Library Management System Documentation

Team HEX CLANE

National Telecommunication Institute

Overview:

This system allows users to manage books from different departments. Users can add, delete, edit, search, and get data of books using the functions provided by the system. The system uses a linked list data structure to store and manipulate the books.

Sprint 1: Planning and Design (18 Oct)

In this sprint, the team will define the scope, objectives, and requirements of the project. The team will also design the main menu and the user interface of the system. The team will develop the following functions:

- mainmenu – This function will display the main menu of the system.
- returnfunc – This function will return to the main menu when the user presses a key.
- Password – This function will ask the user to input a password to access the system.

Sprint 2: Book Management (19 - 20 Oct)

In this sprint, the team will implement the core functionality of the system: managing books from different departments. The team will use the following functions:

- addbooks – This function will add books to a file. The user will need to mention the department to which they want to add the book.
- deletebooks – This function will delete books from a file. The user will need to mention the department from which they want to delete the book.
- editbooks – This function will edit books in a file. The user will need to mention the department and the ID of the book they want to edit.
- searchbooks – This function will search books in a file. The user will need to enter a keyword or an ID to search for books.
- getdata – This function will ask for data input from the user, such as book title, author, price, etc.

Sprint 3: Book Issuing (20 Oct)

In this sprint, the team will implement the feature of issuing books to students. The team will use the following functions:

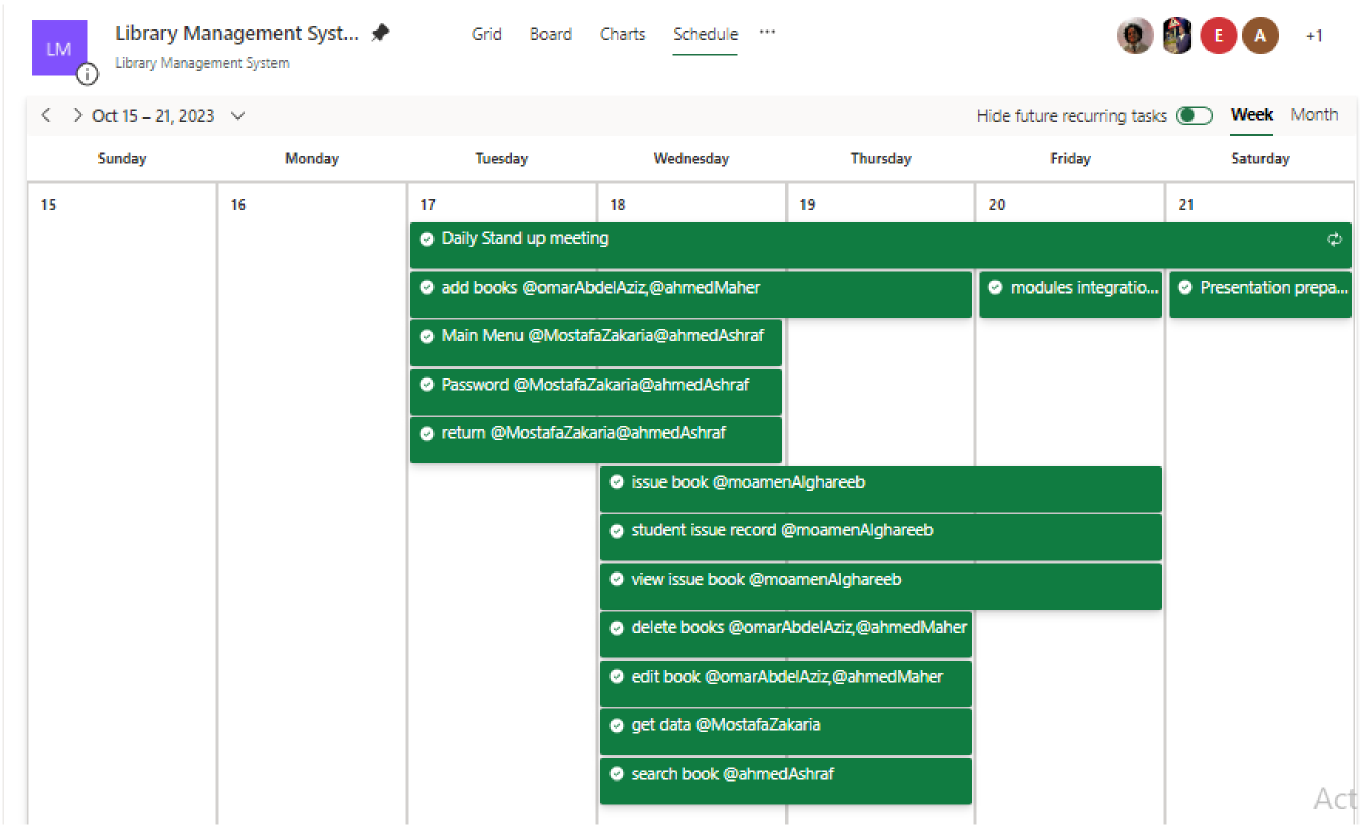
- issuebooks – This function will issue books to students. The user will need to enter the ID of the book and the student's name and ID.

- viewissuebooks – This function will view issued book details. The user will need to mention the department to view the issued books from that department.

- studentissuerecord – This function will keep record of the student to whom the book has been issued.

Sprint 4: Testing and Deployment (21 Oct)

In this sprint, the team will test and deploy the system. The team will perform unit testing, integration testing and system testing on each function and module. The team will also fix any bugs or errors that may arise during testing. The team will then deploy the system to a suitable platform and ensure its proper functioning.



Daily Stand up meeting
add books @omarAbdelAziz,@ahmedMaher
modules integratio...
Presentation prepa...
Main Menu @MostafaZakaria@ahmedAshraf
Password @MostafaZakaria@ahmedAshraf
return @MostafaZakaria@ahmedAshraf
issue book @moamenAlghareeb
student issue record @moamenAlghareeb
view issue book @moamenAlghareeb
delete books @omarAbdelAziz,@ahmedMaher
edit book @omarAbdelAziz,@ahmedMaher
get data @MostafaZakaria
search book @ahmedAshraf

Data Structures:

The system defines four structures: `struct book` , `struct Node` ,`struct student` and `struct student_Node`..

struct book:

This structure represents a book with the following fields:

- **`int ID`:** a unique identifier for the book
- **`char Department[30]`:** the name of the department that the book belongs to
- **`char title[30]`:** the title of the book
- **`char author[30]`:** the name of the author of the book
- **`char publisher[30]`:** the name of the publisher of the book
- **`int numberOfPages`:** the number of pages in the book
- **`int numberOfBooks`:** the total number of copies of the book
- **`int numberOfAvailableBooks`:** the number of copies of the book that are available for borrowing
- **`int numberOfBorrowedBooks`:** the number of copies of the book that are currently borrowed
- **`int price`:** the price of the book in Egyptian pounds

struct Node:

This structure represents a node in the linked list that contains a book and a pointer to the next node.

