UNIVERSITI TEKNOLOGI MALAYSIA (UTM), CAMPUS SKUDAI

SECJ2013

DATA STRUCTURE AND ALGORITHM

PROJECT

PREPARED BY (PAS):

NOOR ALIYAH BINTI AHMAD SAUFI (A22EC0234)

SERI NUR AYUNI BINTI SALIMI (A22EC0268)

PUTERI NURIN I'RDINA BINTI FADZIL (A22EC0261)

SECTION:

SECJ2013-04

LECTURER'S NAME:

DR LIZAWATI BINTI MI YUSUF

DATE OF SUBMISSION:

16th JANUARY 2024

GITHUB SOURCE CODE SUBMISSION LINK:

https://github.com/jjn7702/SECJ2013-DSA/blob/main/Submission/sec04/PAS/Project/files/source%20code/project.cpp

Problem analysis

1. **Objectives**

We have outlined a few objectives for this project, which are crucial in order to provide methods for a librarian to execute a task more efficiently and effectively. In this program, a librarian is able:

- To view and manage all books in the library
- To add informations of a new book or remove a book from the library
- To check the amount of books in the library
- To facilitate the borrowing and returning of books

2. **Project Overview**

For this project, we developed a library management system that allows the librarian to manage the books in the library such as adding a new book or delete a book from the library system. Other than that, this system allows the librarian to insert information about borrowing and returning a book process. For example, if a person wants to borrow a book from the library, the librarian will insert the person's name and book's details to change the status of that book to not available in the library (borrowed). Then, if that person wants to return the book, the librarian will delete his/her borrowing information from the system and the status of the book is changed to available to borrow.

We implemented two operations of data structure and algorithm to achieve our goals. First, we used a stack-linked list operation to allow the insertion and deletion of a book while we used a queue to handle the borrowing and returning of books.

We have 3 classes in this system, class Node, class Stack and class Queue. The Node class holds a few private attributes which are book id, name, author, genre and year of publication. For public methods, this class has an object named next, a constructor to initialize the local attributes, accessor methods to return the value of the attributes and display() function to print all attributes of a book.

Next, we have a Stack class in which we have an object named top as its private attributes. In this class, we have several public methods to implement the stack operation in our system such as push, pop and stackTop. We also define a constructor to initialize the top value to null, isEmpty() function to check whether the stack is empty or not and display() function to print all data in the stack.

Our last class is named Queue. For this class, we define a few attributes which are front, back and count. We also have a name attribute to represent the borrower's name and a class Node object named b represents a book. We have the basic Queue functions such as enQueue to insert new data in the queue, deQueue to remove data and deQueue(int index) to remove data at a specific position in the queue. In addition, we also define the constructor, isEmpty() and isFull() functions to help the process of queue operations in this system.

Design

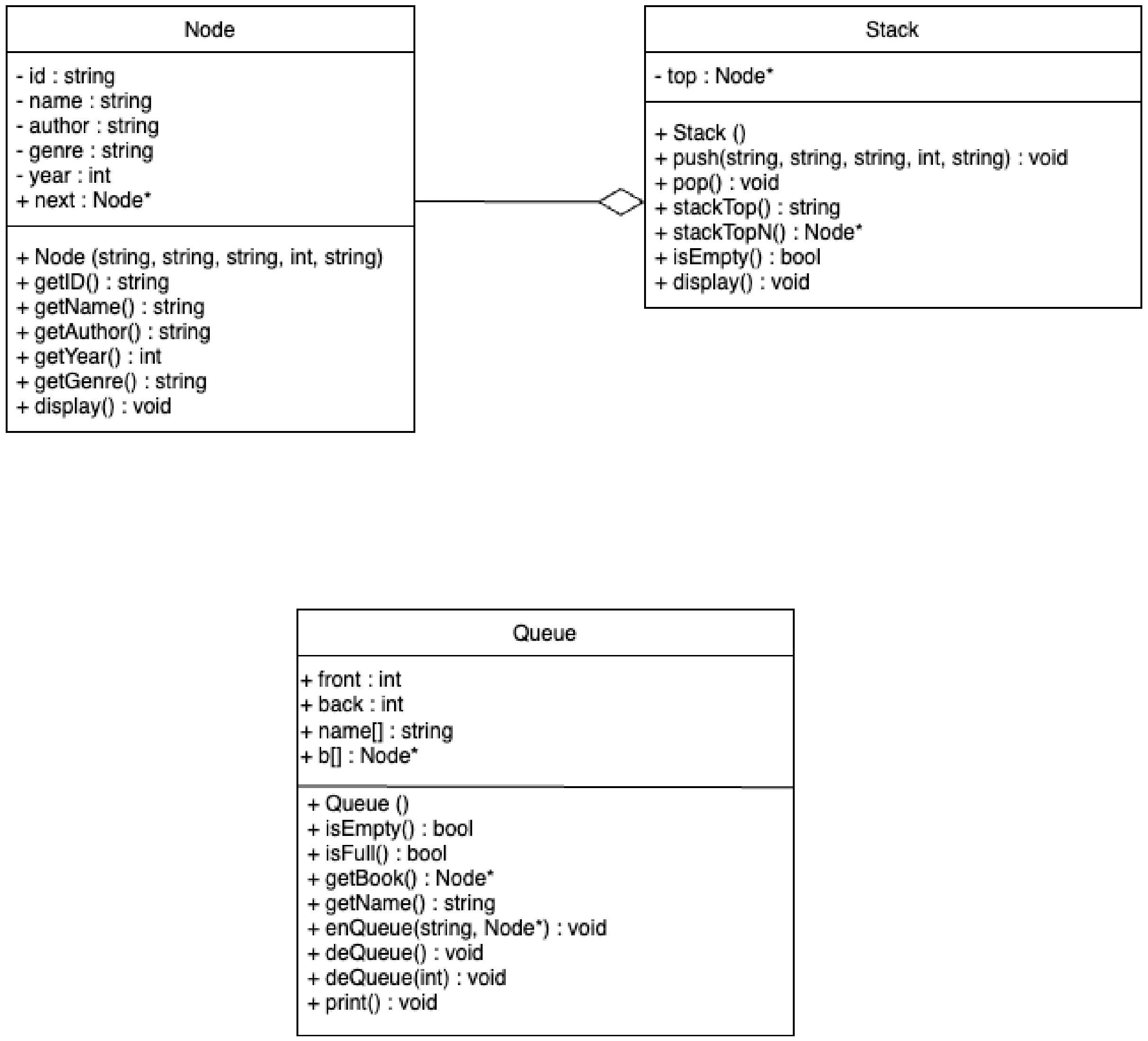


Figure 1: Overall Class Diagram for the Software Application

For the software application, we implemented three different classes which are Node, Stack and Queue. Each of these classes serve different functions and purposes for the software application which will be broken down one by one.

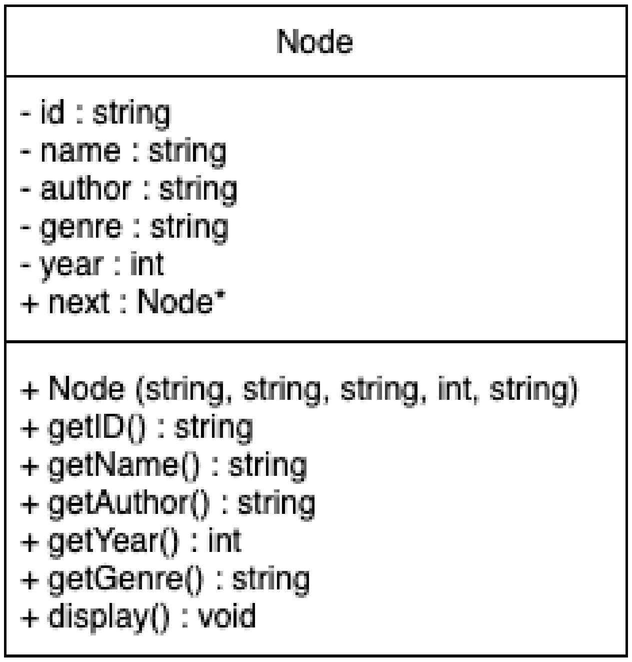


Figure 2: Node Class Diagram for the Software Application

The above figure shows the first implementation of a class called Node. It stores several private attributes such as book id, name, author, genre, and year. It also stores a public attribute which is a next pointer. Additionally, this class also implements a constructor, accessors for each of the private attributes, as well as a display() function for printing purposes. The main purpose of this class is to serve as a linked list implementation for the next class which is Stack.

