 0)
        return Queue_empty;
    else
        return Queue_not_empty;
}
```

- is_Full

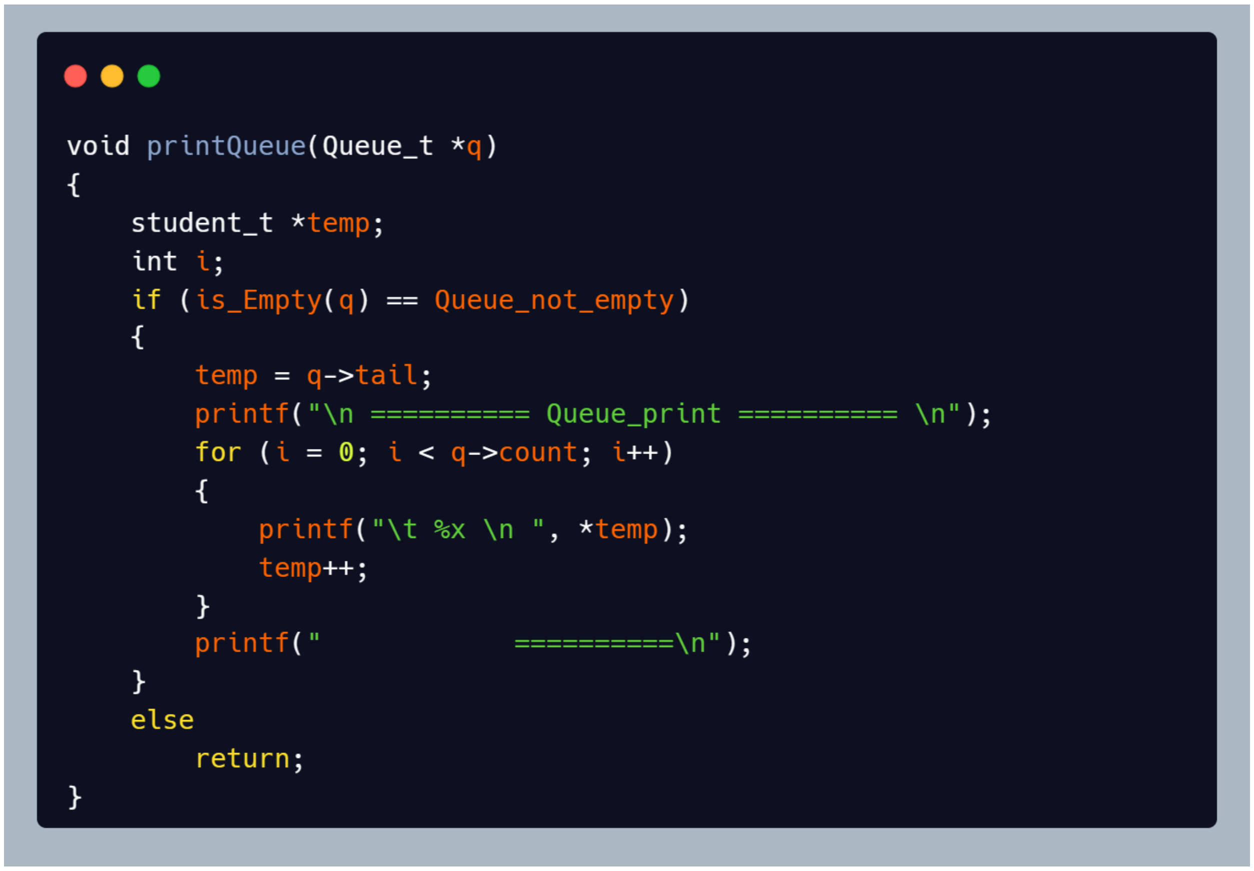
The function is used to check if the queue is full or not.



```c
Queue_status is_Full(Queue_t *q)
{
    if (!(q->head) || !(q->tail) || !(q->base))
        return Queue_Null;
    if (q->count == q->length)
        return Queue_full;
    else
        return Queue_not_full;
}
```

- printQueue

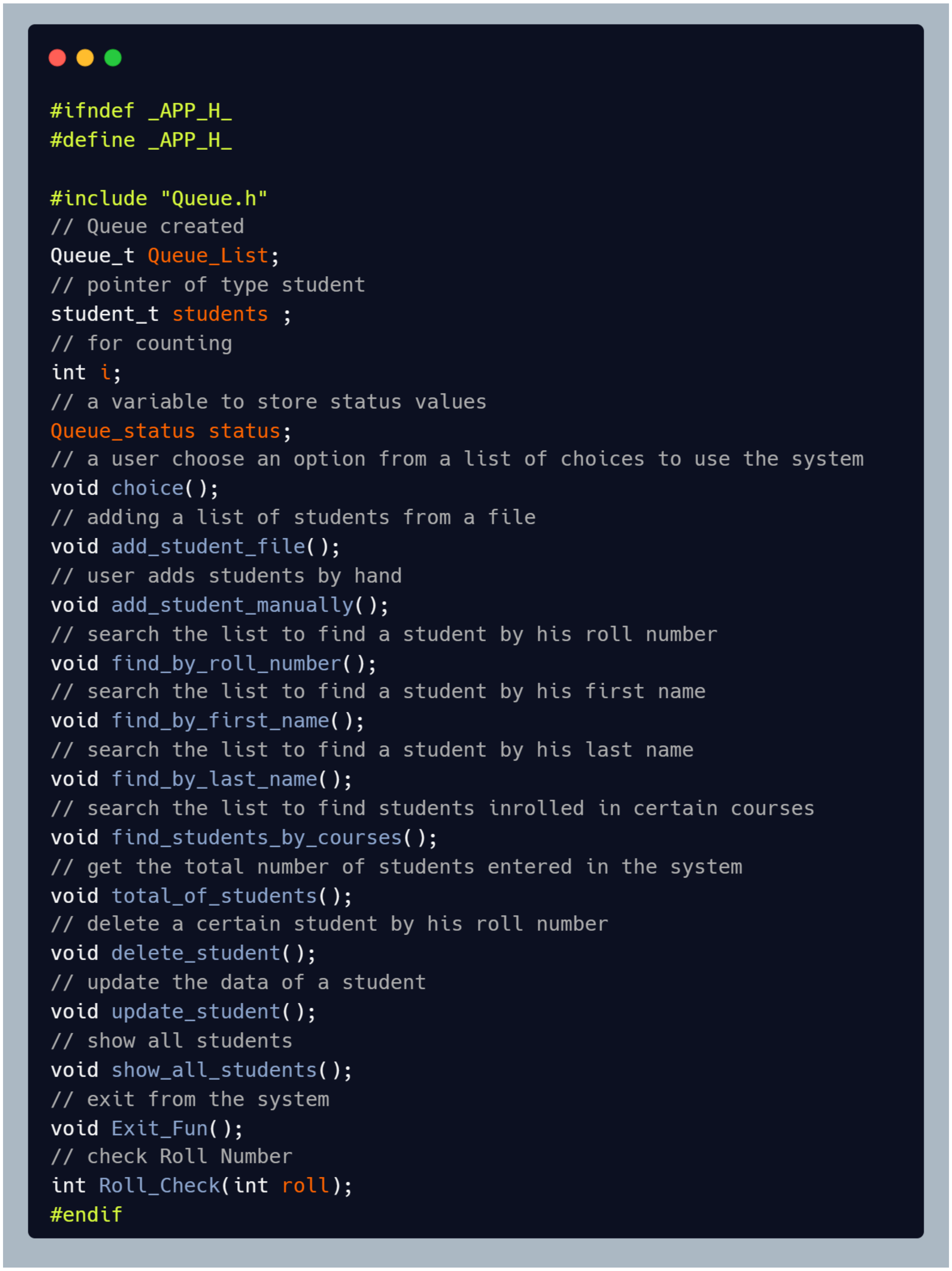
The function is used to print the values of the Queue.