- **`struct book b`:** a book object
- **`struct book *book_ptr`:** a pointer to the next node in the list

struct student

This structure represents a book with the following fields:

- **“char name[30]”** the name of the student
- **“char issuedBookTitle[30]”** the title of the issued book
- **“int ID”** a unique identifier for the student
- **“int issuedBookID”** the ID of the issued book

struct student_node:

This structure represents a node in the linked list that contains a student and a pointer to the next student node.

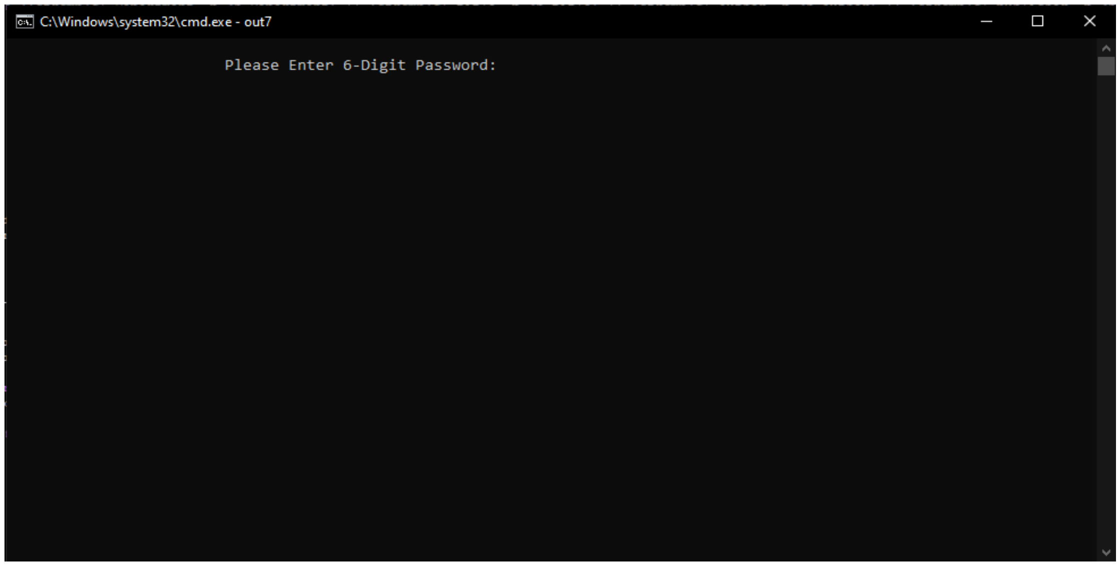
- **`struct student s`:** a book object
- **`struct student *student_ptr`:** a pointer to the next node in the list

Functions

The system provides the following functions for managing books:

1) Password:

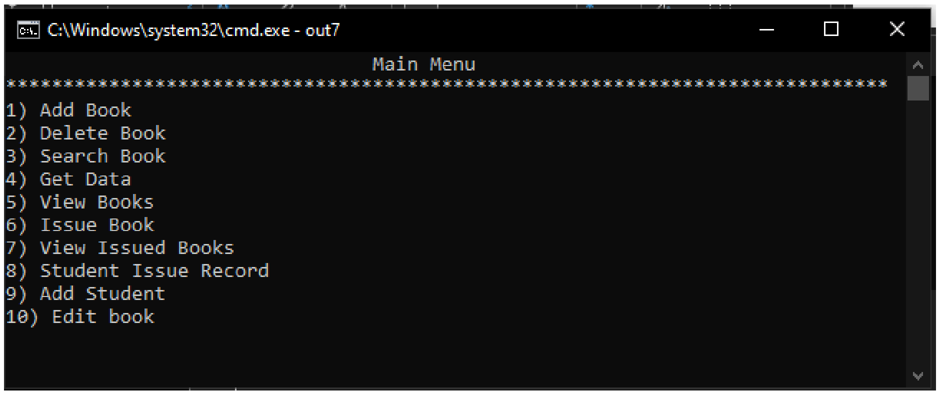
- The function doesn't take nor return any data as argument; the code asks the use to enter a 6- digit password then it compares the input with a pre-defined password of the user.
- If the password was incorrect the menu will display wrong entry and the user is asked for input again.
- You have 3 trails until you are locked out of the system.
- If the password was correct, it displays the main menu.



```
                    Please Enter 6-Digit Password:
```

2) Main menu:

- The program displays the available options for the user to chose from and he is asked to input data to the required task.



```
                                Main Menu
******************************************************************************
1) Add Book
2) Delete Book
3) Search Book
4) Get Data
5) View Books
6) Issue Book
7) View Issued Books
8) Student Issue Record
9) Add Student
10) Edit book
```

3) Return:

The return function will be available at every screen so that you can return to the main menu screen at any time.