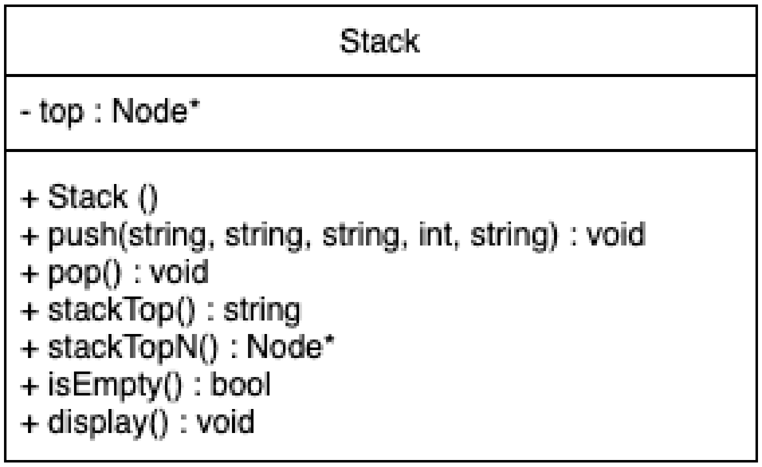


Figure 3: Stack Class Diagram for the Software Application

Apart from that, we have also implemented Stack class in our software application. It only stores one private attribute which is a top pointer from the Node class. Just like Node class, it also implements a constructor and a display function. Plus, it implements a push() function to push book data into the stack, meanwhile pop() function is implemented to pop or remove the top data in the stack. As of stackTop(), the purpose of this function is to return the top book data in the stack meanwhile stackTopN() returns the top pointer from the Node class as it employs the Last-In-First-Out concept.. Lastly, the purpose of isEmpty() function is to return a boolean value to determine whether or not the stack is empty.

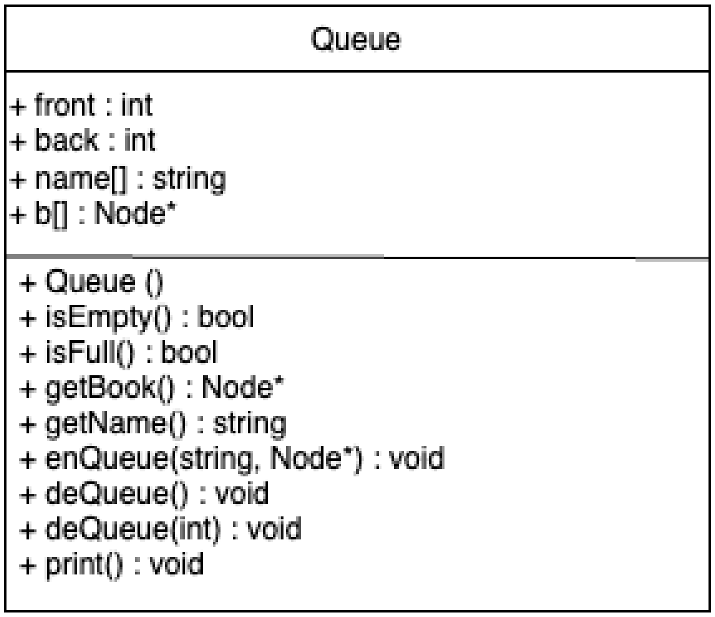


Figure 4: Queue Class Diagram for the Software Application

For the last class, which is Queue, it stores multiple public attributes in its class which are front, back, name array and array of pointers named b. Just like the two aforementioned classes, Queue implements a constructor as well as a display function named print(). Same as the Stack class, Queue also implements an isEmpty() function to return a boolean value to determine whether the queue is empty. It also implements an isFull() function to return a boolean value to determine whether the queue is full. This class implements two different accessors which are getBook() to return the array of pointers, b[] from the class Node, and getName() to return the name of borrower. The Queue class also implements an enQueue() function to add data into the queue and a deQueue() function to remove the first data inside the queue, as it employs the First-In-First-Out concept. Lastly, the class also implements another deQueue() function that removes data at a specific position.