```c
void printQueue(Queue_t *q)
{
    student_t *temp;
    int i;
    if (is_Empty(q) == Queue_not_empty)
    {
        temp = q->tail;
        printf("\n ========== Queue_print ========== \n");
        for (i = 0; i < q->count; i++)
        {
            printf("\t %x \n ", *temp);
            temp++;
        }
        printf("            ==========\n");
    }
    else
        return;
}
```

App.h

- It includes the Queue.h File.
- It has the creation of Queue_List and student.
- Functions Declarations.

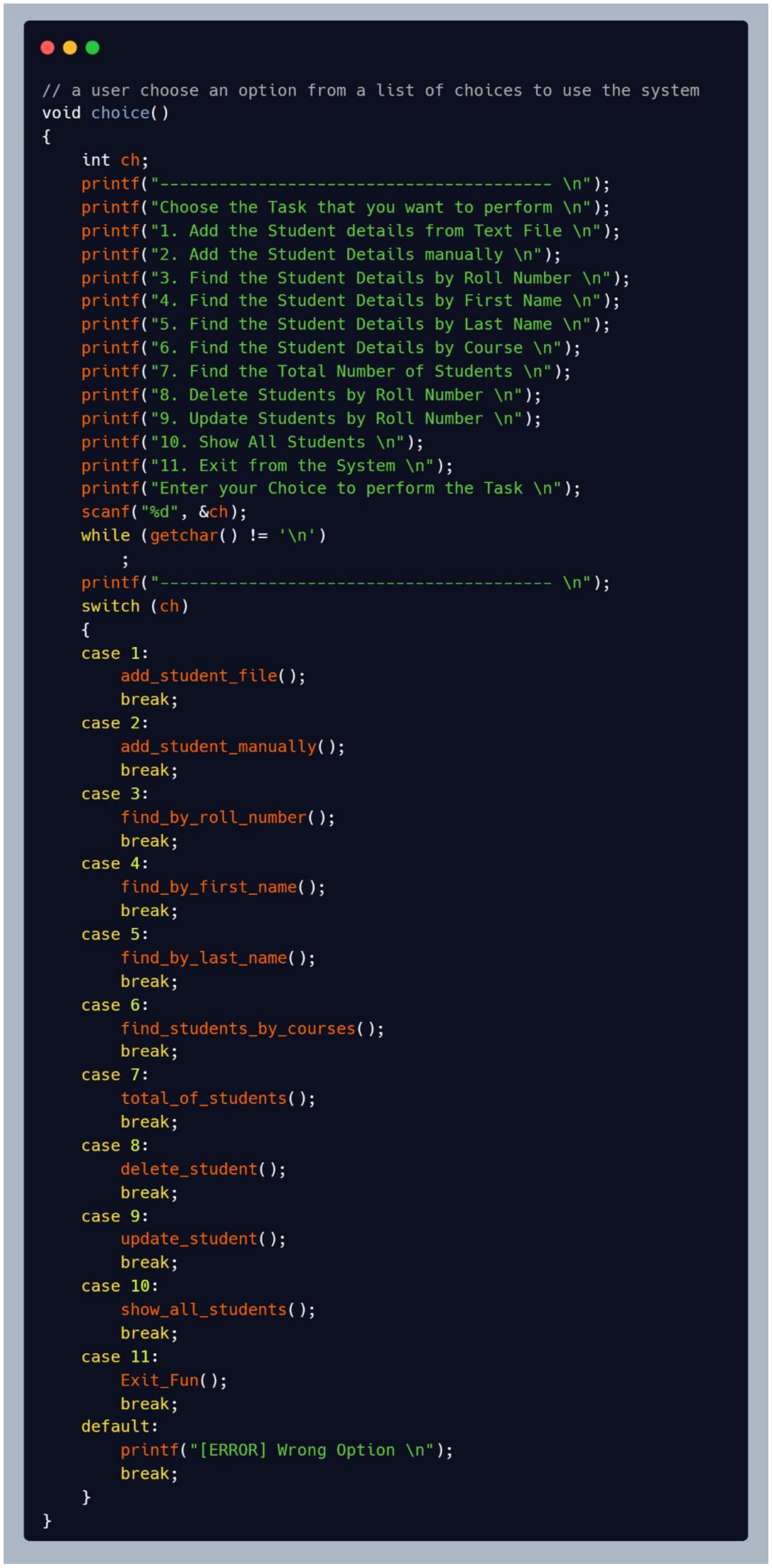


```
#ifndef _APP_H_
#define _APP_H_

#include "Queue.h"
// Queue created
Queue_t Queue_List;
// pointer of type student
student_t students ;
// for counting
int i;
// a variable to store status values
Queue_status status;
// a user choose an option from a list of choices to use the system
void choice();
// adding a list of students from a file
void add_student_file();
// user adds students by hand
void add_student_manually();
// search the list to find a student by his roll number
void find_by_roll_number();
// search the list to find a student by his first name
void find_by_first_name();
// search the list to find a student by his last name
void find_by_last_name();
// search the list to find students inrolled in certain courses
void find_students_by_courses();
// get the total number of students entered in the system
void total_of_students();
// delete a certain student by his roll number
void delete_student();
// update the data of a student
void update_student();
// show all students
void show_all_students();
// exit from the system
void Exit_Fun();
// check Roll Number
int Roll_Check(int roll);
#endif
```

App.c

- Choice

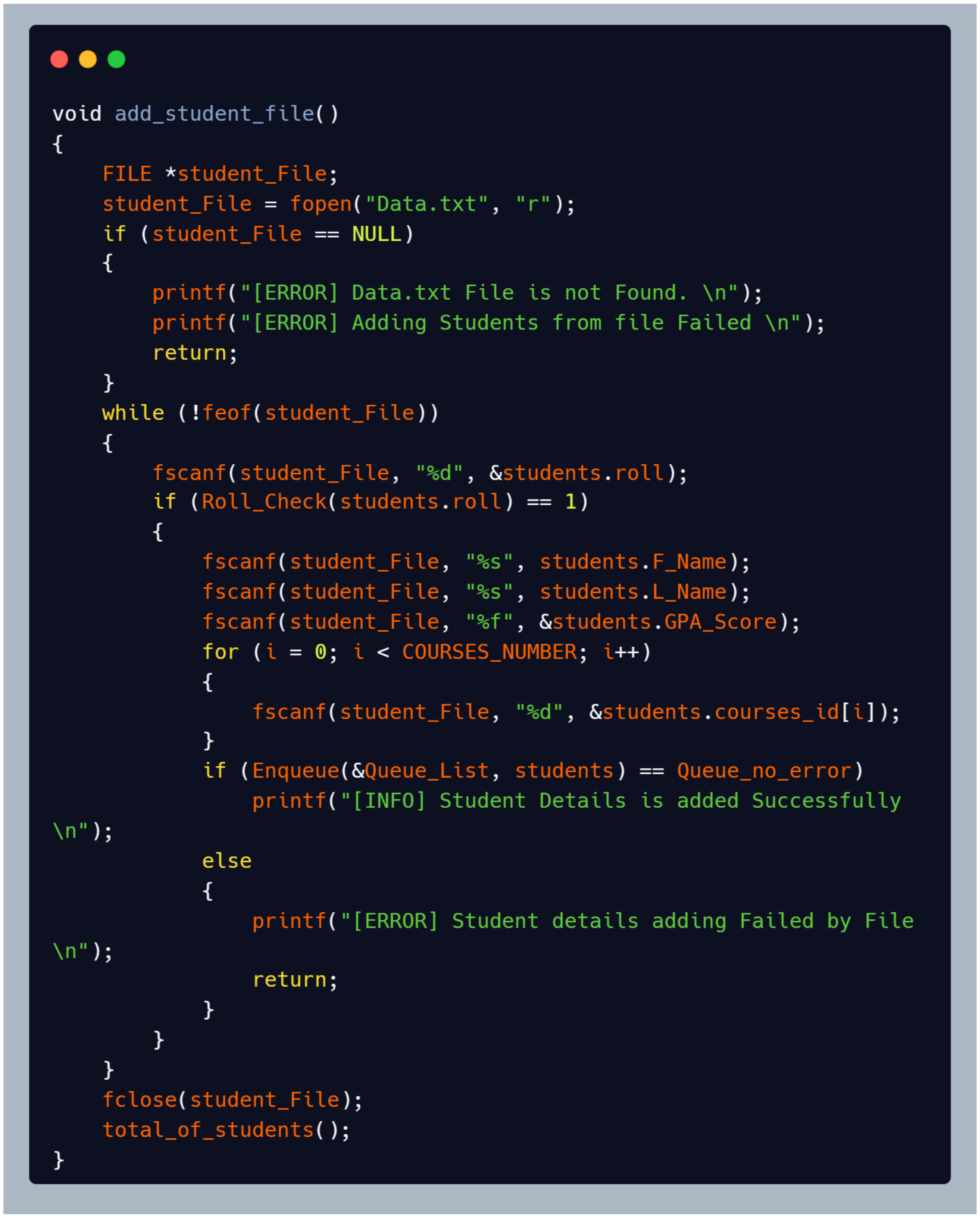
The Function is used to show the user all the functions in the system and asks him which function he wants to use.