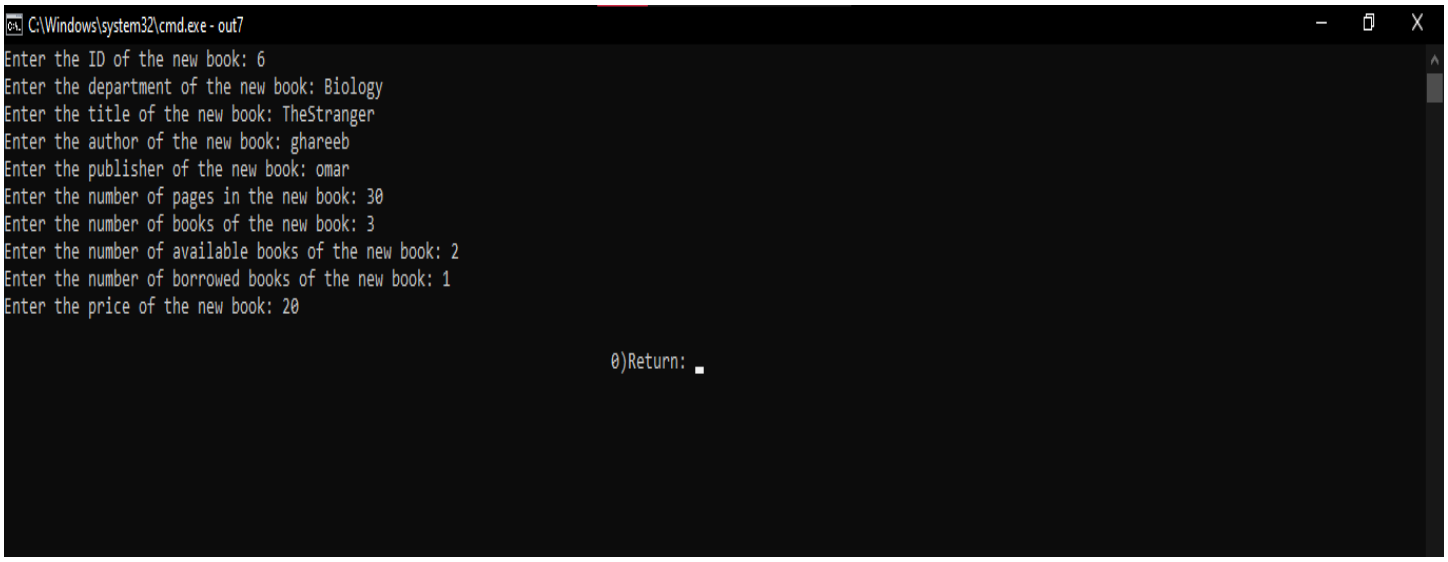
4) **void Add_Book** (struct Node **head, struct book new_book)

- This function adds a new book to the linked list in ascending order by ID. It takes two parameters:

- `struct Node **head`: a pointer to the pointer to the head of the list

- `struct book new_book`: a book object to be added to the list

The function allocates memory for a new node, assigns it the new book, and inserts it into the appropriate position in the list.



```
C:\Windows\system32\cmd.exe - out7
Enter the ID of the new book: 6
Enter the department of the new book: Biology
Enter the title of the new book: TheStranger
Enter the author of the new book: ghareeb
Enter the publisher of the new book: omar
Enter the number of pages in the new book: 30
Enter the number of books of the new book: 3
Enter the number of available books of the new book: 2
Enter the number of borrowed books of the new book: 1
Enter the price of the new book: 20

                                                  0)Return:
```

5) void What_to_edit(struct Node **Book)

This function edits a book in the linked list by asking for user input. It takes one parameter:

- `struct Node **Book`: a pointer to the pointer to the node that contains the book to be edited

The function displays a menu of options for editing different fields of the book, such as department, title, author, etc. It then asks for user input and updates the corresponding field of the book.

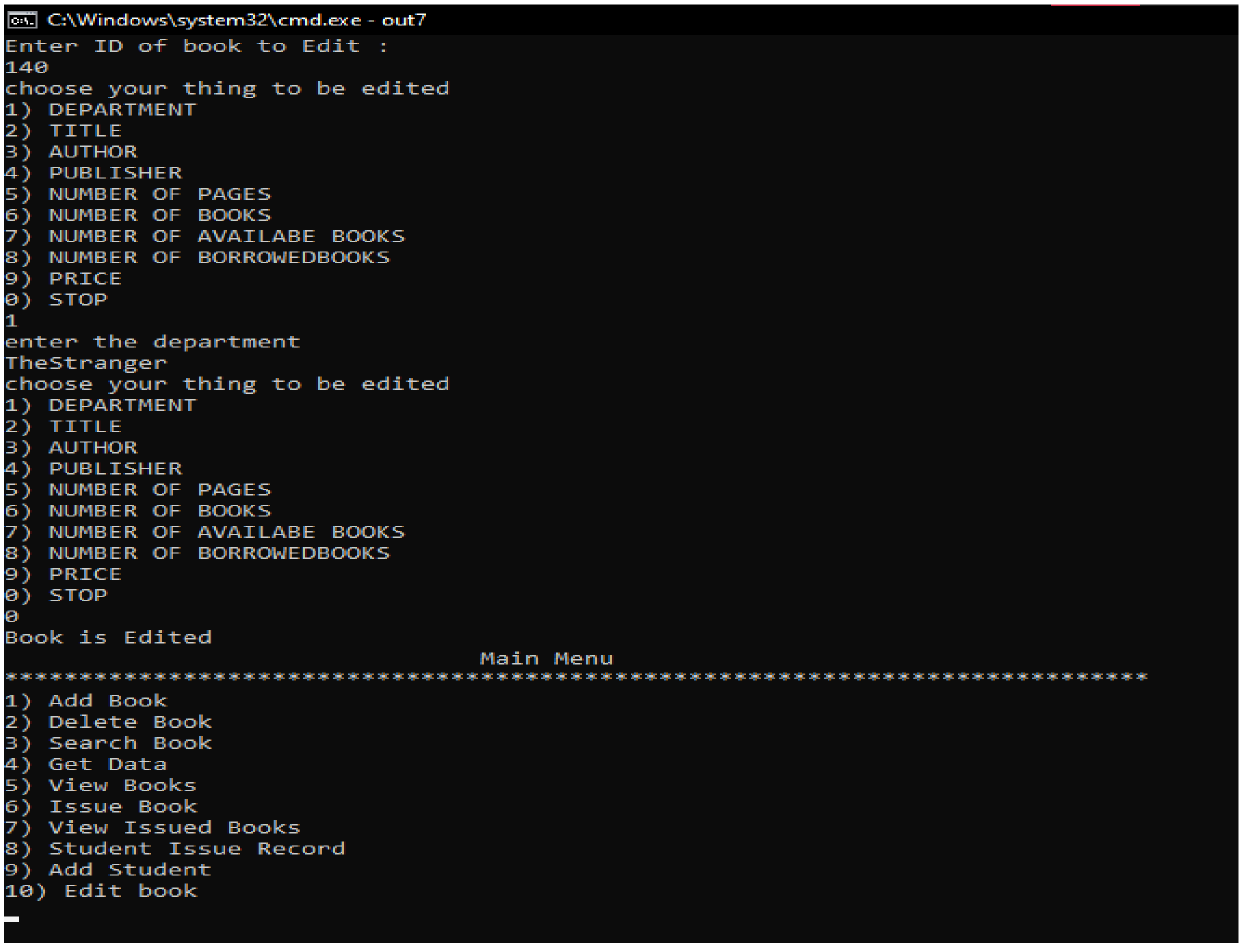
6) int Edit_Book(struct Node **head, int Book_ID**)**

This function edits a book in the linked list by ID. It takes two parameters:

struct Node head: a pointer to the pointer to the head of the list

int Book_ID: an integer representing the ID of the book to be edited.