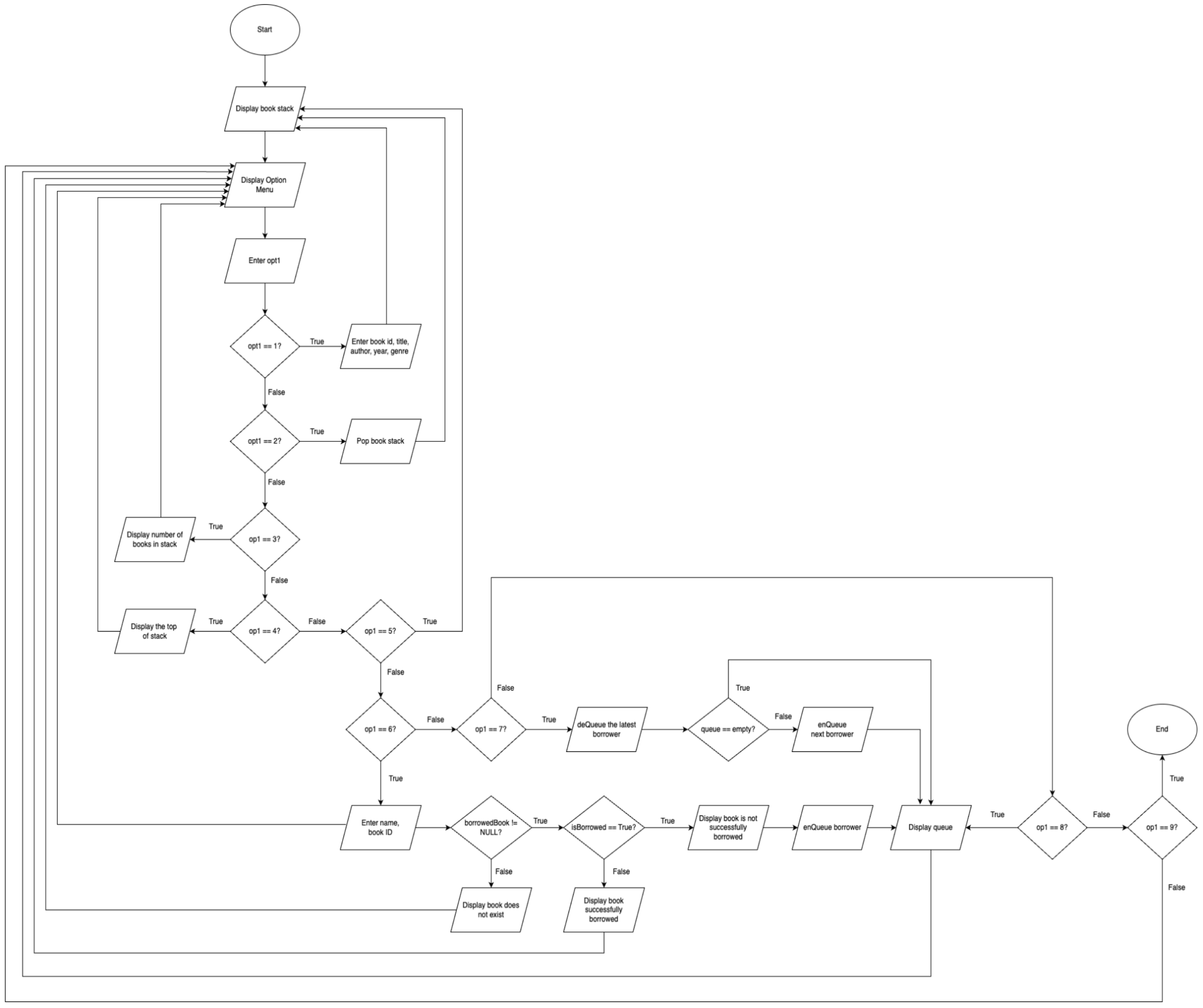


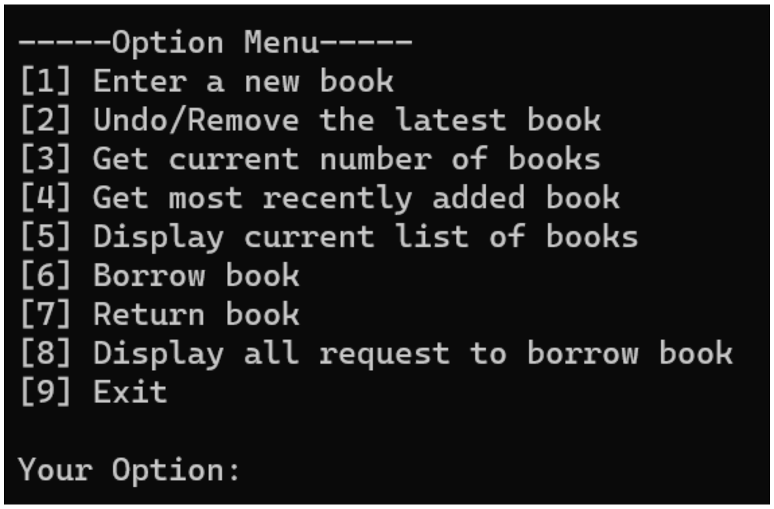
Figure 5: Flowchart for the Software Application Operation

The above diagram illustrates the flow of processes that the software application carries out. It employs multiple different condition statements and loops in order to execute different operations based on different inputs to achieve the desired outputs. The software application carries out different operations such as inserting or removing books into a stack in the library, and allowing books borrowing via queue operations. Each step of the operation is crucial in order to produce the correct output efficiently.

Data structure concept implementation

In this project, we implemented 2 different data structures learnt in the Data Structure and Algorithm course throughout this semester which are stack and queue. The Library Management System is designed to manage and view the library's books that have attributes of book identification, title, author's name, genre and year of publication. Stack data structure is implemented in this system to store book information that is read from an input file, to store the book information entered by the user and to get the details of the most recently added book. On the other hand, Queue data structure is implemented to store all the customer's name that are borrowing books from the library, to store the customer's name that are waiting to borrow books that are currently unavailable, to remove the name of the borrower that has returned the borrowed book and lastly to remove the name of the customer that queued for a book that is previously unavailable and enter the customer's name and the details of the borrowed book into the 'borrower' queue.

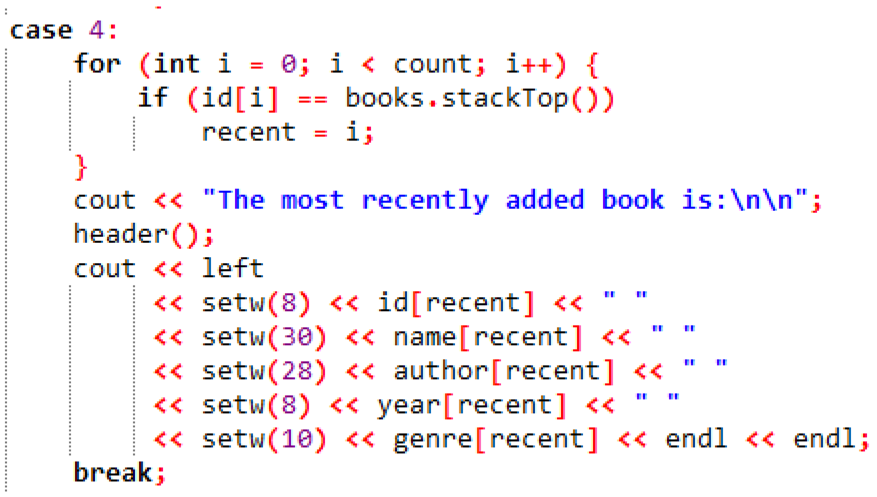
Book information is stored in a stack. For the implementation of stack, 2 classes are written in the code which are class Node and class Stack. In this project, the program reads the book details from an input file named 'inp.txt'. After reading each book's details the system will then push the book information into the stack by using push() function to store them in the stack. Other than reading book information from the input file, our program accepts user input to add new books into the system. The user can enter the book's details such as the book id, title, author, genre, and year of publication. Once the information is entered, it will be pushed into the stack. Switch case is implemented in our program where the switch expression is the user's option from an option menu as shown in figure 6 below.



```
-----Option Menu-----
[1] Enter a new book
[2] Undo/Remove the latest book
[3] Get current number of books
[4] Get most recently added book
[5] Display current list of books
[6] Borrow book
[7] Return book
[8] Display all request to borrow book
[9] Exit

Your Option:
```