```c
// a user choose an option from a list of choices to use the system
void choice()
{
    int ch;
    printf("---------------------------------------- \n");
    printf("Choose the Task that you want to perform \n");
    printf("1. Add the Student details from Text File \n");
    printf("2. Add the Student Details manually \n");
    printf("3. Find the Student Details by Roll Number \n");
    printf("4. Find the Student Details by First Name \n");
    printf("5. Find the Student Details by Last Name \n");
    printf("6. Find the Student Details by Course \n");
    printf("7. Find the Total Number of Students \n");
    printf("8. Delete Students by Roll Number \n");
    printf("9. Update Students by Roll Number \n");
    printf("10. Show All Students \n");
    printf("11. Exit from the System \n");
    printf("Enter your Choice to perform the Task \n");
    scanf("%d", &ch);
    while (getchar() != '\n')
        ;
    printf("---------------------------------------- \n");
    switch (ch)
    {
    case 1:
        add_student_file();
        break;
    case 2:
        add_student_manually();
        break;
    case 3:
        find_by_roll_number();
        break;
    case 4:
        find_by_first_name();
        break;
    case 5:
        find_by_last_name();
        break;
    case 6:
        find_students_by_courses();
        break;
    case 7:
        total_of_students();
        break;
    case 8:
        delete_student();
        break;
    case 9:
        update_student();
        break;
    case 10:
        show_all_students();
        break;
    case 11:
        Exit_Fun();
        break;
    default:
        printf("[ERROR] Wrong Option \n");
        break;
    }
}
```

- Add_student_file

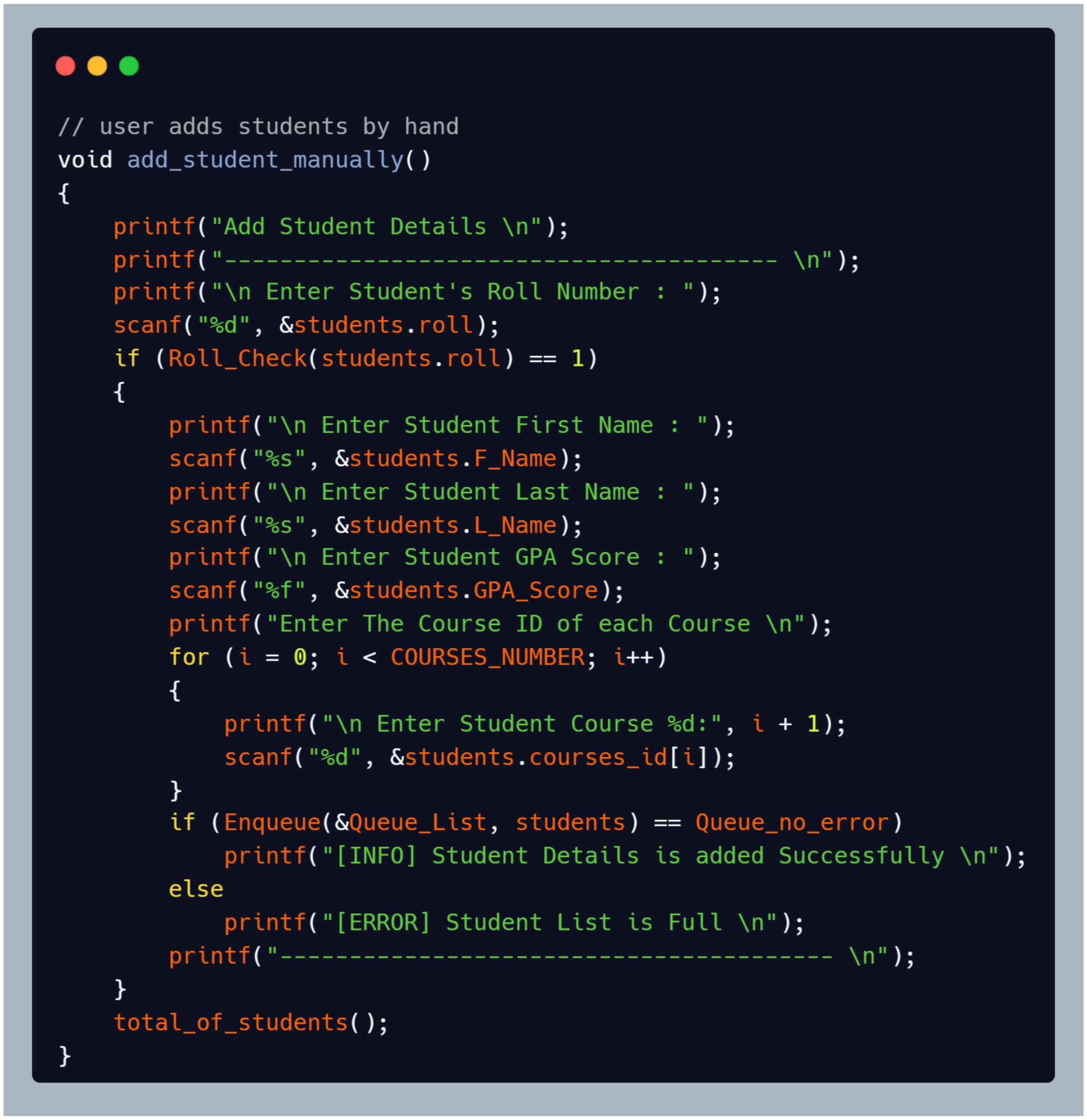
The function is used to add students into the system using a text file that contains the details of the students.



```c
void add_student_file()
{
    FILE *student_File;
    student_File = fopen("Data.txt", "r");
    if (student_File == NULL)
    {
        printf("[ERROR] Data.txt File is not Found. \n");
        printf("[ERROR] Adding Students from file Failed \n");
        return;
    }
    while (!feof(student_File))
    {
        fscanf(student_File, "%d", &students.roll);
        if (Roll_Check(students.roll) == 1)
        {
            fscanf(student_File, "%s", students.F_Name);
            fscanf(student_File, "%s", students.L_Name);
            fscanf(student_File, "%f", &students.GPA_Score);
            for (i = 0; i < COURSES_NUMBER; i++)
            {
                fscanf(student_File, "%d", &students.courses_id[i]);
            }
            if (Enqueue(&Queue_List, students) == Queue_no_error)
                printf("[INFO] Student Details is added Successfully 
\n");
            else
            {
                printf("[ERROR] Student details adding Failed by File 
\n");
                return;
            }
        }
    }
    fclose(student_File);
    total_of_students();
}
```

- Add_student_manually

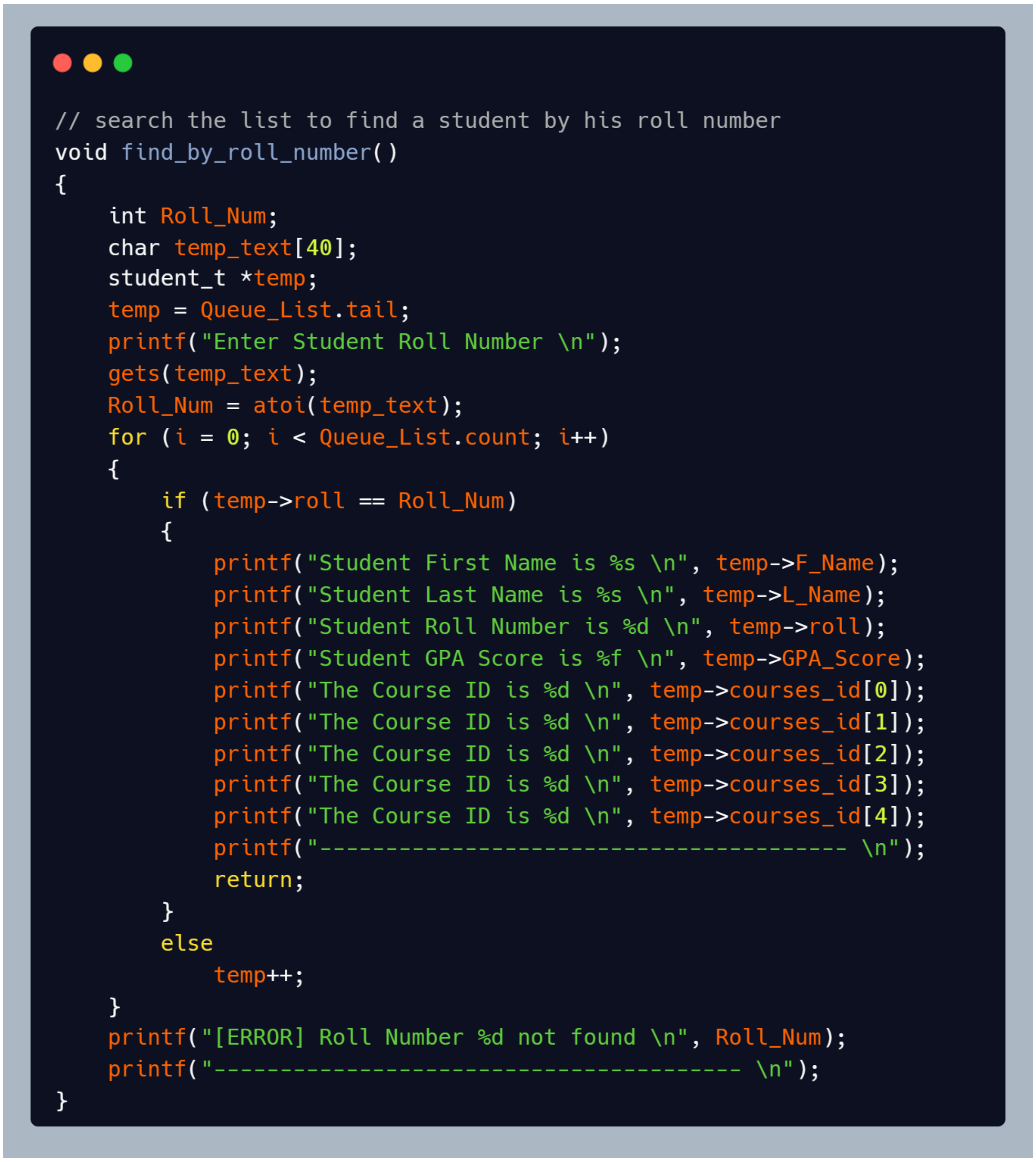
The function is used to add students to the system by the user Manually.



