GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT077-3.5-2-DSTR

DATA STRUCTURES

UC2F1910CS/UC2F1910SE/UC2F1910IS/UC2F1910IS(AP)/

UC2F1910CS(DA)

HAND OUT DATE: 24 JUNE 2020

HAND IN DATE: 04 SEPTEMBER 2020

WEIGHTAGE: 30%

INSTRUCTIONS TO CANDIDATES:

1. **Submit your assignment at the administrative counter.**
2. **Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).**
3. **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**
4. **Cases of plagiarism will be penalized.**
5. **The assignment should be bound in an appropriate style (comb bound or stapled).**
6. **Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**
7. **You must obtain 50% overall to pass this module.**

GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT077-3.5-2-DSTR

DATA STRUCTURES

UC2F1910CS/UC2F1910SE/UC2F1910IS/UC2F1910IS(AP)/

UC2F1910CS(DA)

HAND OUT DATE: 24 JUNE 2020

HAND IN DATE: 04 SETEPMBER 2020

WEIGHTAGE: 30%

PART B: FULL DOCUMENTATION

NAME	STUDENT ID
HEN KIAN JUN	TP047472
LEE WAI YEN	TP044610

Table of Contents

List of Figures

List of Tables

1.0 Introduction

A community library Sri Petaling has been established by Kuala Lumpur City Hall (DBKL) called Anggerik Aranda Community Library (AACL). Anggerik Aranda Community Library (AALC) is a large organization, but unfortunately the organization faced some issues in managing all the library patrons and the books due to lack of a proper library management system. Therefore, Kuala Lumpur City Hall (DBKL) has hired a software development company to develop a library management system (LMS) for Anggerik Aranda Community Library (AALC). After the investigation made by the software development company, the LMS main functionalities have been identified for the LMS, which is to manage the library patrons and the books.

The books in AACL are categorised into two sub-categories, which is Fiction and Non-Fiction. These two categories were also subdivided based on the genres, as shown in Table 1. However, each book in AACL belong to only one category and only one genre. There is possibility of each book title to have more than one copies in the collection. For example, "One Thousand and One Nights" book could have two copies in AACL, but these two books should be identified uniquely in the system. If one copy has been borrowed, the other copy should be available to be borrowed.

Fiction	**Non-Fiction**
Fantasy	Narrative
Science	Biography
Historical	Periodicals
Realistic	Self-Help
Fan	References

Table 1: List of book genres

In terms of library patrons, according to the business process, the LMS is developed in such a way that the system should be able to manage patrons, which include registering patrons, updating the information for particular patrons, and most importantly is books borrowing and returning books functionalities. Each patron in AACL may borrow the most of 3 books at any

given times. By default, the patrons will need to return the borrowed books within 15 days from the borrowing date. The duration however may be extended upon request, but before the 15th days.

The system will be developed using C++ language with the data structure of one dimensional doubly linked list and two dimensional singly linked list, to store library patron information and manage book information as well as book borrow transactions. The following functionalities will be developed for AACL using C++ application. The Table 2 below shown is the functionalities that are proposed to AACL to manage the library books and library patrons as well as the book borrowed transactions.

AACL Library Management System (LMS)		
Books	Book Display	View all available book titles.
	Add Book	Add individual book copy.
	Search book	Search book based on Category, Genre, Title, and availability.
	Update book's information	Update the details of all the books.
	Delete Book	Delete a book based on Book ISBN.
Patrons	Patron Registration	Registering a new patron into the LMS.
	Search Patron	Search for patron based on Name, ID, etc.
	View all Patrons	View all the patrons with active book borrowed.
	Display the last 10 books borrowed	Display the last 10 books borrowed by a patron.
	Update patrons' information	Update the details of all the patrons.
Borrow Books	Add borrow books transaction	Add book borrowed transaction made by a patron. A patron can borrow a maximum of 3 books at any given time.

	Make extension date	Book borrowed extension is possible to be made by a patron, and only one extension can be made for each book borrowed.
	Display the last 10 books borrowed transaction	Print the last 10 books borrowed transaction by all the patrons.
	Display the borrow book lists	Display all the book borrowed transaction by all the patrons.

Table 2: List of functionalities for AACL Library Management System

2.0 Proposed Data Structure

One dimensional doubly linked list and two dimensional singly linked list will be proposed as the data structure into the proposed LMS for AALC. One dimensional doubly linked list will be proposed to store all the library patron information and also the borrow information. Two dimensional singly linked list will be proposed to store book information. The following will be further discussing on one dimensional doubly linked list along with its advantages and also two dimensional singly linked list with its advantages. Justification and explanation are included to state the reason of proposing these two linked lists as the data structure.

2.1 One Dimensional doubly linked list

Doubly linked list is a diversity of the singly linked list (Studytonight.com, 2020) which comprises a collection of consecutively connected records that known as nodes (Arun, 2020). Nodes in the linked list are associated by pointers which serve as the address of the memory location. Furthermore, node also includes a data element and a node pointer for the following node (Thakur, 2017). In each node, it has two lines, which also known as links (Arun, 2020). Usually, the first link will point to the previous node in the link and the second link will point to the next node in the linked list. However, the first node which contain the previous link will point to NULL, likewise, the last node that contains the next node will also point to NULL. These two links allow to traverse from both backward and forward directions within the list, but it requires an additional storing capacity to store the extra links (Studytonight.com, 2020). In terms of the proposed LMS, a requirement stated that the system should be able to print the last 10 books borrowed by a patron. Therefore, doubly list linked list is highly recommended to propose as a proposed data structure to store the borrow list because doubly linked list has a backward traverse function by using ‘tail’ pointer, which allows the print from the behind of a linked list. Doubly linked list is much more efficient compared to singly linked list as when it comes to print the records in descending order, the ‘tail’ pointer will be directly pointed to the last node of the linked list and print the record.

Doubly linked list helps in removing the complexity of insertion and deletion at a known position from O(n) in Singly linked list to O(1) in Doubly linked list.