Figure 6: Option menu

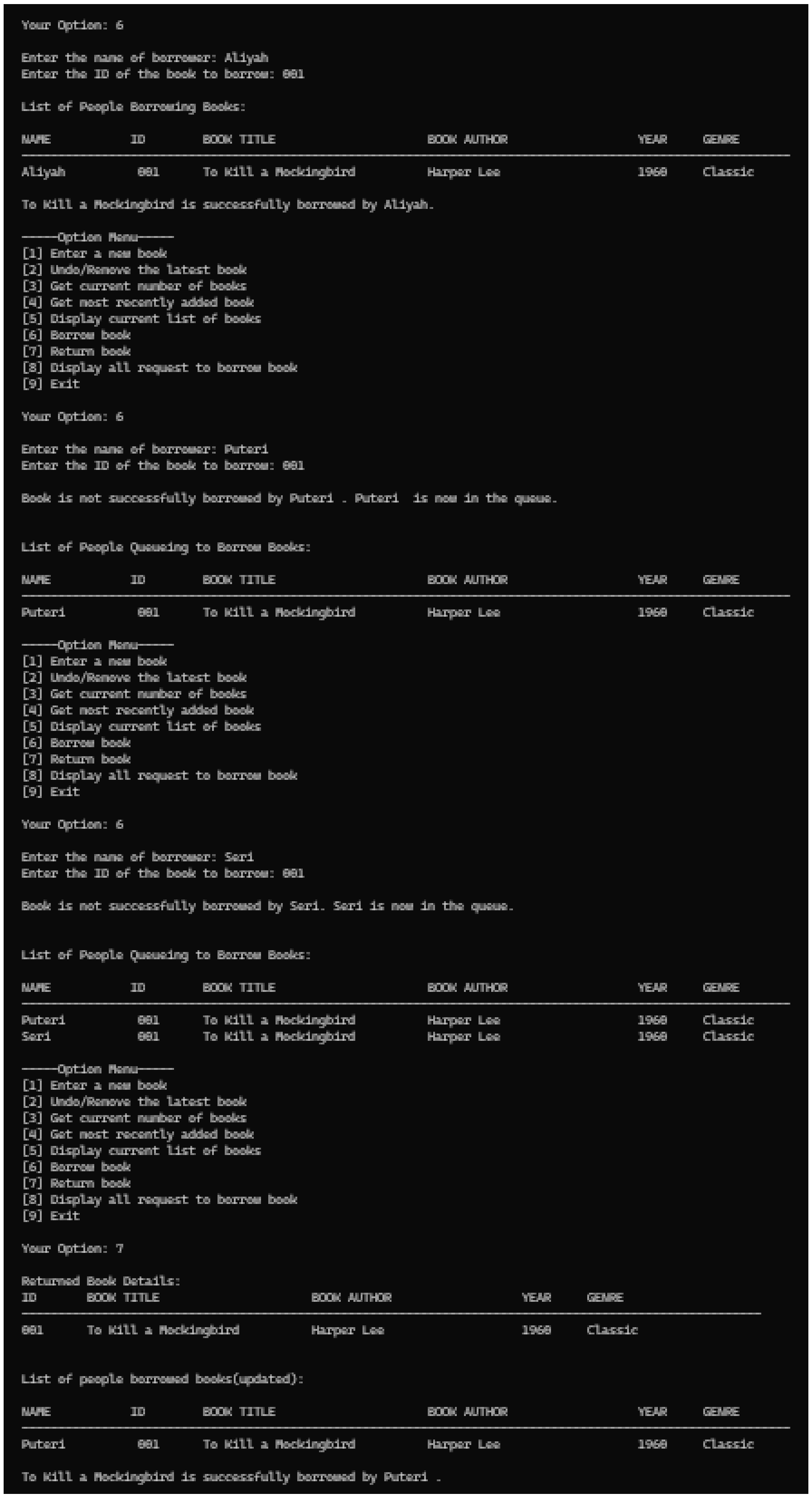
When the user enters option 1, the system will ask for the book's details to be pushed into the stack as explained above. Next, pop() function is used to delete the latest book in the stack. When the user chooses the second option in the option menu, the switch case in the main() function will call the pop() function in class Stack to delete the latest book in the stack. Other than that, The stackTop() function returns the id of the book at the top of the stack. When the user picks option 4 from the menu, the program uses the stackTop() function from class Stack to get the book information stored at the top of the stack. This can be clearly shown by the code segment below.



```
case 4:
    for (int i = 0; i < count; i++) {
        if (id[i] == books.stackTop())
            recent = i;
    }
    cout << "The most recently added book is:\n\n";
    header();
    cout << left
         << setw(8) << id[recent] << " "
         << setw(30) << name[recent] << " "
         << setw(28) << author[recent] << " "
         << setw(8) << year[recent] << " "
         << setw(10) << genre[recent] << endl << endl;
    break;
```

Figure 7: stackTop() function call

For the implementation of queue data structure, we implemented class Queue and 2 objects for the class queue are implemented in the main() function, one named ‘borrow’ and another named ‘request’. The ‘borrow’ queue is used to store the borrower’s name into the queue. When the user chooses option number 6, the system will ask for the name of the borrower and the book id of the book to be borrowed. Based on the book id, if the book is available, the customer will get to borrow the book and the customer's name is put into the ‘borrow’ queue using the enQueue() function that is called in the main() function from the class Queue. On the other hand, if the book is not available, by using the enQueue() function, the customer name is put into the ‘request’ queue. When user enters option number 7, the first borrowed book in the ‘borrow’ queue is returned to the returnedBook object of class Node using the getBook() function from the class queue. Then, deQueue() function is implemented in the main() function to delete the book that has been returned from the ‘borrow’ queue. If there are people requesting for the recently returned book in the ‘request’ queue, the book returned will automatically be given to the first person in the ‘request’ queue. This can be clearly shown in the output below.



```
Your Option: 6

Enter the name of borrower: Aliyah
Enter the ID of the book to borrow: 001

List of People Borrowing Books:

NAME           ID        BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
---------------------------------------------------------------------------------------------------------
Aliyah          001      To Kill a Mockingbird          Harper Lee                   1960     Classic

To Kill a Mockingbird is successfully borrowed by Aliyah.

------Option Menu------
[1] Enter a new book
[2] Undo/Remove the latest book
[3] Get current number of books
[4] Get most recently added book
[5] Display current list of books
[6] Borrow book
[7] Return book
[8] Display all request to borrow book
[9] Exit

Your Option: 6

Enter the name of borrower: Puteri
Enter the ID of the book to borrow: 001

Book is not successfully borrowed by Puteri . Puteri  is now in the queue.


List of People Queueing to Borrow Books:

NAME           ID        BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
---------------------------------------------------------------------------------------------------------
Puteri          001      To Kill a Mockingbird          Harper Lee                   1960     Classic

------Option Menu------
[1] Enter a new book
[2] Undo/Remove the latest book
[3] Get current number of books
[4] Get most recently added book
[5] Display current list of books
[6] Borrow book
[7] Return book
[8] Display all request to borrow book
[9] Exit

Your Option: 6

Enter the name of borrower: Seri
Enter the ID of the book to borrow: 001

Book is not successfully borrowed by Seri. Seri is now in the queue.


List of People Queueing to Borrow Books:

NAME           ID        BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
---------------------------------------------------------------------------------------------------------
Puteri          001      To Kill a Mockingbird          Harper Lee                   1960     Classic
Seri            001      To Kill a Mockingbird          Harper Lee                   1960     Classic

------Option Menu------
[1] Enter a new book
[2] Undo/Remove the latest book
[3] Get current number of books
[4] Get most recently added book
[5] Display current list of books
[6] Borrow book
[7] Return book
[8] Display all request to borrow book
[9] Exit

Your Option: 7

Returned Book Details:
ID       BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
-----------------------------------------------------------------------------------------------------
001      To Kill a Mockingbird          Harper Lee                   1960     Classic


List of people borrowed books(updated):

NAME           ID        BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
---------------------------------------------------------------------------------------------------------
Puteri          001      To Kill a Mockingbird          Harper Lee                   1960     Classic

To Kill a Mockingbird is successfully borrowed by Puteri .
```

Figure 8: Output example

Source Code

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <iomanip>
#define N 100
using namespace std;
void header() {
  //cout << "\nList of Books in the Library:\n" << endl;
  cout << left
      << setw(9) << "ID"
      << setw(31) << "BOOK TITLE"
      << setw(29) << "BOOK AUTHOR"
      << setw(9) << "YEAR"
      << setw(8) << "GENRE"
      << endl
      << "----------------------------------------------------------------------------------------------------"
      << endl;
}
void headerQ() {
  //cout << "\nList of Books in the Library:\n" << endl;
  cout << left
      << setw(15) << "NAME"
      << setw(10) << "ID"
      << setw(31) << "BOOK TITLE"
      << setw(29) << "BOOK AUTHOR"
      << setw(9) << "YEAR"
      << setw(8) << "GENRE"
      << endl
      << "--------------------------------------------------------------------------------------------------------"
      << endl;
}
class Node {
  string id, name, author, genre;
  int year;
public:
  Node *next;
  Node(string _id, string _name, string _author, int _year, string _genre) {
    id = _id;
    name = _name;
    author = _author;
    year = _year;
    genre = _genre;
    next = NULL;
  }
  string getID() {
    return id;
  }
  string getName() {
    return name;
  }
```