```c
// user adds students by hand
void add_student_manually()
{
    printf("Add Student Details \n");
    printf("---------------------------------------- \n");
    printf("\n Enter Student's Roll Number : ");
    scanf("%d", &students.roll);
    if (Roll_Check(students.roll) == 1)
    {
        printf("\n Enter Student First Name : ");
        scanf("%s", &students.F_Name);
        printf("\n Enter Student Last Name : ");
        scanf("%s", &students.L_Name);
        printf("\n Enter Student GPA Score : ");
        scanf("%f", &students.GPA_Score);
        printf("Enter The Course ID of each Course \n");
        for (i = 0; i < COURSES_NUMBER; i++)
        {
            printf("\n Enter Student Course %d:", i + 1);
            scanf("%d", &students.courses_id[i]);
        }
        if (Enqueue(&Queue_List, students) == Queue_no_error)
            printf("[INFO] Student Details is added Successfully \n");
        else
            printf("[ERROR] Student List is Full \n");
        printf("---------------------------------------- \n");
    }
    total_of_students();
}
```

* Find_by_roll_number

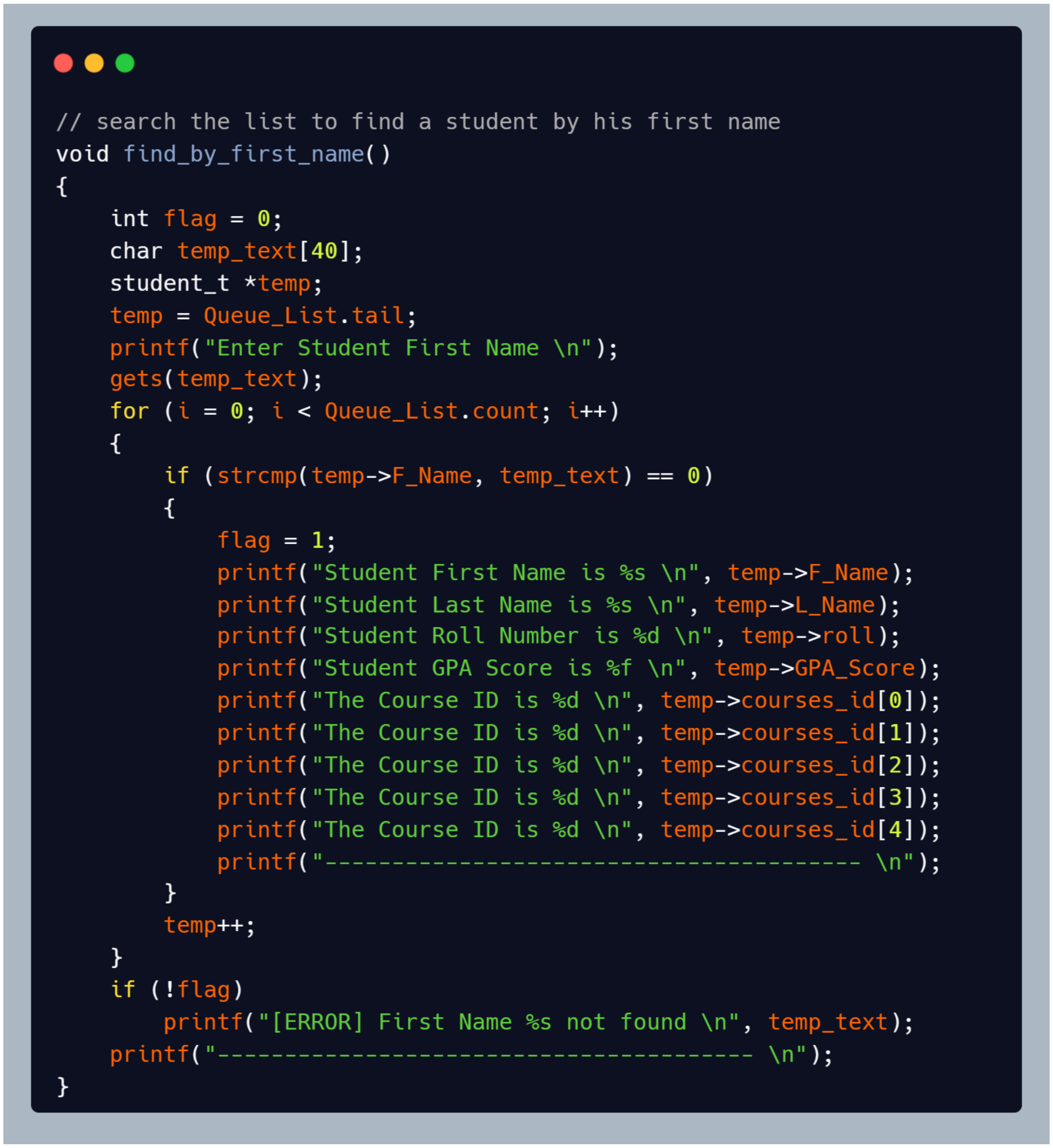
The function is used to search for a student record by his roll number.



```c
// search the list to find a student by his roll number
void find_by_roll_number()
{
    int Roll_Num;
    char temp_text[40];
    student_t *temp;
    temp = Queue_List.tail;
    printf("Enter Student Roll Number \n");
    gets(temp_text);
    Roll_Num = atoi(temp_text);
    for (i = 0; i < Queue_List.count; i++)
    {
        if (temp->roll == Roll_Num)
        {
            printf("Student First Name is %s \n", temp->F_Name);
            printf("Student Last Name is %s \n", temp->L_Name);
            printf("Student Roll Number is %d \n", temp->roll);
            printf("Student GPA Score is %f \n", temp->GPA_Score);
            printf("The Course ID is %d \n", temp->courses_id[0]);
            printf("The Course ID is %d \n", temp->courses_id[1]);
            printf("The Course ID is %d \n", temp->courses_id[2]);
            printf("The Course ID is %d \n", temp->courses_id[3]);
            printf("The Course ID is %d \n", temp->courses_id[4]);
            printf("---------------------------------------- \n");
            return;
        }
        else
            temp++;
    }
    printf("[ERROR] Roll Number %d not found \n", Roll_Num);
    printf("---------------------------------------- \n");
}
```

- Find_by_first_name

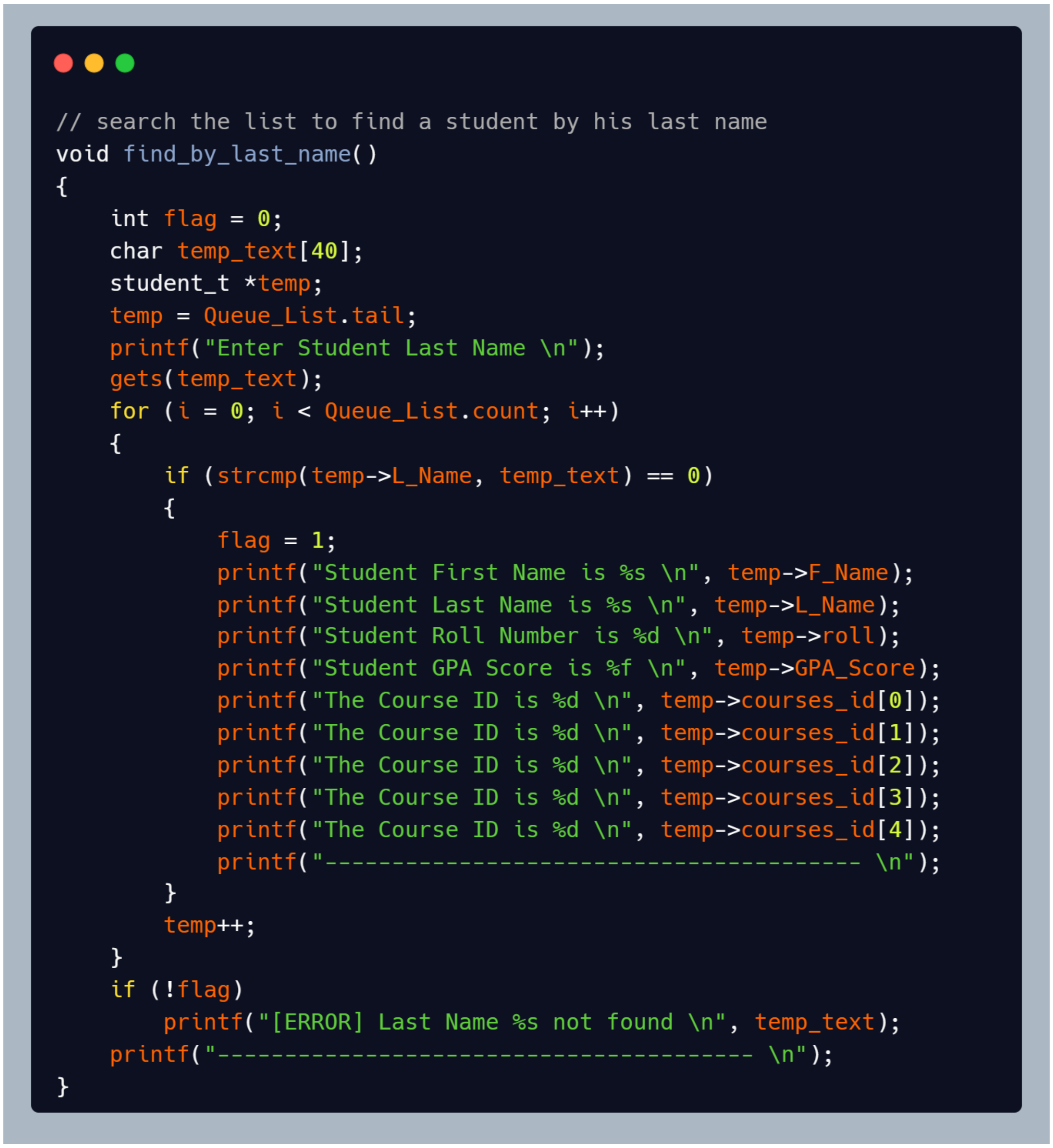
The function is used to search for a student record by his First Name.