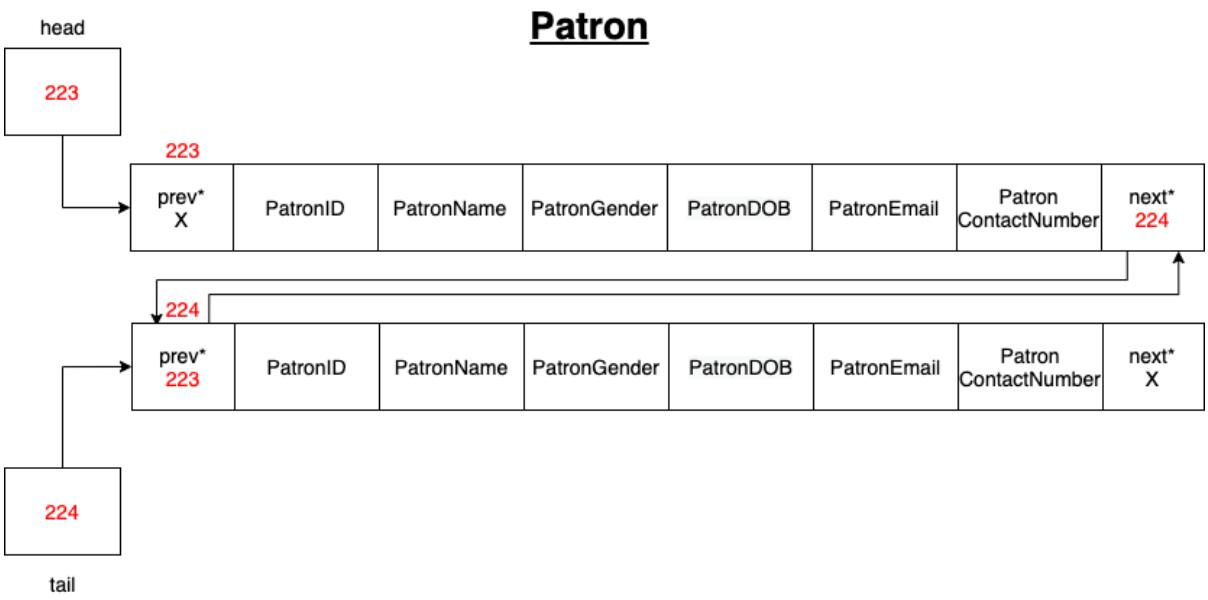


Figure 1: One Dimensional Doubly Linked List for Patron information

The Figure 1 shown above is the example of storing patron information in a doubly linked list. This diagram depicts that the patron can be used to print the records from the beginning or print the records from the behind, because there are two pointers, head and tail, are pointing to the different memory address. However, each node in patron linked list is linked together using its pointers (prev* and next*), so that the patron records can be printed in ascending order or descending order. This doubly linked list is not a circular linked list, therefore the first record of prev pointer and the last record of next pointer will be set to be NULL by default.

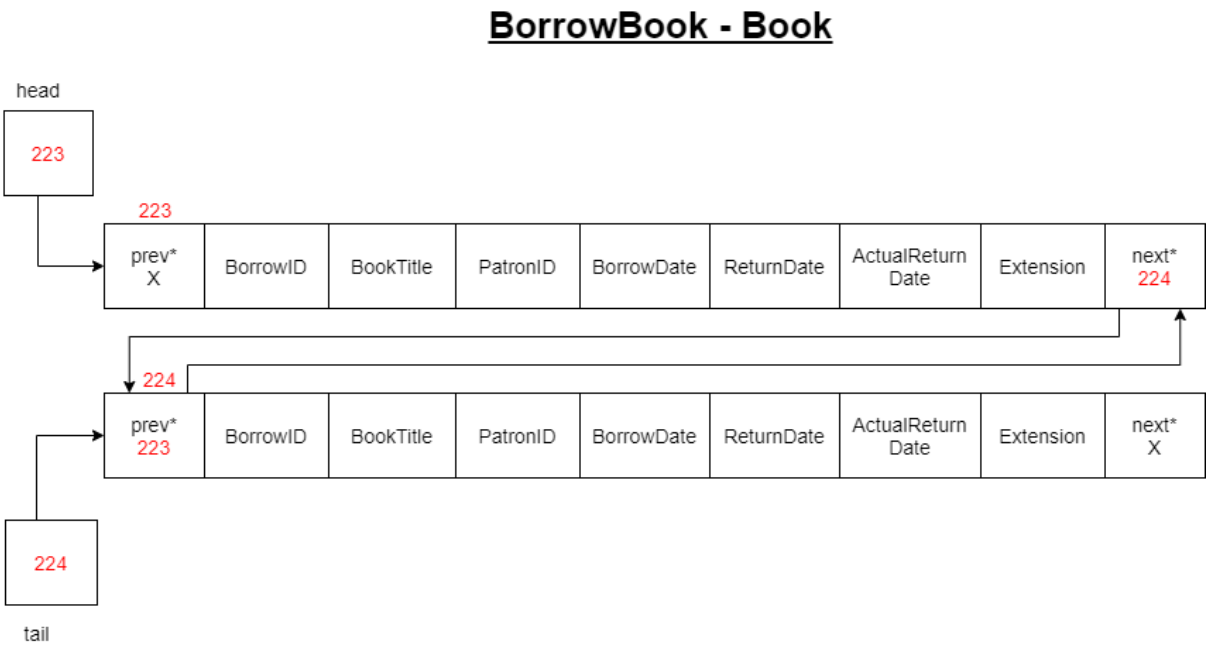


Figure 2: One Dimensional Doubly Linked List for Borrow information

The Figure 2 shown above is the example of storing book borrowed information in a doubly linked list. This diagram depicts that the books borrowed can be used to print all records from the beginning or print all records from the behind, this is also one of the assignment requirements where to print the last 10 books borrowed by a patron. However, each node in book borrowed linked list is linked together using its pointers (prev* and next*), so that the borrow book records can be printed in ascending order or descending order. This doubly linked list is not a circular linked list, therefore the first record of prev pointer and the last record of next pointer will be set to be NULL by default.

Advantage of Doubly Linked List

I. Able to traverse in bi-direction

Doubly linked list allow traverse in both directions (forward and backward). For example, it can travel from the starting to the end and from the end to the starting (Kakria, 2020). Hence, a technique known as traversal will be used to access each node within the linked list (SoftwareTestingHelp, 2020). Doubly linked list is recommended to be implemented when it requires to print the record from the behind as doubly linked list can remove the complexity from O(n) in singly linked list to O(1) using doubly linked list. A pointer called 'tail' will be pointed to the last node of the linked list therefore it can directly address the last node instead of using 'head' pointer to traverse from the beginning until the last node. In terms of the proposed LMS, the proposed LMS should be able to print the last 10 books borrowed by a patron, therefore it is suitable to use doubly linked list to store the borrow list, so that the record can print the last 10 books borrowed. Moreover, it is more efficient ever when it comes to insertion or deletion at a known position, because it can use the condition, such as the index given by the user divided by the size of linked list, to decide the node should be traversed from the beginning or behind, from doing that so, it can reduce the time.

II. Able to create a new node before the given node

A new node can be created and inserted easily in a doubly linked list. It only needs to set the previous node and the next node pointers wisely in order to connect the previous node and the next node to the correct pointers (Mishra, 2016).

III. Delete operation can be done easily

Deleting a node is very simple in Doubly linked list, however, it requires extensive handling whenever the node needed to be removed is the first or last node in the list. It can be easily point the next of the previous node which next to the current node (node to be removed) and the previous next node point to the previous of the current node (Studytonight.com, 2020).

2.2 Two dimensional singly linked list

Two dimensional singly linked list is a set of nodes which construct together into a linear structure. Every node is a composite element that contains the next item and a link to connect another node known as next (Rajput, 2018). In terms of proposed LMS, the library books in the proposed system are stored using two dimensional singly linked list, the reason of proposing two dimensional singly linked list to store the book is, each book will have two copies. Therefore, whenever adding a new book, two copies of the added book will be created automatically. The first dimensional will be storing the similar attributes of the book, such as book title, book author, book category, book genre, and book ISBN, however, the second dimensional will be storing its own unique attributes, which is book ID and followed by book status. This removes the similar attributes to be stored into the linked list but increase the efficiency of its processes. Unlike two dimensional array, two dimensional array will be storing all the information along with even its an empty value, which is low efficient to store all the book information compared to two dimensional linked list. Using two dimensional array might be drained a lot of memory spaces while running the program itself. From using two dimensional singly linked list to store book information, it can reduce all the empty value and increase the efficient performance while retrieving the data from linked list. The Figure 3 shown below is the example of two-dimensional singly linked list will be used to store book information for AACL.