```
    string getAuthor() {
        return author;
    }
    int getYear() {
        return year;
    }
    string getGenre() {
        return genre;
    }
    void display() {
        cout << left
             << setw(8) << id << " "
             << setw(30) << name << " "
             << setw(28) << author << " "
             << setw(8) << year << " "
             << setw(10) << genre << endl;
    }
};
class Stack {
    Node *top;
public:
    Stack() {
        top = NULL;
    }
    void push(string id, string name, string author, int year, string genre) {
        Node *nn = new Node(id, name, author, year, genre);
                nn->next = top;
        top = nn;
    }
    void pop() {
        if (isEmpty())
            cout << "The stack is empty\n";
        else {
            Node *temp = top;
            top = temp->next;
            temp->next = NULL;
            delete temp;
        }
    }
    string stackTop() {
        return top->getID();
    }
    Node *stackTopN() {
                return top;
    }
    bool isEmpty() {
        return top == NULL;
    }
    void display() {
                cout << "\nList of Books in the Library:\n" << endl;
        header();
        Node *temp = top;
        while (temp != NULL) {
            temp->display();
            temp = temp->next;
        }
        cout << endl;
    }
};
class Queue {
    public:
            int front, back, count;
            string name[N];
        Node* b[N];

        Queue() {
            front = 0;
            back = -1;
            count = 0;
        }
        bool isEmpty() {
            return (count == 0);
        }
        bool isFull() {
            return (count == N);
        }
                    Node* getBook() {
                            return b[front];
                    }
                    string getName(int idx) {
                            return name[idx];
                    }
        void enQueue(string n, Node* bb) {
            if (isFull())
                cout << "\tCannot insert. Queue is full!" << endl;
            else {
                back++;
                count++;
                name[back] = n;
                b[back] = bb;
            }
        }
        void deQueue() {
            if (isEmpty())
                cout << "\tCannot remove. Queue is empty!" << endl;
            else {
                front++;
                count--;
            }
        }
        void deQueue(int index) {
                for (int i = index; i < back; i++) {
                        name[i] = name[i+1];
                        b[i] = b[i+1];
                }
                back--;
                count--;
        }
        void print() {
            if (isEmpty())
                cout << "Sorry, the queue is empty" << endl;
            else {
                headerQ();
                for (int i = front; i <= back; i++) {
                    cout << left << setw(15) << name[i] << " " << setw(5);
                    b[i]->display();
                }
                cout << endl;
            }
        }
};
int mainMenu() {
    int opt;
    cout << "-----Option Menu-----\n"
         << "[1] Enter a new book\n"
         << "[2] Undo/Remove the latest book\n"
         << "[3] Get current number of books\n"
         << "[4] Get most recently added book\n"
         << "[5] Display current list of books\n"
         << "[6] Borrow book\n"
         << "[7] Return book\n"
         << "[8] Display all request to borrow book\n"
         << "[9] Exit\n"
         << "\nYour Option: ";
    cin >> opt;

    return opt;
}
int main() {
    string id[N], name[N], author[N], genre[N];
    int year[N], count = 0;
    Stack books;
    string id_, title_, author_, genre_;
    int year_, recent, removed;

                Queue borrow;
    Queue request;
    string borrower;

    ifstream inp ("inp.txt");

    for (int i = 0; i < 10; i++) {
        getline(inp, id[i], ',');
        getline(inp, name[i], ',');
        getline(inp, author[i], ',');
        inp >> year[i];
        inp.ignore();
        getline(inp, genre[i]);
        books.push(id[i], name[i], author[i], year[i], genre[i]);
        count++;
    }
    books.display();
```

```
int opt1 = mainMenu();

while(opt1>0 && opt1<10) {
  switch(opt1) {
  case 1: //Add a new book to the library
          cin.ignore();
    cout << "\n-----Insert Book Details-----\n";
                      cout << "ID: ";
                      getline(cin, id_);
                      id[count] = id_;
                      cout << "Book Title: ";
                      getline(cin, title_);
                      name[count] = title_;
                      cout << "Book Author: ";
                      getline(cin, author_);
                      author[count] = author_;
                      cout << "Year: ";
                      cin >> year_;
                      year[count]  = year_;
                      cin.ignore();
                      cout << "Genre: ";
                      getline(cin, genre_);
                      genre[count] = genre_;
                      cout << endl;
                      books.push(id_,title_,author_,year_,genre_);
                      count++;
                      cout << title_ << " is succesfully added into the library.";
                      books.display();
              break;
        case 2: //Delete latest book in the library
          for (int i = 0; i < count; i++) {
                if (id[i] == books.stackTop())
                  removed = i;
              }
              cout << endl << name[removed] << " is succesfully removed into the library.\n";
              books.pop();
              books.display();
              count--;
              break;
  case 3: //Check amount of books in the library
    cout << "There are currently " << count << " books in the library.\n" << endl;
    break;
  case 4: //Display details of most recent book in the library
                for (int i = 0; i < count; i++) {
                if (id[i] == books.stackTop())
                  recent = i;
              }
              cout << "The most recently added book is:\n\n";
              header();
              cout << left
                              << setw(8) << id[recent] << " "
                              << setw(30) << name[recent] << " "
                              << setw(28) << author[recent] << " "
```

```
                                                << setw(8) << year[recent] << " "
                                                << setw(10) << genre[recent] << endl << endl;
                                break;
                case 5: //Display all books available in the library
                    books.display();
                    break;
                case 6: //Borrow book
                    {
                    cout << "\nEnter the name of borrower: ";
                    cin.ignore();
                    getline(cin, borrower);

                    cout << "Enter the ID of the book to borrow: ";
                    string bID;
                    cin >> bID;

                    Node* borrowedBook = NULL;
                    Node* currBook = books.stackTopN();

                    //To find the soecific ID book entered by user
                    while (currBook != NULL) {
                        if ((currBook->getID() == bID)) {
                                borrowedBook = currBook;
                                break;
                        }
                        currBook = currBook->next;
                    }
                    //To check if the book is available to borrow or not
                    if (borrowedBook != NULL) {
                        bool isBorrowed = false;
                        for (int i = borrow.front; i <= borrow.back; i++) {
                                if (borrow.b[i]->getID() == bID) {
                                        isBorrowed = true;
                                        break;
                                }
                        }
                        if (!isBorrowed) { //Add to borrowing queue if book is available
                                                borrow.enQueue(borrower, borrowedBook);
                                                cout << "\nList of People Borrowing Books: \n" << endl;
                                                borrow.print();
                                                cout << borrowedBook->getName() << " is successfully borrowed by " <<
borrower << ".\n\n";
                                }
                                else { //Add to requesting queue if book is currently borrowed by someone else
                                        request.enQueue(borrower, borrowedBook);
                                        cout << "\nBook is not successfully borrowed by " << borrower << ". " << borrower <<
" is now in the queue." << endl << endl;
                                        cout << "\nList of People Queueing to Borrow Books: \n" << endl;
                                                request.print();
                                }
                                }
                                else {
                                        cout << "\nBook does not exist in the library.\n" << endl;
                                }
                    }
            break;

        case 7: //Return book & let next person borrow the returned book (if any)
            {
            Node *returnedBook = borrow.getBook();
            borrow.deQueue();
            cout << "\nReturned Book Details: " << endl;
            header();
            returnedBook->display();
            cout << endl;

            if (!request.isEmpty()) {
                    bool findBook = false;
                    int index = 0;
                    for (int i = request.front; i <= request.back; i++) {
                    if (request.b[i]->getID() == returnedBook->getID()) {
                            findBook = true;
                            index = i;
                            break;
                    }
                }
                if (!findBook) {
                    cout << "\nThe returned book is not in the request queue." << endl;
                }
                else {
                    //Add requested book to borrow that is same with returned book to borrowing queue
                    borrow.enQueue(request.getName(index), request.b[index]);
                    cout << "\nList of people borrowed books(updated): \n" << endl;
                    borrow.print();
                    cout << returnedBook->getName() << " is successfully borrowed by " << request.getName(index)
<< ".\n\n";
                    //Remove the book & borrower from request queue
                    request.deQueue(index);
                }
            }
            else {
                    cout << "\nNo requests in the queue.\n" << endl;
            }
        break;
            }
        case 8: //Display
            cout << "List of People Queueing to Borrow Books: \n" << endl;
            request.print();
            break;
        case 9:
            exit(0);
        }
        opt1 = mainMenu();
    }