The function searches for the node that contains the book with the given ID and calls What_to_edit function to edit it. It returns 1 if successful, or 0 if not found.



```
C:\Windows\system32\cmd.exe - out7
Enter ID of book to Edit :
140
choose your thing to be edited
1) DEPARTMENT
2) TITLE
3) AUTHOR
4) PUBLISHER
5) NUMBER OF PAGES
6) NUMBER OF BOOKS
7) NUMBER OF AVAILABE BOOKS
8) NUMBER OF BORROWEDBOOKS
9) PRICE
0) STOP
1
enter the department
TheStranger
choose your thing to be edited
1) DEPARTMENT
2) TITLE
3) AUTHOR
4) PUBLISHER
5) NUMBER OF PAGES
6) NUMBER OF BOOKS
7) NUMBER OF AVAILABE BOOKS
8) NUMBER OF BORROWEDBOOKS
9) PRICE
0) STOP
0
Book is Edited
                                Main Menu
*****************************************************************************
1) Add Book
2) Delete Book
3) Search Book
4) Get Data
5) View Books
6) Issue Book
7) View Issued Books
8) Student Issue Record
9) Add Student
10) Edit book
```

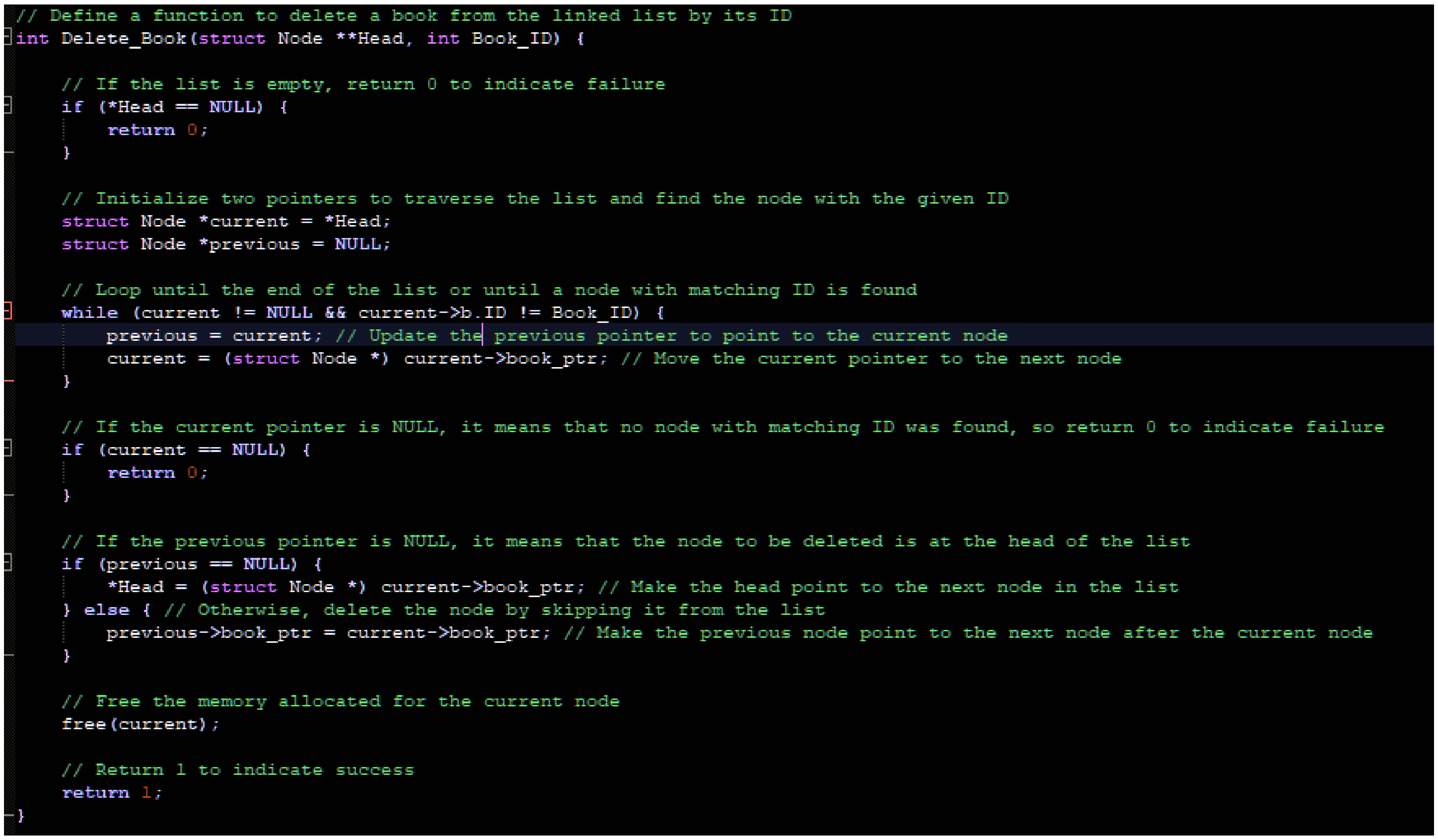
7) int Delete_Book(struct Node **head, int Book_ID)

This function deletes a book from the linked list by ID. It takes two parameters:

- `struct Node **head`: a pointer to the pointer to the head of the list

- `int Book_ID`: an integer representing the ID of the book to be deleted

The function searches for the node that contains the book with the given ID, removes it from the list, and frees its memory. It returns 1 if successful, or 0 if not found.