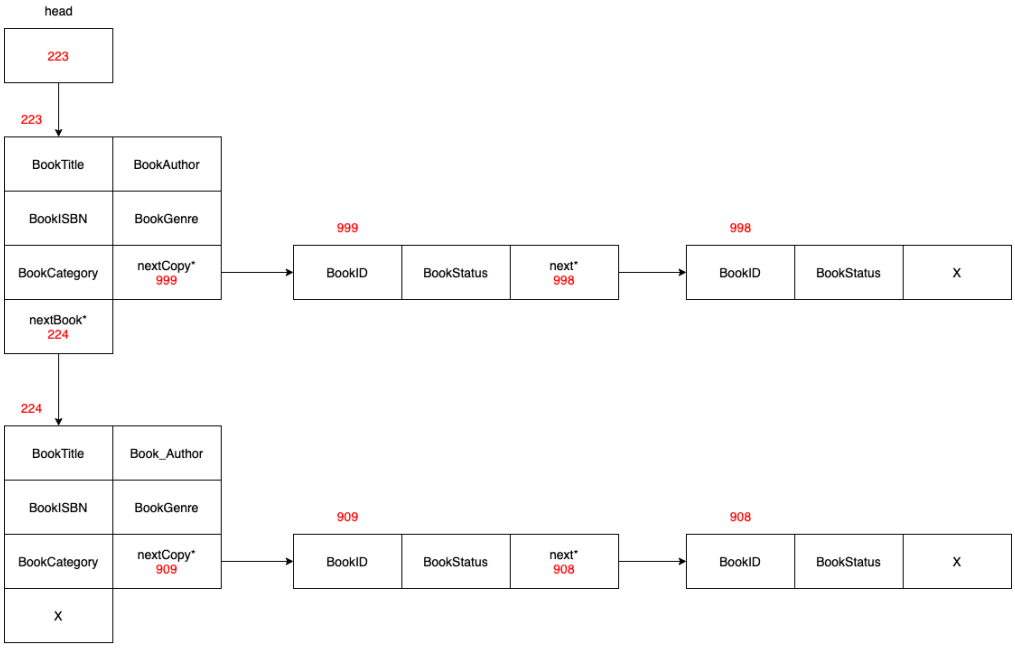


Figure 3: Two-Dimensional Singly Linked List for Book information

Advantages of Two Dimensional Singly Linked List

I. Dynamic Size

The linked list size can be unlimited and is not fixed as much as CPU can assign to the operation. Because linked lists size can expand or decrease. Therefore, the linked list size will constantly change with every insertion or deletion (Yashas, 2020). Two dimensional singly linked list does not need to specify how many sizes needed to be inserted, so that it is suitable to propose as data structure to store the book information.

II. Save Memory Space

Singly Linked List only able to use one pointer variable to connect in order for the node to consume less memory space comparing to the other nodes such as Doubly Linked List and Circular Linked List. Therefore, whenever using singly linked list, the elements could possibly be contained into the memory space available and data can easily store into the computer (yangerleo, 2019). Compared to array list, array list will waste the memory spaces because array list will have the possibility of storing empty value.

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <ctime>
#include <time.h>
#include <iomanip>
```

Figure 4: Library of LMS

The Figure 4 shown above is the import libraries that will be used in Anggerik Aranda Community Library (AALC) Library Management System.

```
// Structs and Classes Declaration
struct DoublyNodeTypeForPatron {
    string PatronID;
    string PatronName;
    string PatronGender;
    string PatronDOB;
    string PatronEmail;
    string PatronContactNumber;

    DoublyNodeTypeForPatron* next;
    DoublyNodeTypeForPatron* prev;
};

class PatronList {
public:
    int sizeforPatron = 0;
    void AddPatron(string, string, string, string, string, string);
    void SearchPatron_ID(string);
    void SearchPatron_NAME(string);
    void PrintfromBeginning();
    void PrintfromBehind();
    void UpdatePatronInformation(string);
    void getPatronwithActiveBookBorrow();
    void PrintLast10BooksByAPatron();

    void PrintPatronID();
};

```

Figure 5: Struct and Class of Patron linked list

The Figure 5 shown above is the struct and class for patron linked list. Line 13 to 18 is the data members of each node, which consist of patron ID, patron name, patron gender, patron DOB (Date of birth), patron email, and patron contact number. Line 20 and 21 are the pointers which is to point to either previous node or next node, which will also include in node type as data member. Line 26 to 36 is all the public methods of patron list.

```
39 struct DoublyNodeTypeForBorrow {
40     int BorrowID;
41     string BookTitle;
42     string PatronID;
43     string BorrowDate;
44     string ReturnDate;
45     string ActualReturnedDate;
46     bool Extension;
47 
48     DoublyNodeTypeForBorrow* next;
49     DoublyNodeTypeForBorrow* prev;
50 };
51 
52 class BorrowList {
53 public:
54     int BookSize = 0;
55 
56     void AddBorrow(string, string);
57     void PrintfromBeginning();
58     void PrintfromBehind();
59     void UpdateActualReturnedDate(int);
60     void Print10Last();
61     void MakeExtension(int);
62     void UpdateExtensionDate(int, string);
63     void SetBookToUnavailable(string);
64     void PrintBorrowList();
65     void PrintBorrowID();
66 
67     //Get today date and return date (15 days after)
68     string getToday();
69     string getReturnedDate();
70 };
```

Figure 6: Struct and Class of Borrow linked list

The Figure 6 shown above is the struct and class for borrow linked list. Line 40 to 49 is the data members of each node, which consist of borrow ID, book title, patron ID, borrow date, return date, actual returned date, and extension. Line 48 and 49 are the pointers which is to point to either previous node or next node, which will also include in node type as data member. Line 54 to 69 is all the public methods of patron list.

https://drive.google.com/file/d/1PRwxUL48ovhBxyyKY6p7a27DHh7TDesF/view?usp=sharing

```
78 struct colNode {
79     int BookID;
80     bool Status;
81     colNode* next;
82 };
83
84 struct BookTitleList {
85     string BookTitle;
86     BookTitleList* next;
87 };
88
89 struct rowNode {
90     string BookTitle;
91     string BookAuthor;
92     string BookGenre;
93     string BookCategory;
94     int BookISBN;
95     colNode* nextCopy;
96     rowNode* nextBook;
97 };
```

Figure 7: Struct of Book linked list

The Figure 7 shown above is the struct for book linked list. The line 78 to 82 and 89 to 97 are Struct for the multidimensional book linked list. Line 84 to 87 is another Struct for BookTitle linked list. Line 90 to 94 is the data members of each node, which consists of book title, book author, book genre, book category, and book ISBN. Line 95 and 96 are the pointer which is to point to the copy node and next book node, which will also include in node type as data member. Line 79 and 80 are the data members of each column node, which consists of book ID and status. Line 81 is the pointer which is to point to the next copy.