            return 0;
}
```

User Manual/Guide:

(Texts in **Blue** and **Bold** are user inputs)

1. **Enter a new book**

   Your Option: **1**

   -----Insert Book Details-----

   ID: **B011**

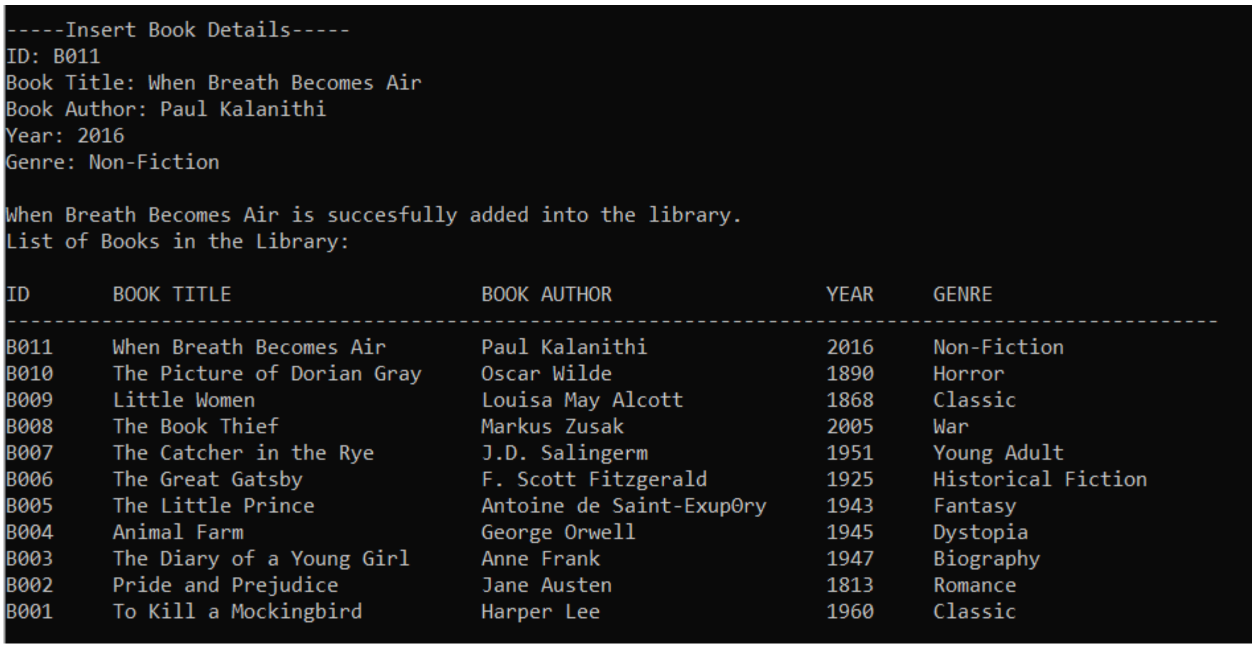
   Book Title: **When Breath Becomes Air**

   Book Author: **Paul Kalanithi**

   Year: **2016**

   Genre: **Non-Fiction**

   Example Output:

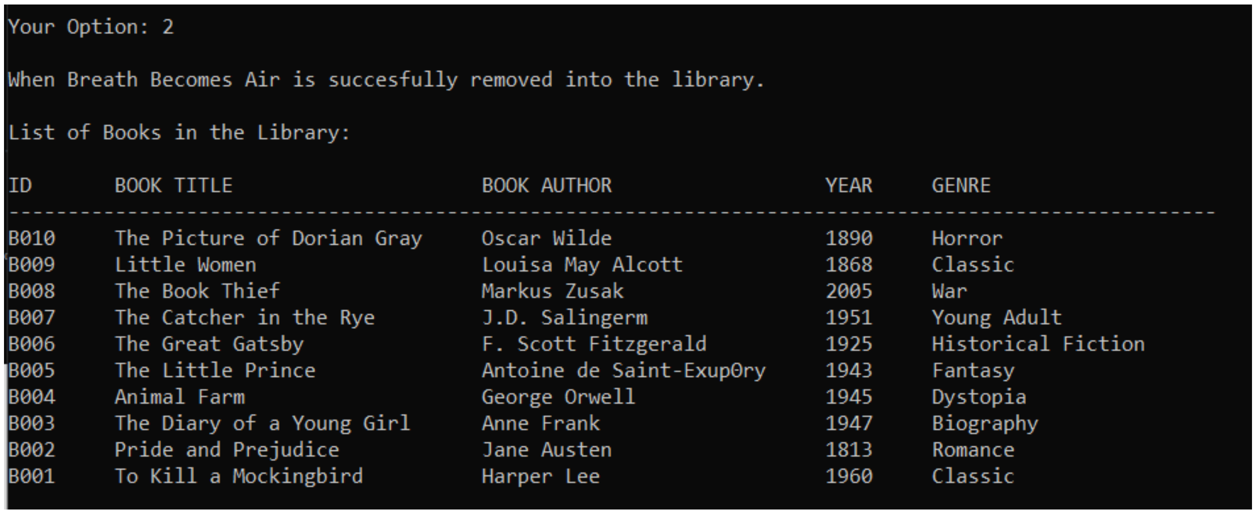


```
-----Insert Book Details-----
ID: B011
Book Title: When Breath Becomes Air
Book Author: Paul Kalanithi
Year: 2016
Genre: Non-Fiction

When Breath Becomes Air is succesfully added into the library.
List of Books in the Library:

ID       BOOK TITLE                     BOOK AUTHOR                 YEAR     GENRE
--------------------------------------------------------------------------------------------------------
B011     When Breath Becomes Air        Paul Kalanithi              2016     Non-Fiction
B010     The Picture of Dorian Gray     Oscar Wilde                 1890     Horror
B009     Little Women                   Louisa May Alcott           1868     Classic
B008     The Book Thief                 Markus Zusak                2005     War
B007     The Catcher in the Rye         J.D. Salingerm              1951     Young Adult
B006     The Great Gatsby               F. Scott Fitzgerald         1925     Historical Fiction
B005     The Little Prince              Antoine de Saint-Exup0ry    1943     Fantasy
B004     Animal Farm                    George Orwell               1945     Dystopia
B003     The Diary of a Young Girl      Anne Frank                  1947     Biography
B002     Pride and Prejudice            Jane Austen                 1813     Romance
B001     To Kill a Mockingbird          Harper Lee                  1960     Classic
```

2. **Undo/Remove the latest book**

   Your Option: **2**

Example Output:

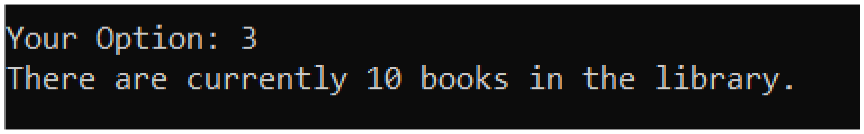


```
Your Option: 2

When Breath Becomes Air is succesfully removed into the library.