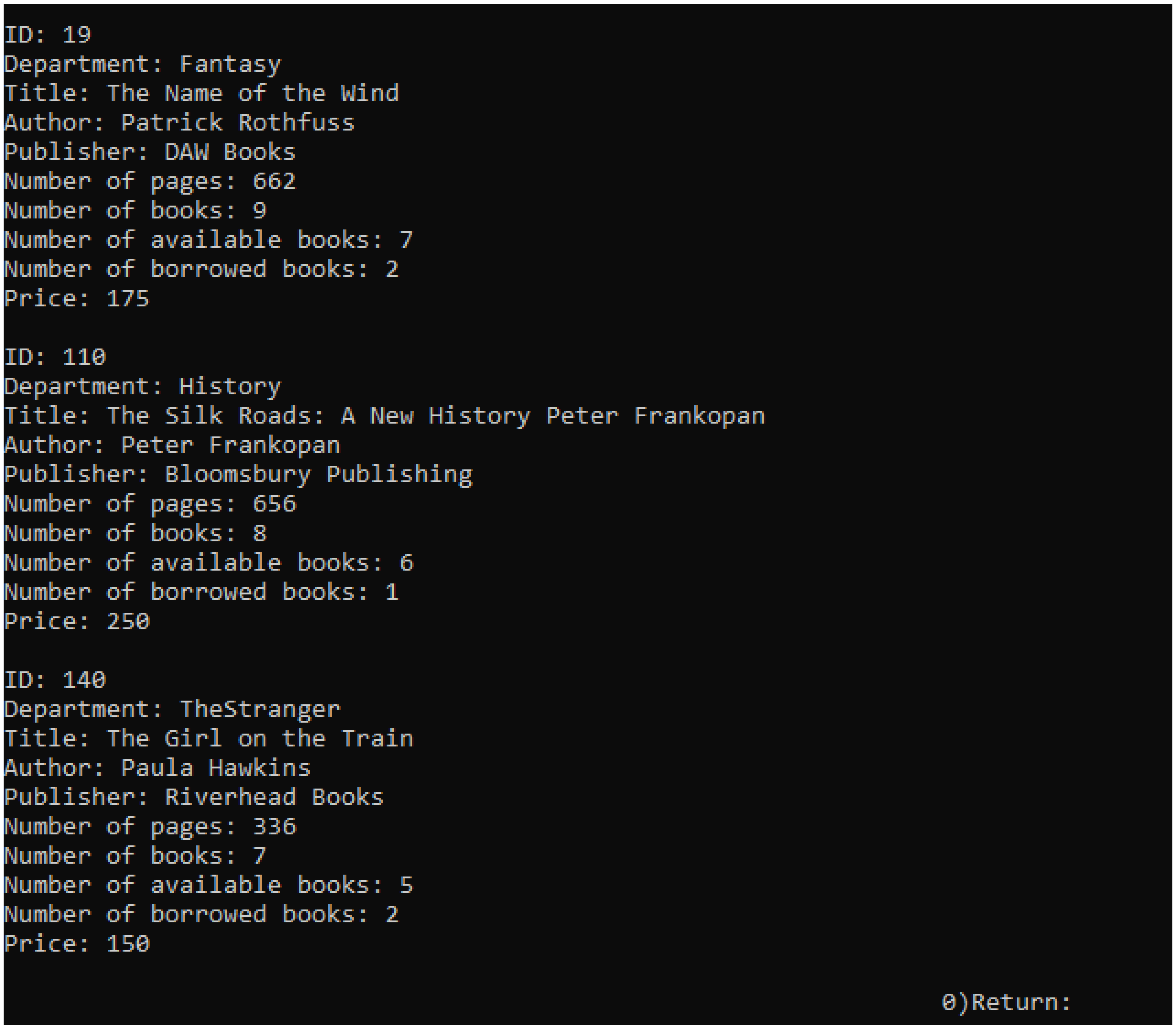


```
// Define a function to delete a book from the linked list by its ID
int Delete_Book(struct Node **Head, int Book_ID) {

    // If the list is empty, return 0 to indicate failure
    if (*Head == NULL) {
        return 0;
    }

    // Initialize two pointers to traverse the list and find the node with the given ID
    struct Node *current = *Head;
    struct Node *previous = NULL;

    // Loop until the end of the list or until a node with matching ID is found
    while (current != NULL && current->b.ID != Book_ID) {
        previous = current; // Update the previous pointer to point to the current node
        current = (struct Node *) current->book_ptr; // Move the current pointer to the next node
    }

    // If the current pointer is NULL, it means that no node with matching ID was found, so return 0 to indicate failure
    if (current == NULL) {
        return 0;
    }

    // If the previous pointer is NULL, it means that the node to be deleted is at the head of the list
    if (previous == NULL) {
        *Head = (struct Node *) current->book_ptr; // Make the head point to the next node in the list
    } else { // Otherwise, delete the node by skipping it from the list
        previous->book_ptr = current->book_ptr; // Make the previous node point to the next node after the current node
    }

    // Free the memory allocated for the current node
    free(current);

    // Return 1 to indicate success
    return 1;
}
```

8) void view_books(struct Node **head**)**

This function prints out all books in the linked list. It takes one parameter:

struct Node head: a pointer to the pointer to the head of the list The function iterates through the list and prints out the details of each book, such as ID, department, title, author, etc. It also prints a blank line after each book for readability.



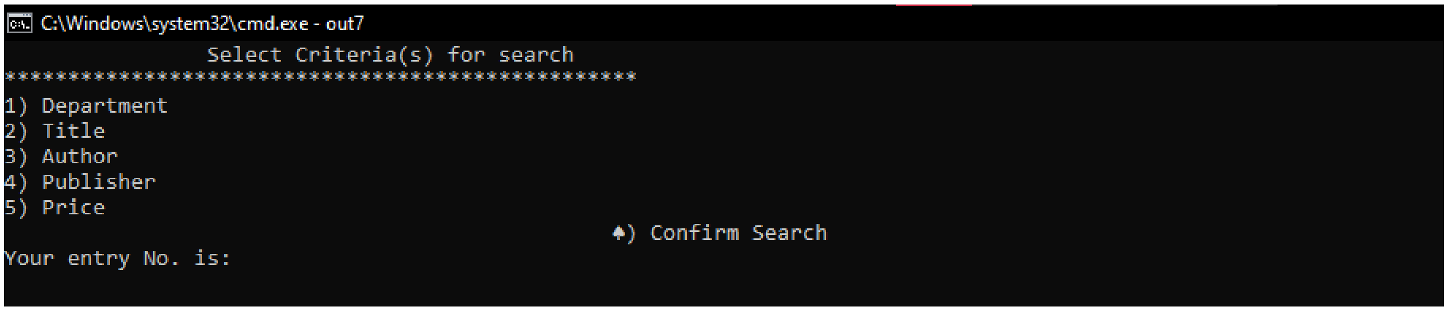
```
ID: 19
Department: Fantasy
Title: The Name of the Wind
Author: Patrick Rothfuss
Publisher: DAW Books
Number of pages: 662
Number of books: 9
Number of available books: 7
Number of borrowed books: 2
Price: 175

ID: 110
Department: History
Title: The Silk Roads: A New History Peter Frankopan
Author: Peter Frankopan
Publisher: Bloomsbury Publishing
Number of pages: 656
Number of books: 8
Number of available books: 6
Number of borrowed books: 1
Price: 250

ID: 140
Department: TheStranger
Title: The Girl on the Train
Author: Paula Hawkins
Publisher: Riverhead Books
Number of pages: 336
Number of books: 7
Number of available books: 5
Number of borrowed books: 2
Price: 150

                                                                0)Return:
```