```
99  class BookList {
100 private:
101     int getRandomISBN();
102 public:
103     colNode* AddFirstCopy();
104     colNode* AddSecondCopy();
105     void AddNewBook(string, string, string, string);
106     void Print();
107     void PrintAvailableBookTitle();
108     void Search_Category(string);
109     void Search_Genre(string);
110     void Search_Title(string);
111     void UpdateBookInfo(string, string, string, string, int);
112     void PrintISBN();
113     void DeleteBook(int);
114 };
```

Figure 8: Class of Book linked list

The Figure 8 is the class for book linked list. The line 101 is the private method, and line 103 to 113 are the public methods.

```
117 // Global Variable Declaration
118 DoublyNodeTypeForPatron* headforPatron;
119 DoublyNodeTypeForPatron* tailforPatron;
120
121 DoublyNodeTypeForBorrow* headforBorrow;
122 DoublyNodeTypeForBorrow* tailforBorrow;
123
124 rowNode* head;
125 colNode* colHead;
126 BookTitleList* bookAvailableList;
127
128 int sizeRow = 0;
129 int sizeCol = 0;
130 const int MAX_COPIES = 2;
131 int sizeforPatron = 0;
132 int BorrowBookSize = 1;
133 const int MAX_PRINT = 10;
134 const int MAX_BORROW = 3;
```

Figure 9: Global variables used in LMS

The Figure 9 shown above is the global variables used in LMS. Line 118 and 119 is to create a head pointer for patron list. Line 121 and 122 is to create a head pointer for borrow list. Line 124 and 125 is to create a head pointer for book list. Line 128 to 134 are the variables that will be used in some methods of classes.

3.0 Algorithms (Flow Charts)

getGender

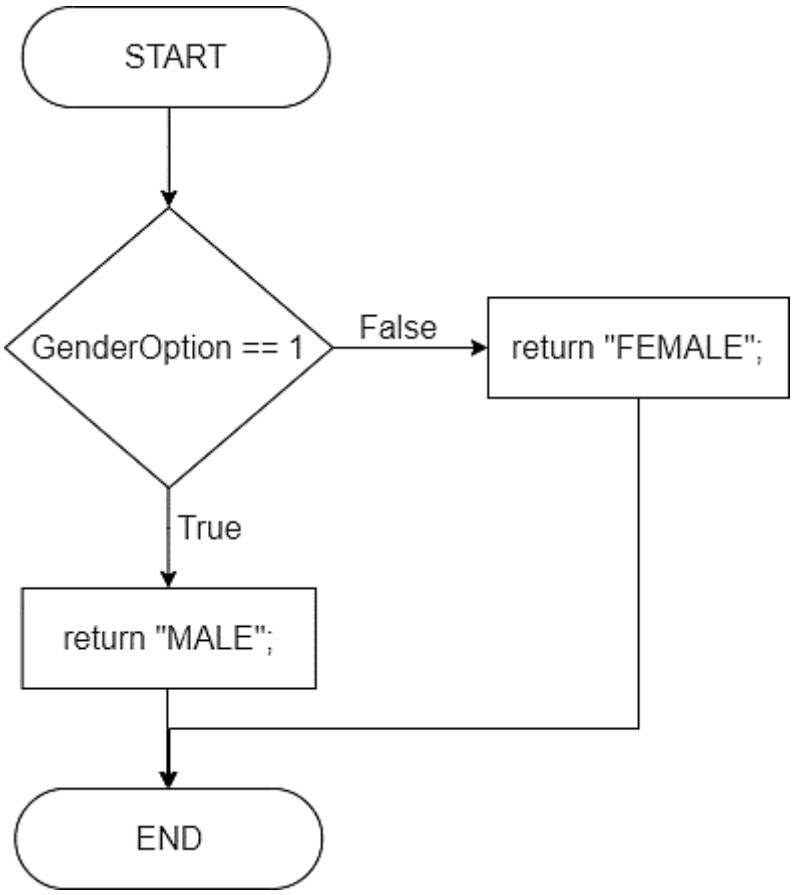


Figure 10: getGender method flowchart

Code Snippet

```
139 string getGender(int GenderOption) {
140     //This method will be return either MALE or FEMALE as string
141     if(GenderOption == 1) {
142         return "MALE";
143     } else {
144         return "FEMALE";
145     }
146 };
```

Figure 11: getGender method

The Figure 11 shown above is the getGender method. This method it to return either Male or Female as string type when passing the value from the object. The GenderOption will only be either 1 or 0 and it will be controlled and called from other methods of linked list such as patron.

checkBorrowID

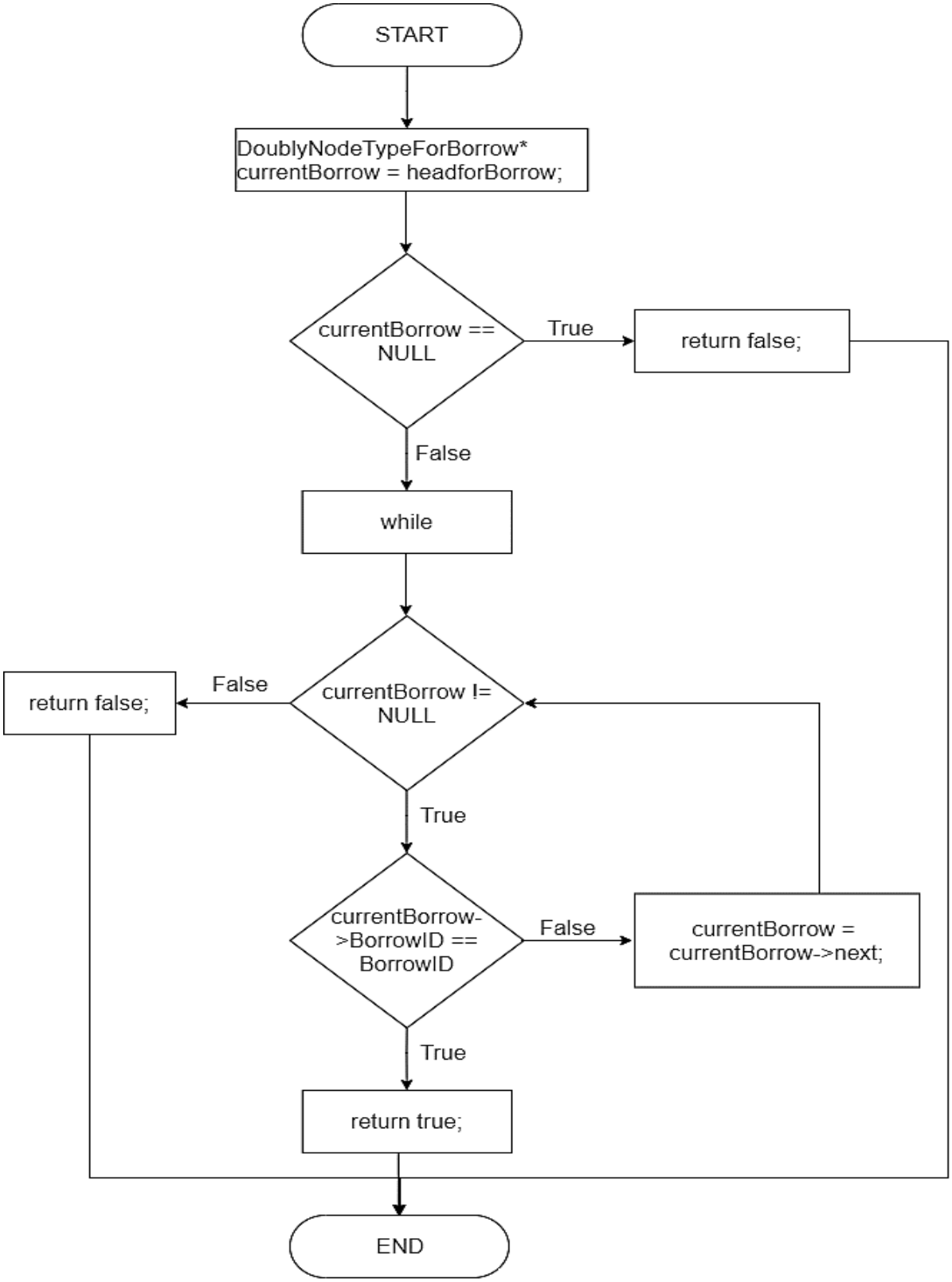


Figure 12: checkBorrowID method flowchart

Code Snippet