List of Books in the Library:

ID       BOOK TITLE                    BOOK AUTHOR                  YEAR     GENRE
----------------------------------------------------------------------------------------------------
B010     The Picture of Dorian Gray    Oscar Wilde                  1890     Horror
B009     Little Women                  Louisa May Alcott            1868     Classic
B008     The Book Thief                Markus Zusak                 2005     War
B007     The Catcher in the Rye        J.D. Salingerm               1951     Young Adult
B006     The Great Gatsby              F. Scott Fitzgerald          1925     Historical Fiction
B005     The Little Prince             Antoine de Saint-Exup0ry     1943     Fantasy
B004     Animal Farm                   George Orwell                1945     Dystopia
B003     The Diary of a Young Girl     Anne Frank                   1947     Biography
B002     Pride and Prejudice           Jane Austen                  1813     Romance
B001     To Kill a Mockingbird         Harper Lee                   1960     Classic
```

3. **Get Current Number of books**

   Your Option: **3**

Example Output:

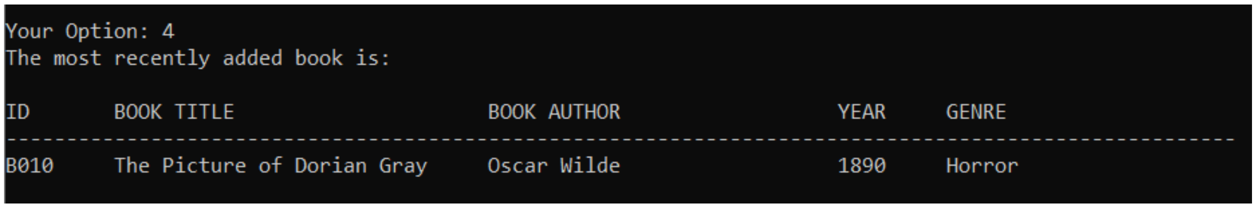


```
Your Option: 3
There are currently 10 books in the library.
```

4. **Get most recently added book**

   Your Option: **4**

Example Output:

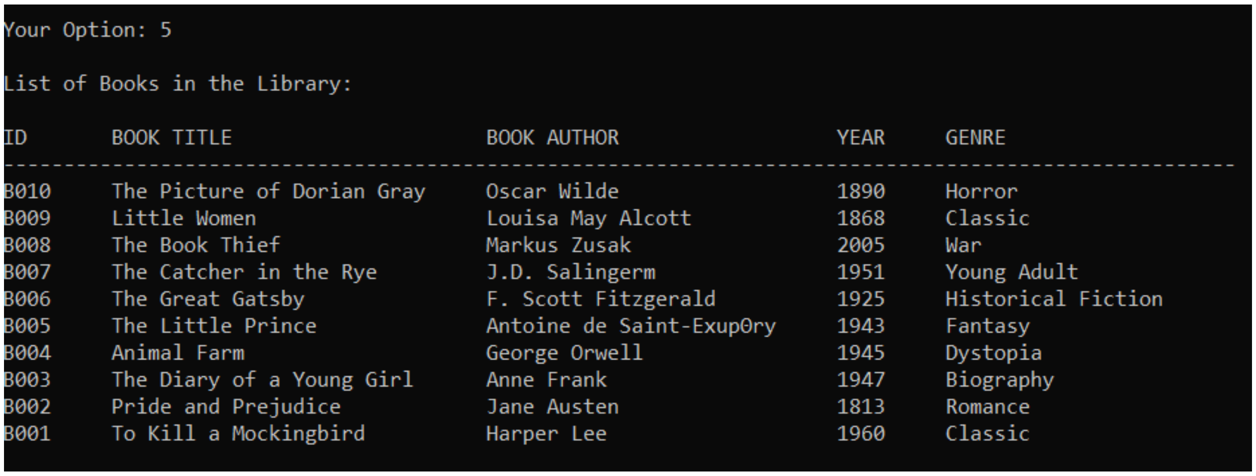


```
Your Option: 4
The most recently added book is:

ID       BOOK TITLE                    BOOK AUTHOR                  YEAR     GENRE
----------------------------------------------------------------------------------------------------
B010     The Picture of Dorian Gray    Oscar Wilde                  1890     Horror
```

5. **Display current list of books**

   Your Option: **5**

Example Output:



```
Your Option: 5

List of Books in the Library:

ID       BOOK TITLE                     BOOK AUTHOR                YEAR     GENRE
------------------------------------------------------------------------------------------------------
B010     The Picture of Dorian Gray     Oscar Wilde                1890     Horror
B009     Little Women                   Louisa May Alcott          1868     Classic
B008     The Book Thief                 Markus Zusak               2005     War
B007     The Catcher in the Rye         J.D. Salingerm             1951     Young Adult
B006     The Great Gatsby               F. Scott Fitzgerald        1925     Historical Fiction
B005     The Little Prince              Antoine de Saint-Exup0ry   1943     Fantasy
B004     Animal Farm                    George Orwell              1945     Dystopia
B003     The Diary of a Young Girl      Anne Frank                 1947     Biography
B002     Pride and Prejudice            Jane Austen                1813     Romance
B001     To Kill a Mockingbird          Harper Lee                 1960     Classic
```

6. **Borrow Book**

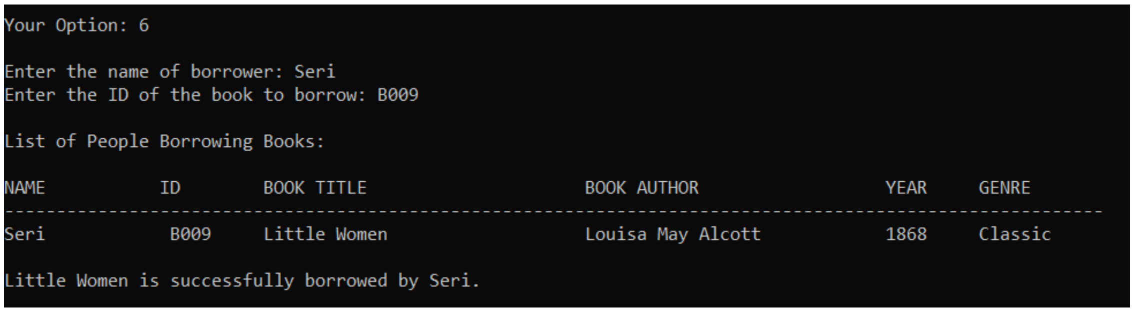
   Your Option: **6**

   a. **If the book is available to borrow**

      Enter the name of borrower: **Seri**

      Enter the ID of the book to borrow**: B009**

      Example Output:



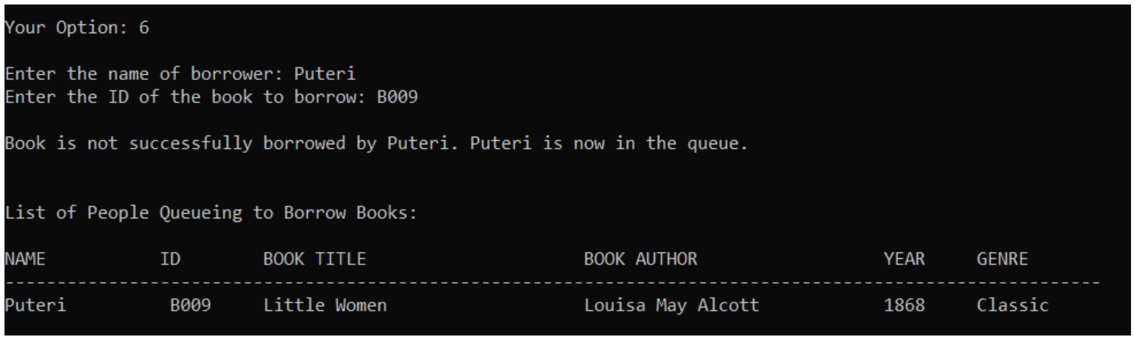
```
Your Option: 6

Enter the name of borrower: Seri
Enter the ID of the book to borrow: B009

List of People Borrowing Books:

NAME           ID        BOOK TITLE                     BOOK AUTHOR                YEAR     GENRE
---------------------------------------------------------------------------------------------------------
Seri            B009     Little Women                   Louisa May Alcott          1868     Classic

Little Women is successfully borrowed by Seri.
```

   b. **If the book is already borrowed by someone else**

      Enter the name of borrower: **Puteri**

      Enter the ID of the book to borrow**: B009**

Example Output:



```
Your Option: 6

Enter the name of borrower: Puteri
Enter the ID of the book to borrow: B009

Book is not successfully borrowed by Puteri. Puteri is now in the queue.