10) void Search_Books(struct Node **head)

This function searches for books in the linked list by keyword or ID. It takes one parameter:

- `struct Node *head`: a pointer to the head of the list

The function asks for user input for either a keyword or an ID to search for books. It then traverses through the list and prints out any books that match or contain the input.



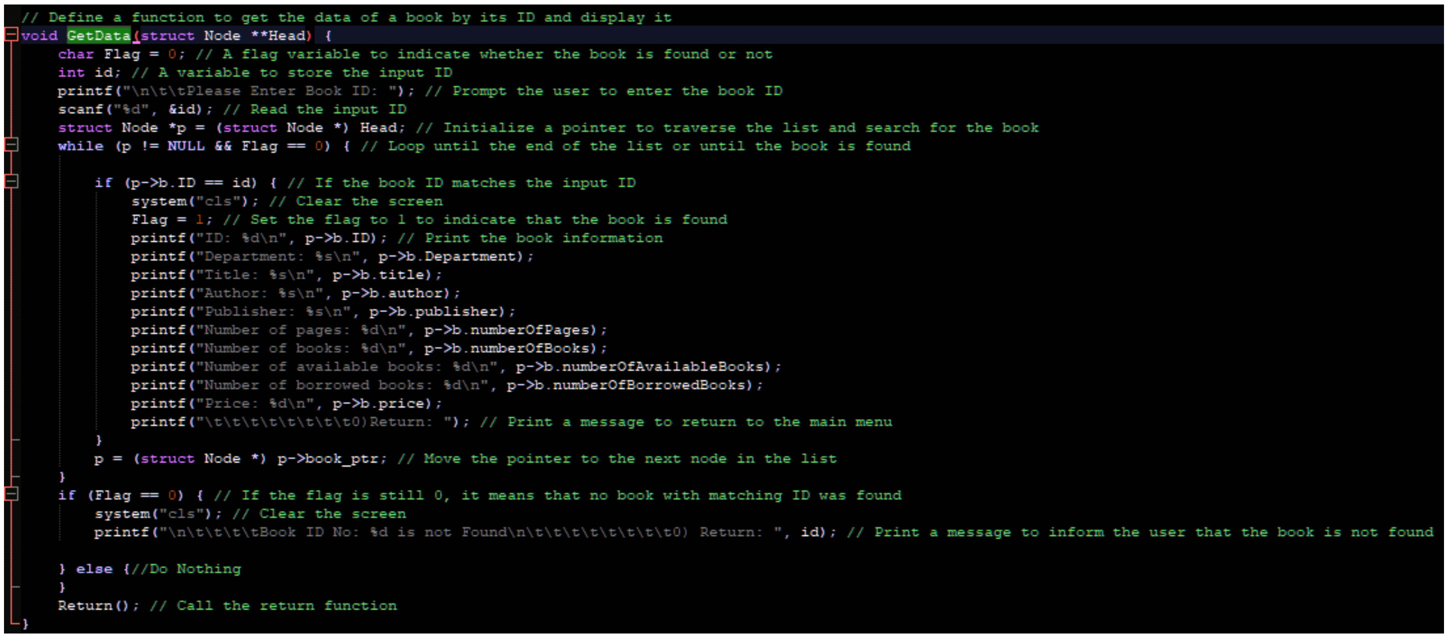
```
C:\Windows\system32\cmd.exe - out7
                Select Criteria(s) for search
**************************************************
1) Department
2) Title
3) Author
4) Publisher
5) Price
                                                ♠) Confirm Search
Your entry No. is:
```

11) void Get_Data(struct Node **head)

This function asks for user input for data of a new book and adds it to the list. It takes one parameter:

- `struct Node **head`: a pointer to the pointer to the head of the list

The function prompts for user input for book ID and print the book contents.