```c
// search the list to find a student by his first name
void find_by_first_name()
{
    int flag = 0;
    char temp_text[40];
    student_t *temp;
    temp = Queue_List.tail;
    printf("Enter Student First Name \n");
    gets(temp_text);
    for (i = 0; i < Queue_List.count; i++)
    {
        if (strcmp(temp->F_Name, temp_text) == 0)
        {
            flag = 1;
            printf("Student First Name is %s \n", temp->F_Name);
            printf("Student Last Name is %s \n", temp->L_Name);
            printf("Student Roll Number is %d \n", temp->roll);
            printf("Student GPA Score is %f \n", temp->GPA_Score);
            printf("The Course ID is %d \n", temp->courses_id[0]);
            printf("The Course ID is %d \n", temp->courses_id[1]);
            printf("The Course ID is %d \n", temp->courses_id[2]);
            printf("The Course ID is %d \n", temp->courses_id[3]);
            printf("The Course ID is %d \n", temp->courses_id[4]);
            printf("---------------------------------------- \n");
        }
        temp++;
    }
    if (!flag)
        printf("[ERROR] First Name %s not found \n", temp_text);
    printf("---------------------------------------- \n");
}
```

- Find_by_last_name

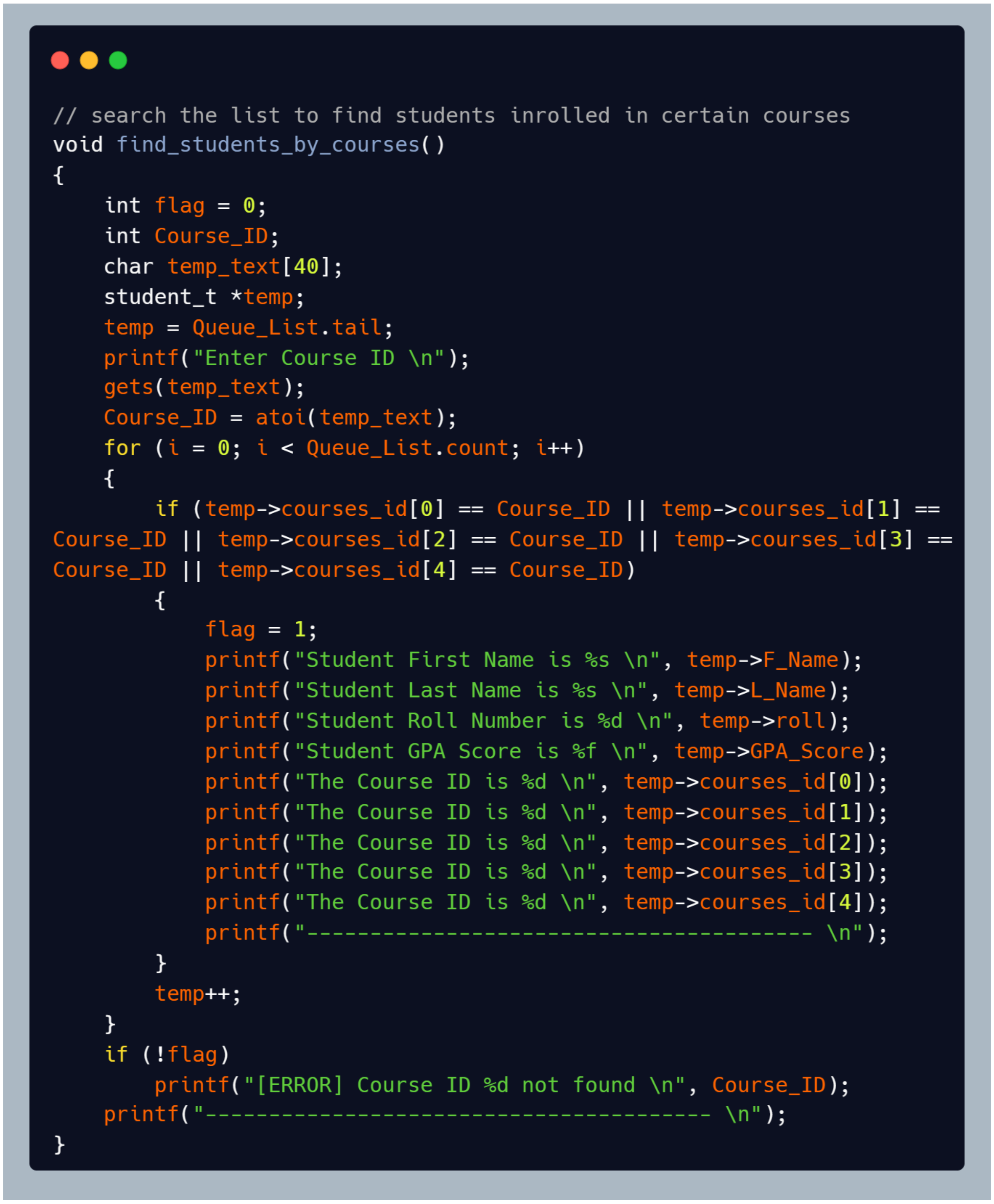
The function is used to search for a student record by his Last Name.