```
148 bool checkBorrowID(int BorrowID) {
149     //Copy value of headforBorrow into currentBorrow
150     DoublyNodeTypeForBorrow* currentBorrow = headforBorrow;
151     if (currentBorrow == NULL) {
152         //Check if the borrow list is not pointing to any link list, then return false
153         return false;
154     }
155     else {
156 
157         while (currentBorrow != NULL) {
158             //If the borrow list is pointing to a link list, then check if the borrow ID is matched to the given borrow ID
159             if (currentBorrow->BorrowID == BorrowID) {
160                 //If the borrow ID is found, then return true
161                 return true;
162             }
163             //If not, then move to the next link list record
164             currentBorrow = currentBorrow->next;
165         }
166     }
167     //Return false, if the given borrow ID does not match any in link list of borrow
168     return false;
169 };
```

Figure 13: checkBorrowID method

The Figure 13 shown above is the checkBorrowID method. The line 151 to 154 is to check if the head pointer is pointing to node, if not, then the line 157 to 165 is to iterate all the borrow list to check if the given borrow ID is already existed in the borrow link list. At the end of this method, will be returning either true or false value. If the given borrow ID is found in the method, then return true, else return false.

case_insensitive_match

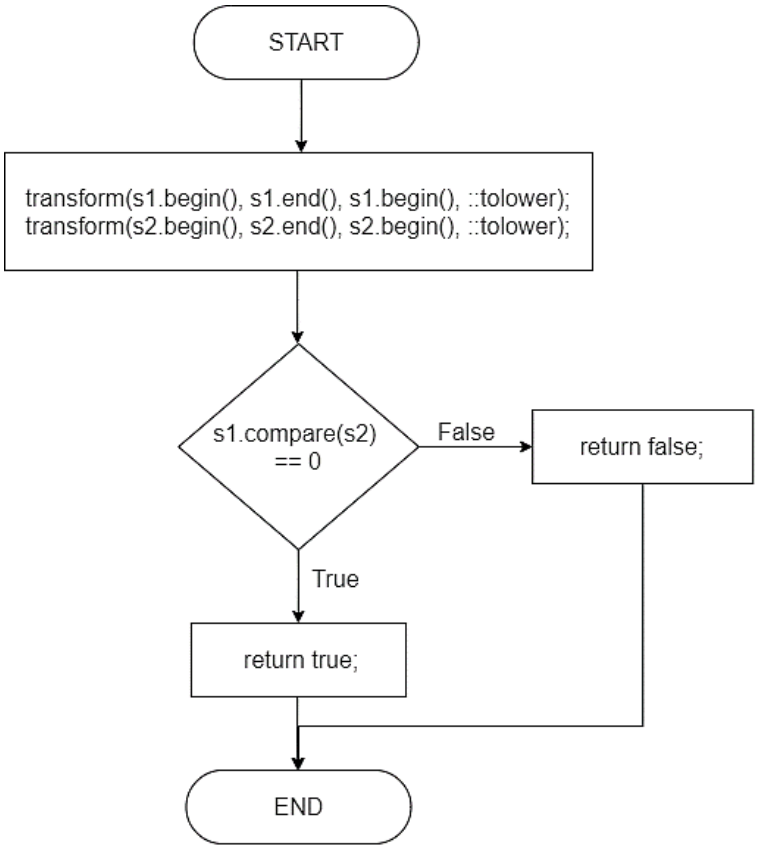


Figure 15: case_insensitive_match method flowchart

Code Snippet