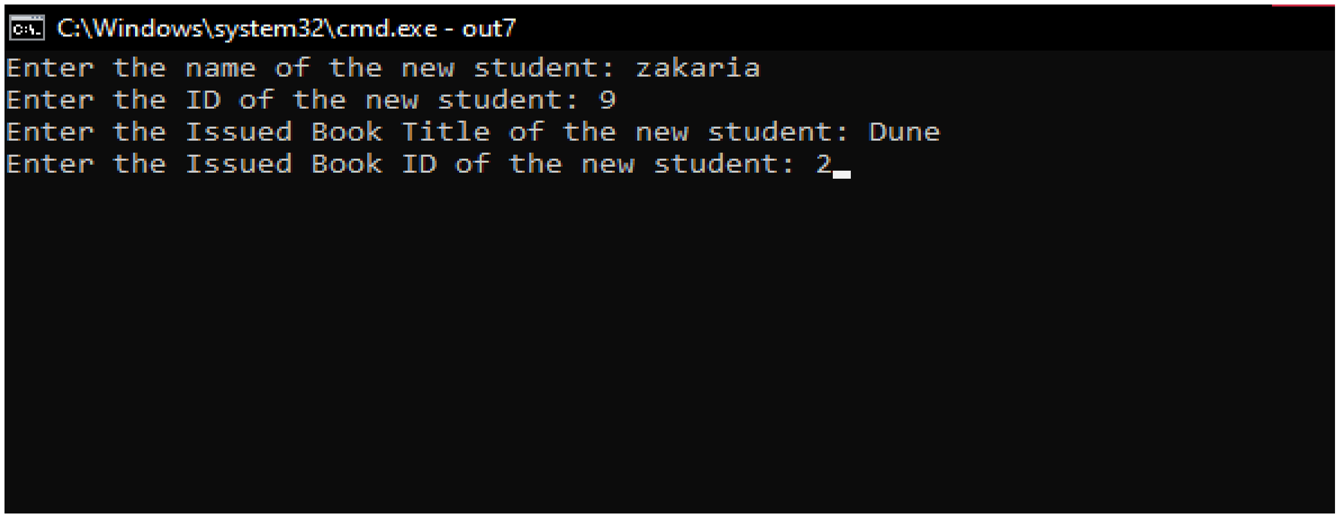
```c
// Define a function to get the data of a book by its ID and display it
void GetData(struct Node **Head) {
    char Flag = 0; // A flag variable to indicate whether the book is found or not
    int id; // A variable to store the input ID
    printf("\n\t\tPlease Enter Book ID: "); // Prompt the user to enter the book ID
    scanf("%d", &id); // Read the input ID
    struct Node *p = (struct Node *) Head; // Initialize a pointer to traverse the list and search for the book
    while (p != NULL && Flag == 0) { // Loop until the end of the list or until the book is found

        if (p->b.ID == id) { // If the book ID matches the input ID
            system("cls"); // Clear the screen
            Flag = 1; // Set the flag to 1 to indicate that the book is found
            printf("ID: %d\n", p->b.ID); // Print the book information
            printf("Department: %s\n", p->b.Department);
            printf("Title: %s\n", p->b.title);
            printf("Author: %s\n", p->b.author);
            printf("Publisher: %s\n", p->b.publisher);
            printf("Number of pages: %d\n", p->b.numberOfPages);
            printf("Number of books: %d\n", p->b.numberOfBooks);
            printf("Number of available books: %d\n", p->b.numberOfAvailableBooks);
            printf("Number of borrowed books: %d\n", p->b.numberOfBorrowedBooks);
            printf("Price: %d\n", p->b.price);
            printf("\t\t\t\t\t\t\t\t0)Return: "); // Print a message to return to the main menu
        }
        p = (struct Node *) p->book_ptr; // Move the pointer to the next node in the list
    }
    if (Flag == 0) { // If the flag is still 0, it means that no book with matching ID was found
        system("cls"); // Clear the screen
        printf("\n\t\t\t\tBook ID No: %d is not Found\n\t\t\t\t\t\t\t\t0) Return: ", id); // Print a message to inform the user that the book is not found

    } else {//Do Nothing
    }
    Return(); // Call the return function
}
```

12) void Add_Student(Student_Node_t **student_head, student_t new_student)

This function adds a new Student to the linked list in ascending order by ID. It takes two parameters:

- ` Student_Node_t ** student_head `: a pointer to the pointer to the head of the student list
- ` student_t  new_student `: a student object to be added to the list

The function allocates memory for a new student node, assigns it the new student, and inserts it into the appropriate position in the list.



C:\Windows\system32\cmd.exe - out7

```
Enter the name of the new student: zakaria
Enter the ID of the new student: 9
Enter the Issued Book Title of the new student: Dune
Enter the Issued Book ID of the new student: 2
```

13) void issueBook(struct Node** head,Student_Node_t **student_head)

This function searches for books in the linked list by title or ID. Then search for student in student linked list by name or ID then issue the book if available to the student

It takes two parameters:

- ` struct Node** head `: a pointer to the head of the book list

- “Student_Node_t **student_head” : a pointer to the head of the students list

The function asks for user input for either a title or an ID to search for books. It then traverses through the book list and check if any available copies to be issued then The function asks for user input for either a name or an ID to search for student and check if the student have issued book already or not if he has issued book he cannot issue now until he delete the issue but if he hasn’t issued book assign the book to him in the list.

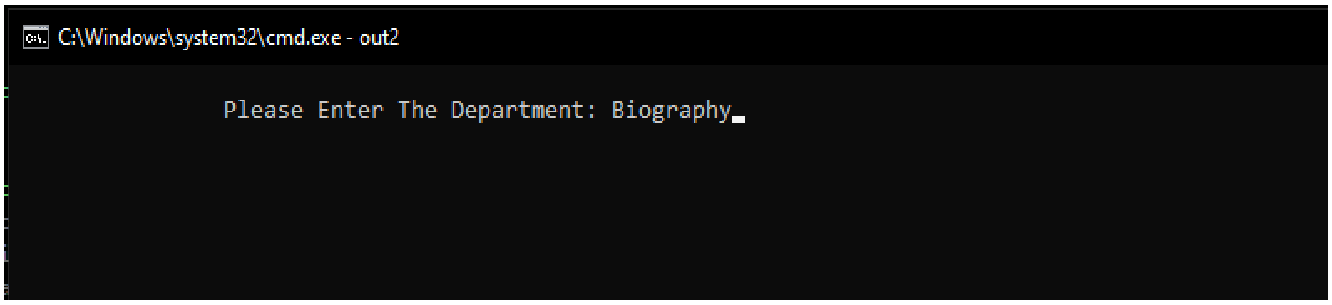
14) void viewIssueBooks (struct Node** head)

This function will view issued book details. The user will need to mention the department to view the issued books from that department.