```c
// search the list to find a student by his last name
void find_by_last_name()
{
    int flag = 0;
    char temp_text[40];
    student_t *temp;
    temp = Queue_List.tail;
    printf("Enter Student Last Name \n");
    gets(temp_text);
    for (i = 0; i < Queue_List.count; i++)
    {
        if (strcmp(temp->L_Name, temp_text) == 0)
        {
            flag = 1;
            printf("Student First Name is %s \n", temp->F_Name);
            printf("Student Last Name is %s \n", temp->L_Name);
            printf("Student Roll Number is %d \n", temp->roll);
            printf("Student GPA Score is %f \n", temp->GPA_Score);
            printf("The Course ID is %d \n", temp->courses_id[0]);
            printf("The Course ID is %d \n", temp->courses_id[1]);
            printf("The Course ID is %d \n", temp->courses_id[2]);
            printf("The Course ID is %d \n", temp->courses_id[3]);
            printf("The Course ID is %d \n", temp->courses_id[4]);
            printf("---------------------------------------- \n");
        }
        temp++;
    }
    if (!flag)
        printf("[ERROR] Last Name %s not found \n", temp_text);
    printf("---------------------------------------- \n");
}
```

- Find_students_by_courses

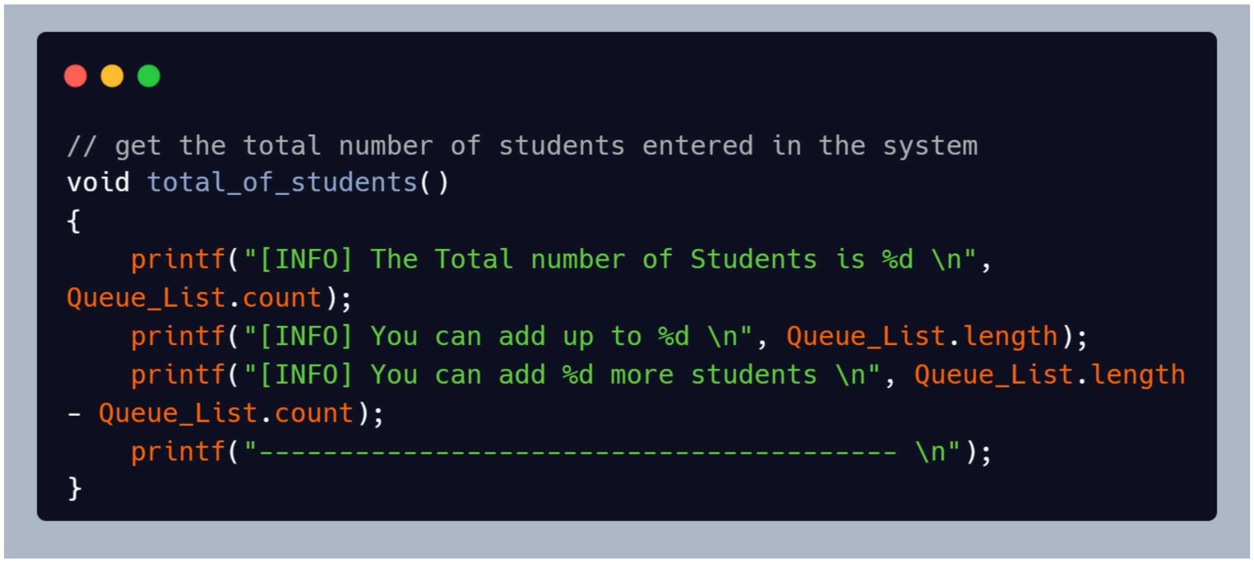
The function is used to search for students record by the course id they are in.



```
// search the list to find students inrolled in certain courses
void find_students_by_courses()
{
    int flag = 0;
    int Course_ID;
    char temp_text[40];
    student_t *temp;
    temp = Queue_List.tail;
    printf("Enter Course ID \n");
    gets(temp_text);
    Course_ID = atoi(temp_text);
    for (i = 0; i < Queue_List.count; i++)
    {
        if (temp->courses_id[0] == Course_ID || temp->courses_id[1] ==
Course_ID || temp->courses_id[2] == Course_ID || temp->courses_id[3] ==
Course_ID || temp->courses_id[4] == Course_ID)
        {
            flag = 1;
            printf("Student First Name is %s \n", temp->F_Name);
            printf("Student Last Name is %s \n", temp->L_Name);
            printf("Student Roll Number is %d \n", temp->roll);
            printf("Student GPA Score is %f \n", temp->GPA_Score);
            printf("The Course ID is %d \n", temp->courses_id[0]);
            printf("The Course ID is %d \n", temp->courses_id[1]);
            printf("The Course ID is %d \n", temp->courses_id[2]);
            printf("The Course ID is %d \n", temp->courses_id[3]);
            printf("The Course ID is %d \n", temp->courses_id[4]);
            printf("---------------------------------------- \n");
        }
        temp++;
    }
    if (!flag)
        printf("[ERROR] Course ID %d not found \n", Course_ID);
    printf("---------------------------------------- \n");
}
```

- ## Total_of_students

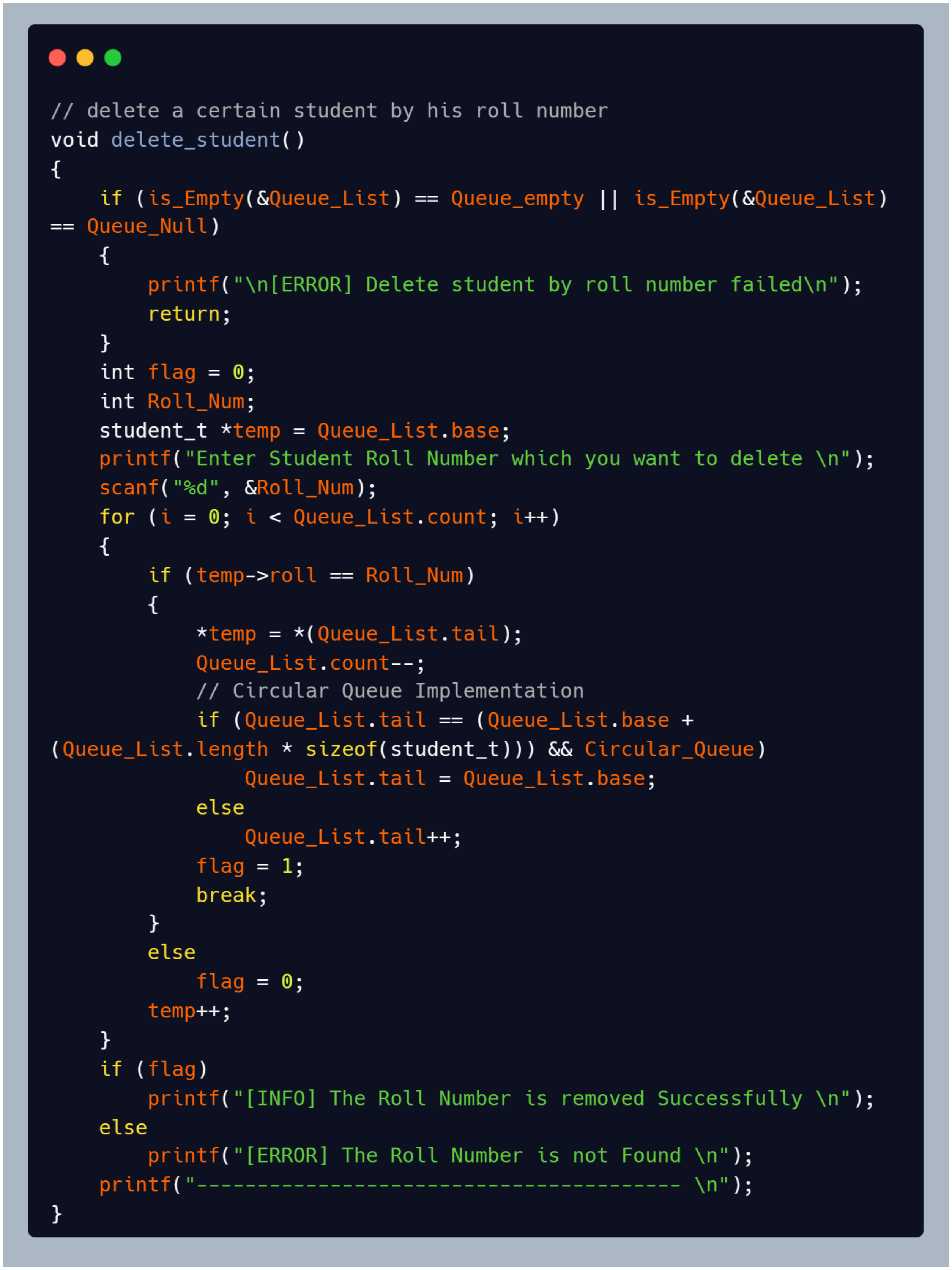
The function is used to get the total number of students in the system.