```
171 bool case_insensitive_match(string s1, string s2) {
172     //This method is to convert string to all lower cases for the given string 1 and string 2
173     transform(s1.begin(), s1.end(), s1.begin(), ::tolower);
174     transform(s2.begin(), s2.end(), s2.begin(), ::tolower);
175 
176     //After converting to the lower case, then start to compare if the two given strings are matched
177     if (s1.compare(s2) == 0)
178         return true; //The strings are same
179     return false; //not matched
180 }
```

Figure 14: case_insensitive_match method

The Figure 14 shown above is the case_insensitive_match method. The method will be receiving two strings as parameters to make a comparison to check if the two given strings are equals to 0 by converting all the string to be lower cases, which line 173 and 174 are making two given string to be lower cases, and line 177 is to compare if two given strings are equal to 0 and then return true value to its method. The method will return false to the method only if the comparison in line 177 is failed.

totalBookBorrowByPatron

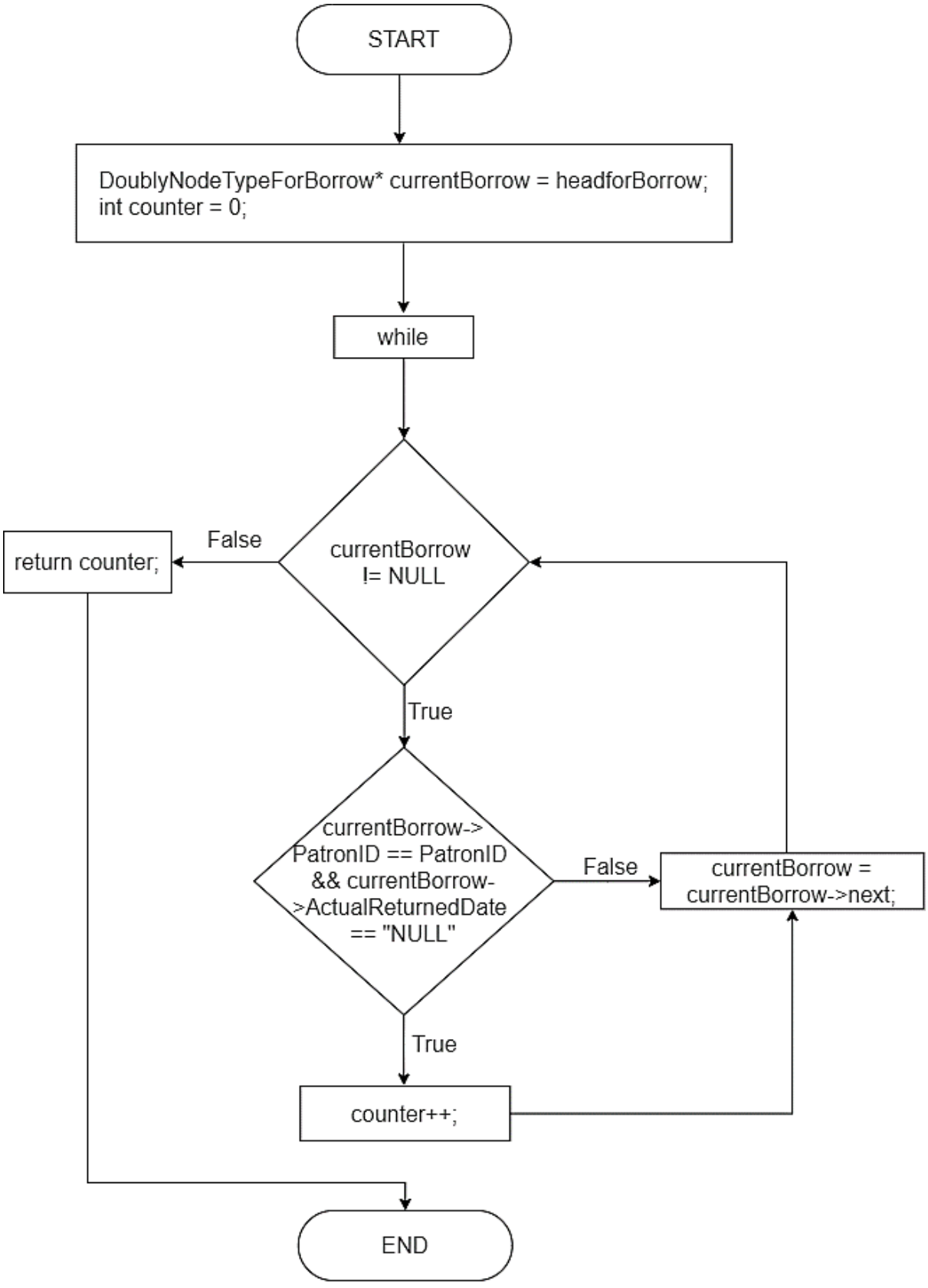


Figure 16: totalBookBorrowByPatron method flowchart

Code Snippet

```
int totalBookBorrowedByPatron(string PatronID) {
    //Copy value of headforBorrow into currentBorrow
    DoublyNodeTypeForBorrow* currentBorrow = headforBorrow;

    //Instatianted counter to be 0
    int counter = 0;

    //Iterate if the currentBorrow is not NULL
    while(currentBorrow != NULL) {
        //Check if the Patron ID of currentBorrow is matched to the given PatronID and the actual returned date is NULL
        if(currentBorrow -> PatronID == PatronID && currentBorrow->ActualReturnedDate == "NULL") {
            //Increment 1 to counter
            counter++;
        }
        //Move to the next borrow
        currentBorrow = currentBorrow -> next;
    }
    //Return counter value to the method
    return counter;
};
```

Figure 17: totalBookBorrowedByPatron method

The Figure 17 shown above is the totalBookBorrowedByPatron method. This method is mainly to count the number of books borrowed by the patron based on the given patron ID. The method will go through the whole borrow list to check if the given patron ID matches with the patron ID of borrow list while the actual returned date is still NULL, then increment 1 to the counter to indicate the number of book borrowed.

3.1 Books list

3.1.1 Add book

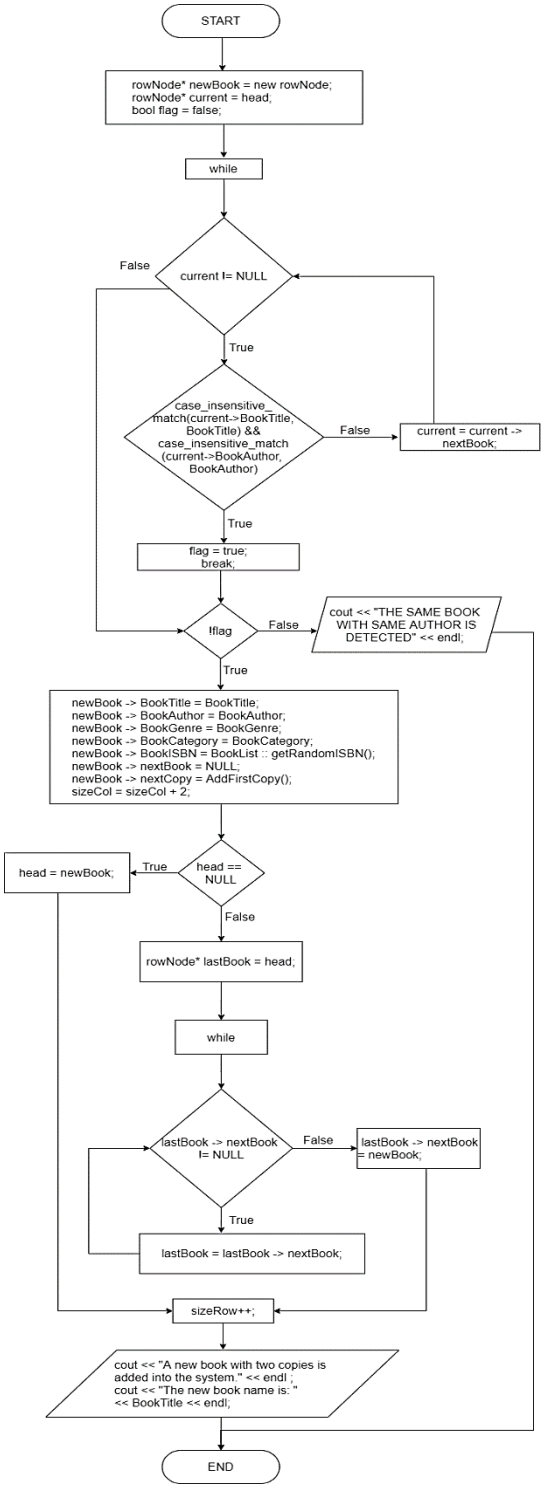


Figure 18: Books - Add Book flowchart

LINK: https://drive.google.com/file/d/1PKDFfTdiSEM2FiE7DIAUS9W-5TCPOEjR/view?usp=sharing

Code Snippet