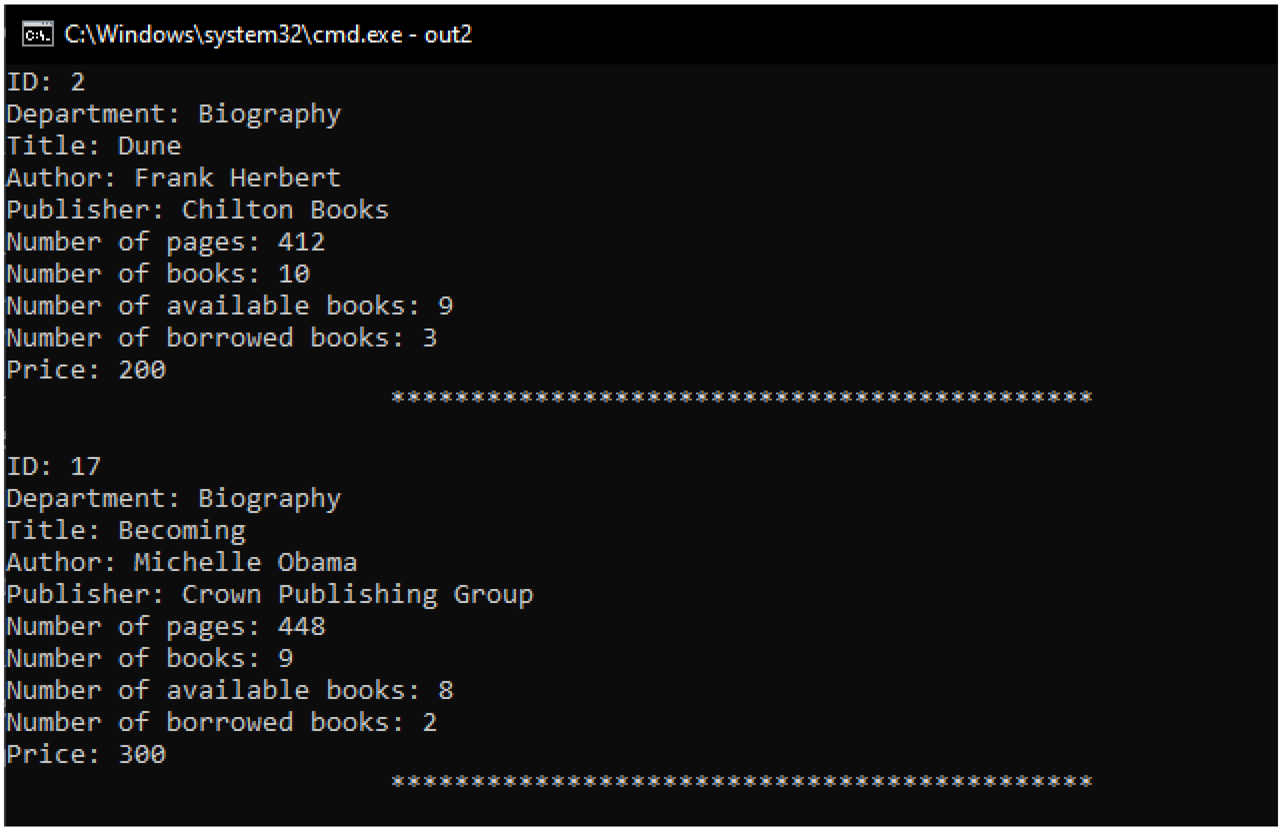
It takes one parameter:

struct Node head: a pointer to the pointer to the head of the list

The function iterates through the list and prints out the details of each issued book int the mentioned department by the user, such as ID, department, title, author, etc. It also prints a blank line after each book for readability.



```
C:\Windows\system32\cmd.exe - out2

              Please Enter The Department: Biography_
```



```
C:\Windows\system32\cmd.exe - out2
ID: 2
Department: Biography
Title: Dune
Author: Frank Herbert
Publisher: Chilton Books
Number of pages: 412
Number of books: 10
Number of available books: 9
Number of borrowed books: 3
Price: 200
                        ********************************************

ID: 17
Department: Biography
Title: Becoming
Author: Michelle Obama
Publisher: Crown Publishing Group
Number of pages: 448
Number of books: 9
Number of available books: 8
Number of borrowed books: 2
Price: 300
                        ********************************************
```

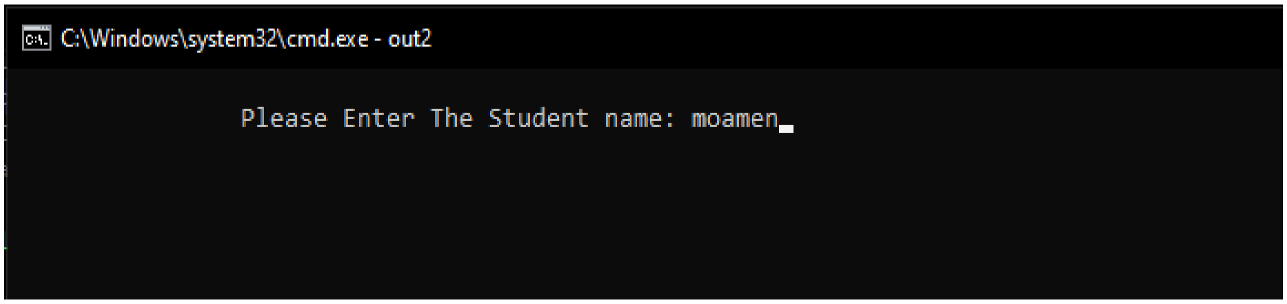
15) void studentissuerecord (struct Student_Node** student_head)

This function will keep record of the student to whom the book has been issued.

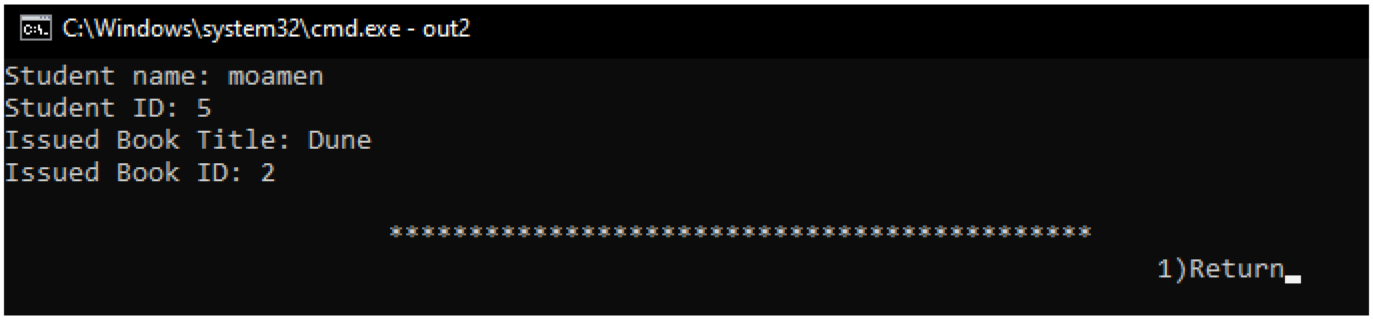
It takes one parameter:

struct Student_Node** student_head

The function askes the user for student name or ID and iterates through the list and prints out the details of student issued book name and ID.



```
C:\Windows\system32\cmd.exe - out2

              Please Enter The Student name: moamen
```



```
C:\Windows\system32\cmd.exe - out2
Student name: moamen
Student ID: 5
Issued Book Title: Dune
Issued Book ID: 2

                        ********************************************
                                                                        1)Return
```