```
// get the total number of students entered in the system
void total_of_students()
{
    printf("[INFO] The Total number of Students is %d \n",
Queue_List.count);
    printf("[INFO] You can add up to %d \n", Queue_List.length);
    printf("[INFO] You can add %d more students \n", Queue_List.length
- Queue_List.count);
    printf("---------------------------------------- \n");
}
```

- ## Delete_student

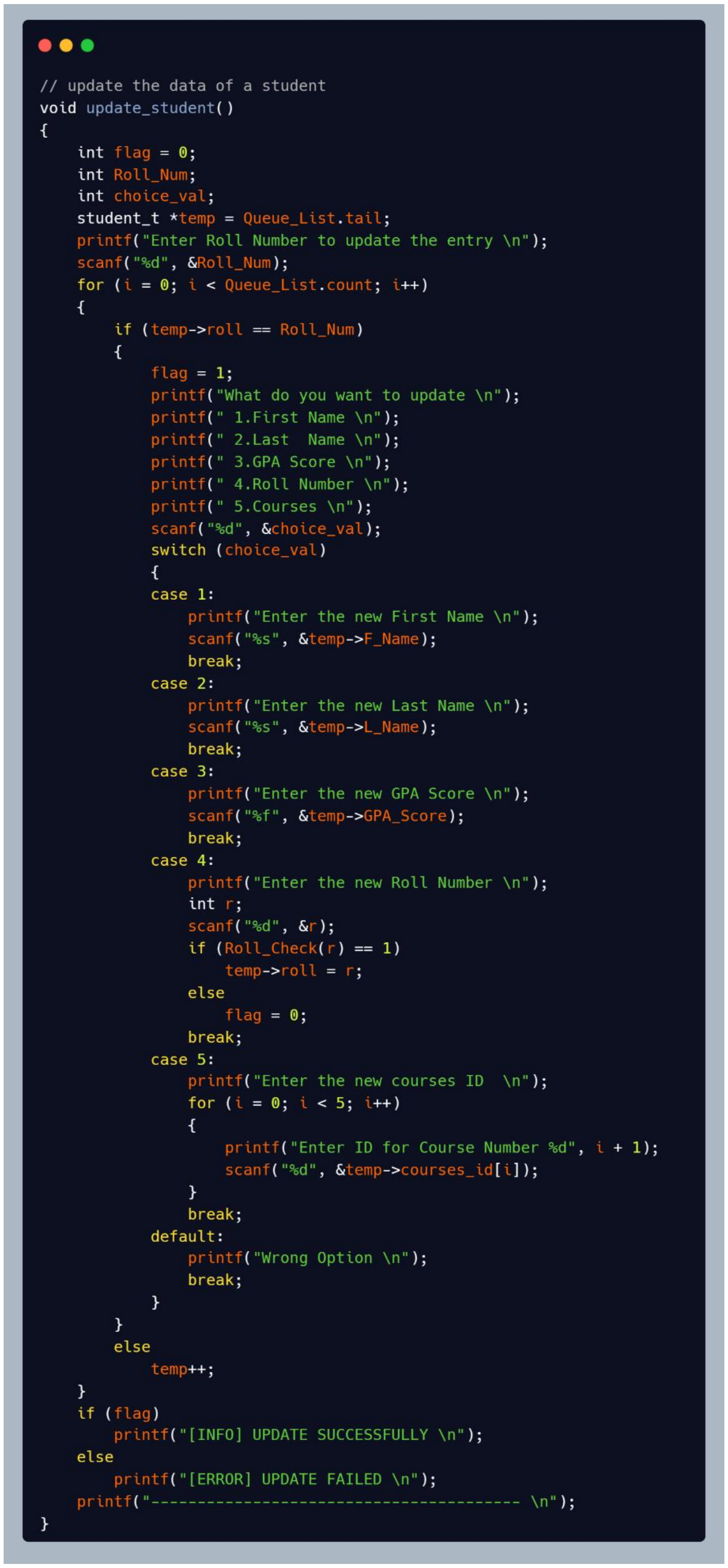
The function is used to delete a student record from the system.



```
// delete a certain student by his roll number
void delete_student()
{
    if (is_Empty(&Queue_List) == Queue_empty || is_Empty(&Queue_List)
== Queue_Null)
    {
        printf("\n[ERROR] Delete student by roll number failed\n");
        return;
    }
    int flag = 0;
    int Roll_Num;
    student_t *temp = Queue_List.base;
    printf("Enter Student Roll Number which you want to delete \n");
    scanf("%d", &Roll_Num);
    for (i = 0; i < Queue_List.count; i++)
    {
        if (temp->roll == Roll_Num)
        {
            *temp = *(Queue_List.tail);
            Queue_List.count--;
            // Circular Queue Implementation
            if (Queue_List.tail == (Queue_List.base +
(Queue_List.length * sizeof(student_t))) && Circular_Queue)
                Queue_List.tail = Queue_List.base;
            else
                Queue_List.tail++;
            flag = 1;
            break;
        }
        else
            flag = 0;
        temp++;
    }
    if (flag)
        printf("[INFO] The Roll Number is removed Successfully \n");
    else
        printf("[ERROR] The Roll Number is not Found \n");
    printf("---------------------------------------- \n");
}
```

- Update_student

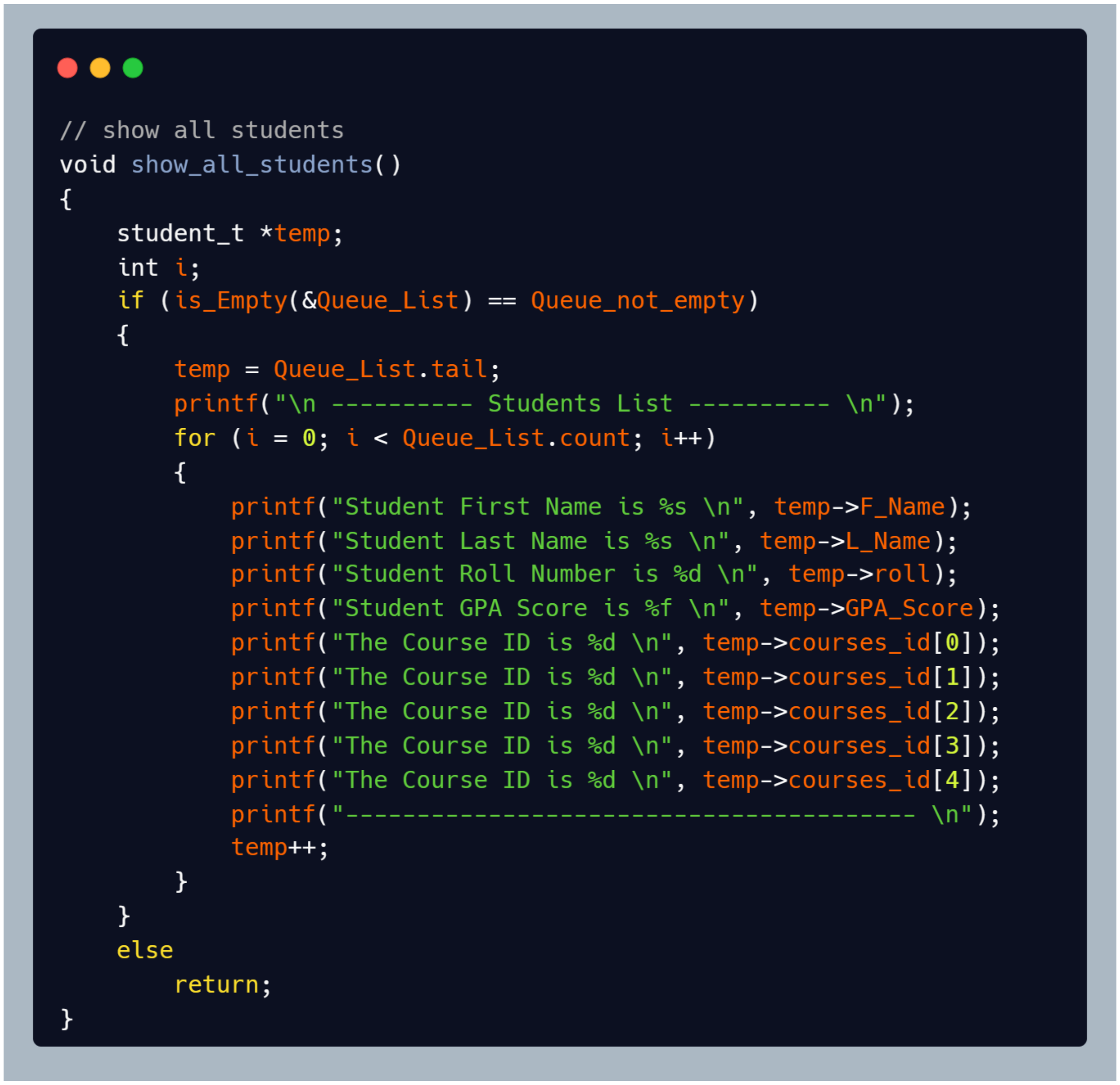
The function is used to update any detail in the student record, the roll number can be changed if the new number does not match any roll number in the system.