```
900 void BookList::AddNewBook(string BookTitle, string BookAuthor, string BookGenre, string BookCategory) {
901     rowNode* newBook = new rowNode; //Create a new node type
902     rowNode* current = head; //Copy value of head into current
903     bool flag = false;
904 
905     while(current != NULL) { //Iterate if the current is not NULL
906         //Check if the added new book does not have the same name with the same author
907         if(case_insensitive_match(current -> BookTitle, BookTitle) && case_insensitive_match(current -> BookAuthor, BookAuthor)) {
908             flag = true; //Set flag to be true
909             break; //Break the iteration
910         }
911         current = current -> nextBook; //Move to the next book
912     }
913 
914     if(!flag) { //If the added new book is not added before
915         newBook -> BookTitle = BookTitle;
916         newBook -> BookAuthor = BookAuthor;
917         newBook -> BookGenre = BookGenre;
918         newBook -> BookCategory = BookCategory;
919         newBook -> BookISBN = BookList::getRandomISBN();
920         newBook -> nextBook = NULL;
921         newBook -> nextCopy = AddFirstCopy();
922         sizeCol = sizeCol + 2;
923 
924         if(head == NULL ){ //If the head is NULL
925             head = newBook;
926         } else {
927             rowNode* lastBook = head; //Copy value of head into lastBook
928             while(lastBook -> nextBook != NULL){ //Iterate if the next book of last book is not NULL
929                 lastBook = lastBook -> nextBook; //Move to the next book
930             }
931             lastBook -> nextBook = newBook; //Assign newBook node type to the previous nextBook
932         }
933         sizeRow++;
934         cout << "A new book with two copies is added into the system." << endl;
935         cout << "The new book name is: " << BookTitle << endl;
936     } else {
937         cout << "THE SAME BOOK WITH SAME AUTHOR IS DETECTED";
938         cout << endl;
939     }
940 
941 };
```

Figure 19: AddNewBook method - bookList

The Figure 19 shown above is the AddNewBook method in BookList class. Line 901 to 903 is variables declaration. Line 905 to 912 is to check if the given book title and the given book author are the exactly same as the data that stored in the linked list by converting into lower cases to do comparison, set the flag to be true and break the iteration if two conditions are met. Line 915 to 935 will be executed only if the flag is a true value. Line 912 will be calling AddFirstCopy method to add copy. Line 924 to 932 is to add the new book to either first node or the last node of the linked list. Line 925 will execute if the head pointer is pointing to NULL, while the Line 928 to 930 will be iterate the linked list until its last node if the head pointing is not pointing to NULL. Line 931 is assigning the node of new book to the last node of linked list. Line 936 will execute if the same book title with its same author is detected in the Line 907.

Result

```
BOOK MENU:
1 – Add a new book
2 – Search
3 – Print
4 – Update book information
5 – Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 1
Enter the new book title: This is a new book
Enter the new book author: This has no author
Is the new book Fiction? (1 – YES / 0 – NO): 1
1 – Fantasy
2 – Science
3 – Historical
4 – Realistic
5 – Fan
Your option is: 1
A new book with two copies is added into the system.
The new book name is: This is a new book
```

Figure 20: Sample output 1 - AddNewBook method

The Figure 20 shown above is the sample output for AddNewBook method. The user will need to enter 1 to add a new book. Title and author can be string types, and the book is fiction or non-fiction can be entered by user using numeric which is either 1 or 0. The user can enter 1 to 5 to assign the genre to the new book. After entering all the details, then the successful message will be prompted to the user saying that a new book with two copies is added into the system. The fiction and genre options will be iterated until the user gives the valid option as shown in the title if the user gives a numeric which is not within the given options.

```
BOOK TITLE                      BOOK AUTHOR                 GENRE          CATEGORY       ISBN      AVAILABILITY
HARRY POTTER                    J. K. ROWLING               FANTASY        FICTION        6808      2
MEIN KAMPF                      ADOLF HITLER                HISTORICAL     FICTION        5250      2
13 REASONS WHY                  JAY ASHER                   REALISTIC      FICTION        74        2
THE INVISIBLE MAN               H. G. WELLS                 SCIENCE        FICTION        3659      2
THE HUNGER GAMES                SUZANNE COLLINS             NARRATIVE      NON-FICTION    8931      2
I HAVE A DREAM                  MARTIN LUTHER KING JR.      BIOGRAPHY      NON-FICTION    1273      2
CRIME AND PERIODICALS           NOVA EVERLY                 PERIODICALS    NON-FICTION    7545      2
A NEW EARTH                     ECKHART TOLLE               SELF-HELP      NON-FICTION    879       2
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL             REFERENCE      NON-FICTION    7924      2

1 - Print all book title
2 - Print Available book title
If you wish to quit Print Section, you may enter 0 *
Your option is: 0

BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 1
Enter the new book title: harry potter
Enter the new book author: J. K. ROWLING
Is the new book Fiction? (1 - YES / 0 - NO): 1
1 - Fantasy
2 - Science
3 - Historical
4 - Realistic
5 - Fan
Your option is: 1
THE SAME BOOK WITH SAME AUTHOR IS DETECTED
```

Figure 21: Sample output 2 - AddNewBook method

The Figure 21 shown above is another sample output for AddNewBook method. This is to show that if the user enters the same book title with the same author name into the system. The system is to prevent this to be happened, therefore, the error message will be prompted to the user indicating that the new added book is similar to the books existed in the system.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 1
Enter the new book title: A THOUSAND LOVE
Enter the new book author: JOHN DOE
Is the new book Fiction? (1 - YES / 0 - NO): 2
Please enter 1 or 0 ONLY.
Is the new book Fiction? (1 - YES / 0 - NO): 1
1 - Fantasy
2 - Science
3 - Historical
4 - Realistic
5 - Fan
Your option is: 6
Please enter 1 - 5 ONLY.
1 - Fantasy
2 - Science
3 - Historical
4 - Realistic
5 - Fan
Your option is: 1
A new book with two copies is added into the system.
The new book name is: A THOUSAND LOVE
```

Figure 22: Sample output 3 - AddNewBook method

The Figure 22 shown above is another sample output for AddNewBook method. When deciding the book is fiction or non-fiction, the user will just need to provide either 1 or 0 (1, indicating fiction, 0 indicating non-fiction), the system will be automatically assigned fiction or non-fiction as string into the system. If the user enters 2 as input, then the system will request the user to enter again for the fiction section. In terms of book genre, the user will be allowed to enter the number from 1 to 5 only, because the system will be using the number to return a book genre into the system. If the user enters 6 as input, which is not in the scope, the user will then be required to enter again for the book genre section.