List of People Queueing to Borrow Books:

NAME          ID        BOOK TITLE                     BOOK AUTHOR                 YEAR     GENRE
---------------------------------------------------------------------------------------------------
Puteri         B009      Little Women                   Louisa May Alcott           1868     Classic
```

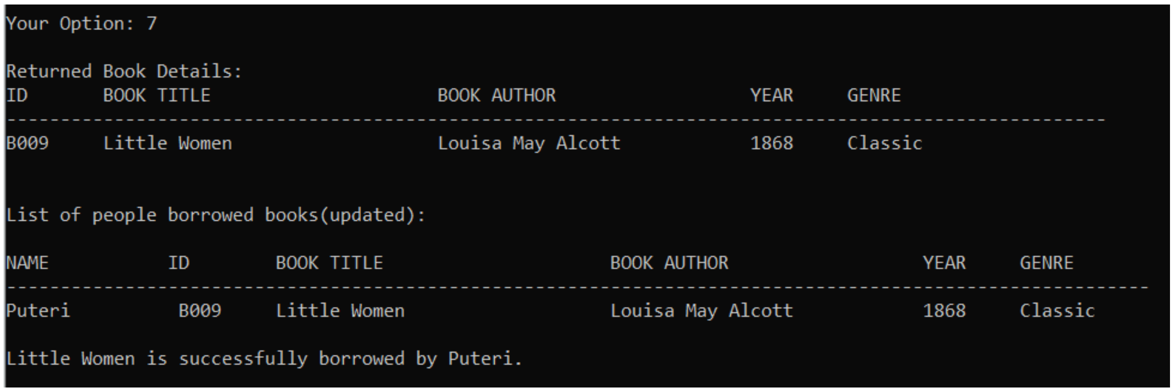
7. **Return Book**

   Your Option: 7

   a. **If there is the next person in request queue that wants to borrow the returned book**

      Example situation: Seri returned book B009 and Puteri is in the request queue to borrow book B009

      Example Output:

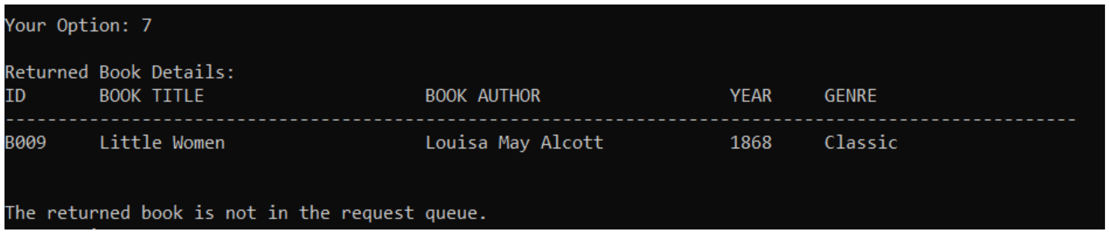


```
Your Option: 7

Returned Book Details:
ID       BOOK TITLE                     BOOK AUTHOR                 YEAR     GENRE
-------------------------------------------------------------------------------------
B009     Little Women                   Louisa May Alcott           1868     Classic


List of people borrowed books(updated):

NAME          ID        BOOK TITLE                     BOOK AUTHOR                 YEAR     GENRE
---------------------------------------------------------------------------------------------------
Puteri         B009      Little Women                   Louisa May Alcott           1868     Classic

Little Women is successfully borrowed by Puteri.
```

   b. **If there is no person in the request queue that wants to borrow the returned book**

Example situation: Puteri returned book B009 but no one in the request queue wants to borrow book B009.

Example Output:

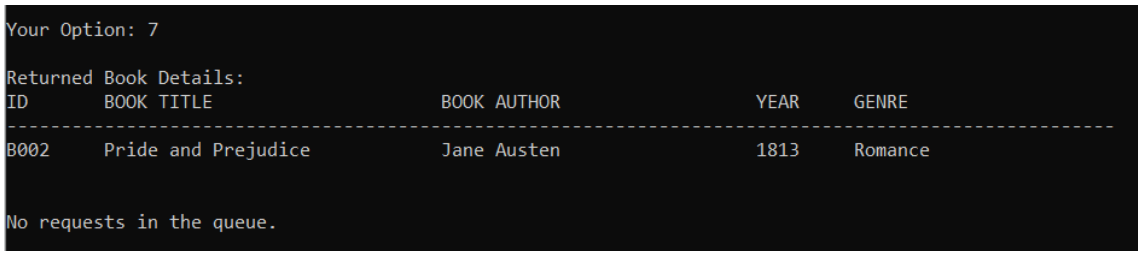


```
Your Option: 7

Returned Book Details:
ID       BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
----------------------------------------------------------------------------------------------------
B009     Little Women                   Louisa May Alcott            1868     Classic


The returned book is not in the request queue.
```

c. **If the request to borrow books queue is empty**

Example situation: Aliyah returned book B002 but the request queue is empty.

Example Output:



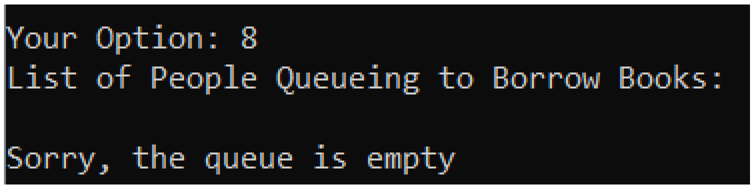
```
Your Option: 7

Returned Book Details:
ID       BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
----------------------------------------------------------------------------------------------------
B002     Pride and Prejudice            Jane Austen                  1813     Romance


No requests in the queue.
```

8. **Display all request to borrow book**

   Your Option: **8**

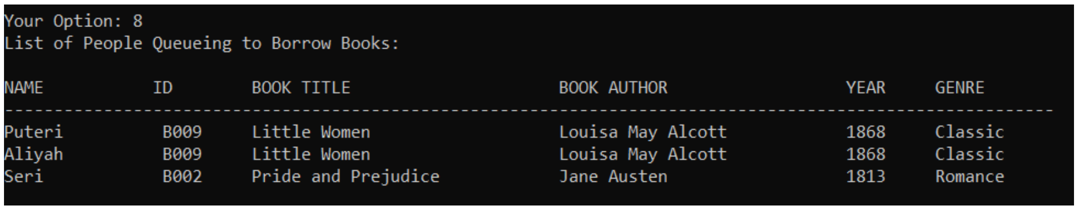
   Example Output:

   a. **If empty**



```
Your Option: 8
List of People Queueing to Borrow Books:

Sorry, the queue is empty
```

b. If not empty



```
Your Option: 8
List of People Queueing to Borrow Books:

NAME           ID        BOOK TITLE                     BOOK AUTHOR                  YEAR     GENRE
----------------------------------------------------------------------------------------------------------
Puteri          B009     Little Women                   Louisa May Alcott            1868     Classic
Aliyah          B009     Little Women                   Louisa May Alcott            1868     Classic
Seri            B002     Pride and Prejudice            Jane Austen                  1813     Romance
```

9. Exit

Your Option: **9**

The program will terminate