```c
// update the data of a student
void update_student()
{
    int flag = 0;
    int Roll_Num;
    int choice_val;
    student_t *temp = Queue_List.tail;
    printf("Enter Roll Number to update the entry \n");
    scanf("%d", &Roll_Num);
    for (i = 0; i < Queue_List.count; i++)
    {
        if (temp->roll == Roll_Num)
        {
            flag = 1;
            printf("What do you want to update \n");
            printf(" 1.First Name \n");
            printf(" 2.Last  Name \n");
            printf(" 3.GPA Score \n");
            printf(" 4.Roll Number \n");
            printf(" 5.Courses \n");
            scanf("%d", &choice_val);
            switch (choice_val)
            {
            case 1:
                printf("Enter the new First Name \n");
                scanf("%s", &temp->F_Name);
                break;
            case 2:
                printf("Enter the new Last Name \n");
                scanf("%s", &temp->L_Name);
                break;
            case 3:
                printf("Enter the new GPA Score \n");
                scanf("%f", &temp->GPA_Score);
                break;
            case 4:
                printf("Enter the new Roll Number \n");
                int r;
                scanf("%d", &r);
                if (Roll_Check(r) == 1)
                    temp->roll = r;
                else
                    flag = 0;
                break;
            case 5:
                printf("Enter the new courses ID  \n");
                for (i = 0; i < 5; i++)
                {
                    printf("Enter ID for Course Number %d", i + 1);
                    scanf("%d", &temp->courses_id[i]);
                }
                break;
            default:
                printf("Wrong Option \n");
                break;
            }
        }
        else
            temp++;
    }
    if (flag)
        printf("[INFO] UPDATE SUCCESSFULLY \n");
    else
        printf("[ERROR] UPDATE FAILED \n");
    printf("---------------------------------------- \n");
}
```

- ## Show_all_students

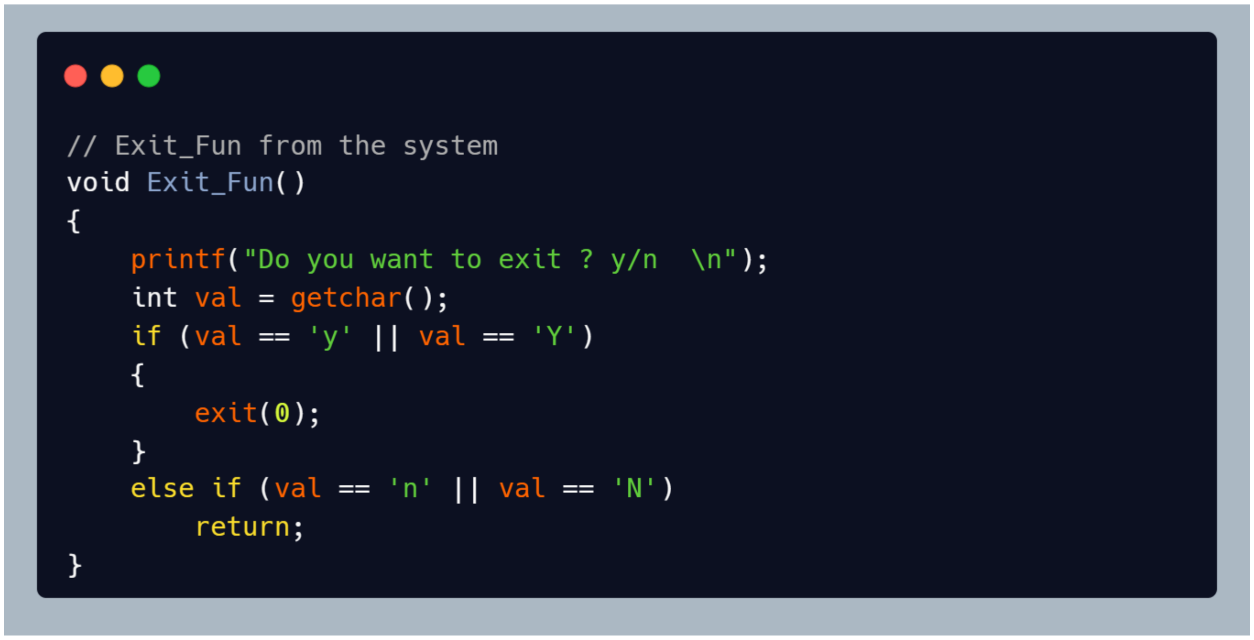
The function is used to show all students recorded in the system.



```c
// show all students
void show_all_students()
{
    student_t *temp;
    int i;
    if (is_Empty(&Queue_List) == Queue_not_empty)
    {
        temp = Queue_List.tail;
        printf("\n ---------- Students List ---------- \n");
        for (i = 0; i < Queue_List.count; i++)
        {
            printf("Student First Name is %s \n", temp->F_Name);
            printf("Student Last Name is %s \n", temp->L_Name);
            printf("Student Roll Number is %d \n", temp->roll);
            printf("Student GPA Score is %f \n", temp->GPA_Score);
            printf("The Course ID is %d \n", temp->courses_id[0]);
            printf("The Course ID is %d \n", temp->courses_id[1]);
            printf("The Course ID is %d \n", temp->courses_id[2]);
            printf("The Course ID is %d \n", temp->courses_id[3]);
            printf("The Course ID is %d \n", temp->courses_id[4]);
            printf("---------------------------------------- \n");
            temp++;
        }
    }
    else
        return;
}
```

- ## Exit_Fun

The Function is used to exit from the system if the user wants to.

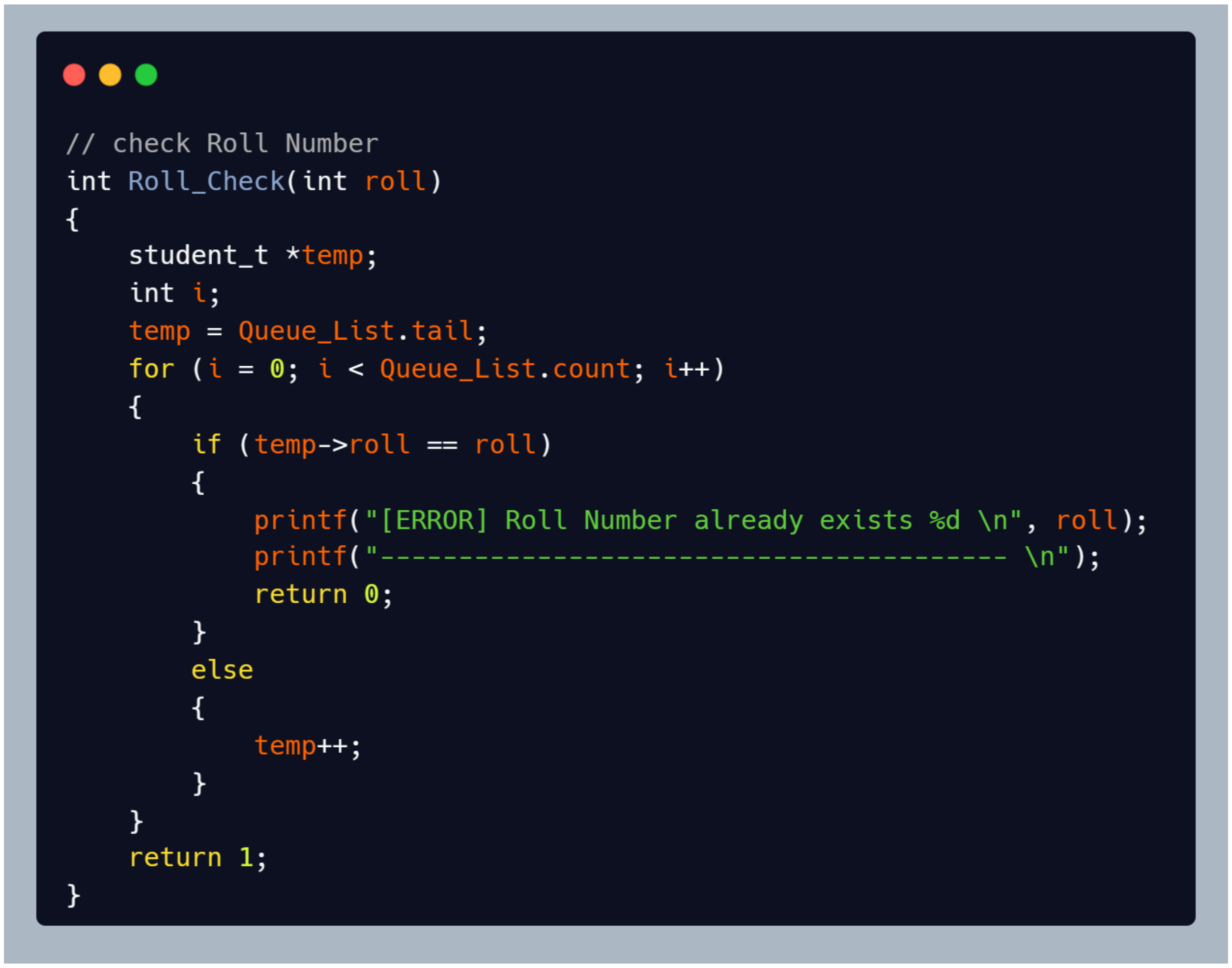


```
// Exit_Fun from the system
void Exit_Fun()
{
    printf("Do you want to exit ? y/n  \n");
    int val = getchar();
    if (val == 'y' || val == 'Y')
    {
        exit(0);
    }
    else if (val == 'n' || val == 'N')
        return;
}
```

- Roll_Check

The function is used to check if the roll number exists in the system or not.

It is used in adding and updating records, so the user won’t add an already recorded roll number.



```c
// check Roll Number
int Roll_Check(int roll)
{
    student_t *temp;
    int i;
    temp = Queue_List.tail;
    for (i = 0; i < Queue_List.count; i++)
    {
        if (temp->roll == roll)
        {
            printf("[ERROR] Roll Number already exists %d \n", roll);
            printf("---------------------------------------- \n");
            return 0;
        }
        else
        {
            temp++;
        }
    }
    return 1;
}
```