3.1.2 Print book list

3.1.2.1 Print books number availability

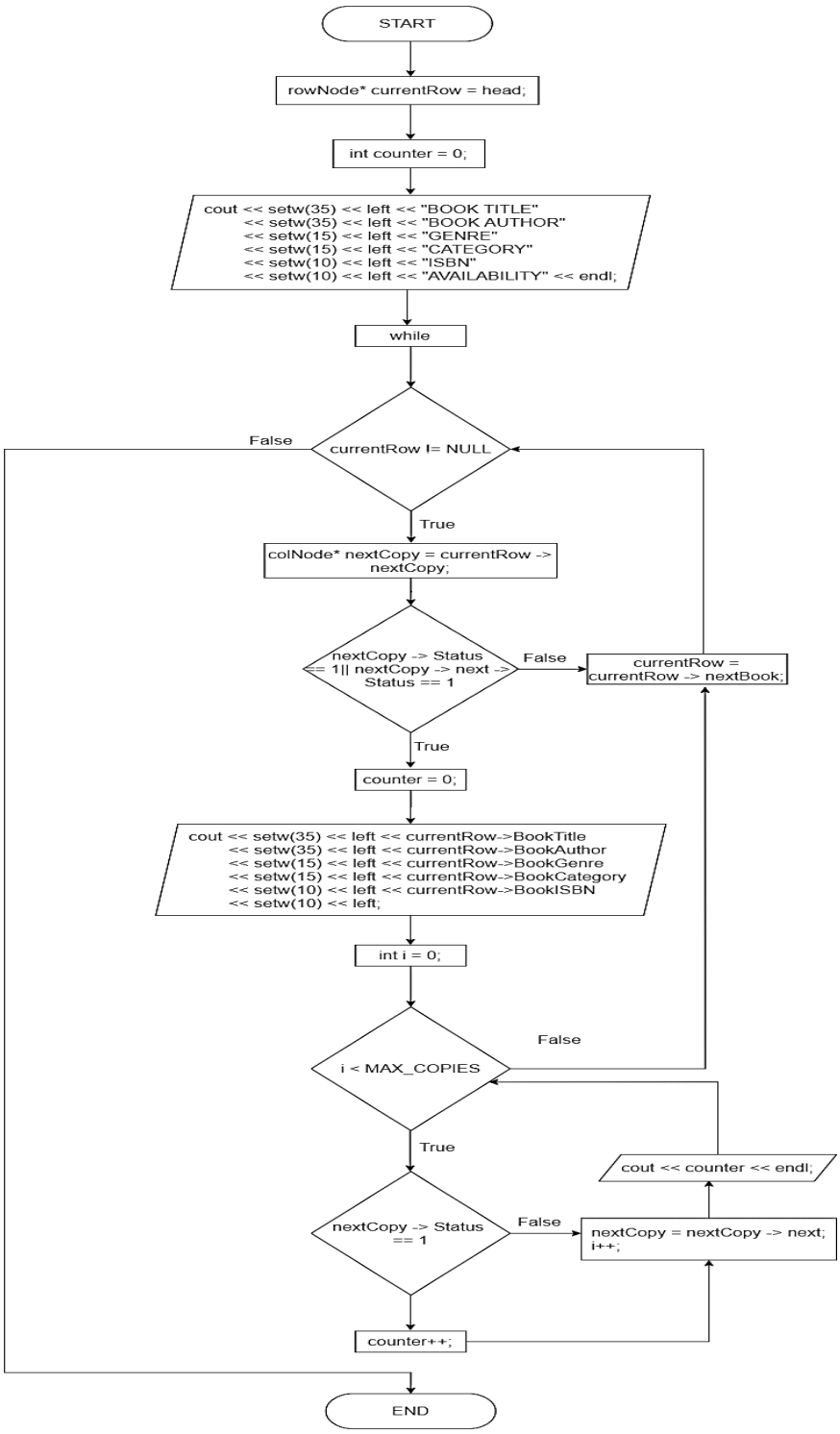


Figure 23: Books - Print books number availability flowchart

MAX_COPIES is a constant variable with the value of 2, which represents each book will have two copies.

Code Snippet

```
996  void BookList::Print() {
997      rowNode* currentRow = head; //Copy value of head into currentRow
998      int counter = 0;
999      cout << endl << endl;
1000     cout << setw(35) << left << "BOOK TITLE" << setw(35) << left << "BOOK AUTHOR" << setw(15) << left << "GENRE" << setw(15) <<
             left << "CATEGORY" << setw(10) << left << "ISBN" << setw(10) << left << "AVAILABILITY" << endl;
1001     while(currentRow != NULL) { //Iterate if the currentRow is not NULL
1002         colNode* nextCopy = currentRow -> nextCopy;
1003         if(nextCopy -> Status == 1 || nextCopy -> next -> Status == 1) { //Check if either books is available
1004             counter = 0;
1005             cout << setw(35) << left << currentRow -> BookTitle << setw(35) << left << currentRow -> BookAuthor << setw(15) << left
                     << currentRow -> BookGenre << setw(15) << left << currentRow -> BookCategory << setw(10) << left << currentRow ->
                     BookISBN << setw(10) << left;
1006             for(int i =0; i < MAX_COPIES; i++) {
1007                 if(nextCopy -> Status == 1) { //Check if the status is 0 then increment 1 to counter
1008                     counter ++;
1009                 }
1010                 nextCopy = nextCopy -> next; //Move to the next copy
1011             }
1012             cout << counter << endl; //Print the number of available copy
1013
1014         }
1015         currentRow = currentRow -> nextBook; //Move to the next book
1016
1017     }
1018     cout << endl;
1019 };
```

Figure 24: Print method - bookList

The Figure 24 shown above is the Print method. Line 997 to 998 are the variable declarations. Line 1002 to 1012 will be executed only if the book linked list is not pointing to NULL. Line 1002 is the another pointer declaration. Line 1004 to 1010 will be executed only if the condition in Line 1003 is correct, which is one of the status of both book copies is available(1). Line 1006 to 1011 is to count the number of available copy and add 1 as increment to the counter variable when the condition in Line 1007 is true. Line 1012 is to print the number of available books, by accessing the counter variable.

Result

```
BOOK MENU:
1 – Add a new book
2 – Search
3 – Print
4 – Update book information
5 – Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 3
1 – Print all book title
2 – Print Available book title
Your option is: 2

BOOK TITLE                      BOOK AUTHOR                 GENRE          CATEGORY       ISBN      AVAILABILITY
HARRY POTTER                    J. K. ROWLING               FANTASY        FICTION        6808      2
MEIN KAMPF                      ADOLF HITLER                HISTORICAL     FICTION        5250      2
13 REASONS WHY                  JAY ASHER                   REALISTIC      FICTION        74        2
THE INVISIBLE MAN               H. G. WELLS                 SCIENCE        FICTION        3659      2
THE HUNGER GAMES                SUZANNE COLLINS             NARRATIVE      NON-FICTION    8931      2
I HAVE A DREAM                  MARTIN LUTHER KING JR.      BIOGRAPHY      NON-FICTION    1273      2
CRIME AND PERIODICALS           NOVA EVERLY                 PERIODICALS    NON-FICTION    7545      2
A NEW EARTH                     ECKHART TOLLE               SELF-HELP      NON-FICTION    879       2
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL             REFERENCE      NON-FICTION    7924      2
This is a new book              This has no author          FANTASY        FICTION        4441      2
```

Figure 25: Sample output 1 - Print method - bookList

The Figure 25 is the sample output for Print method. The user will need to enter 3 to enter the print section. After entering the print section, the user can enter either 1 or 2 to print the book linked list.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 3
1 - Print all book title
2 - Print Available book title
Your option is: 3
Please enter 1-2 ONLY.
1 - Print all book title
2 - Print Available book title
If you wish to quit Print Section, you may enter 0 *
Your option is: 2

BOOK TITLE                      BOOK AUTHOR                 GENRE          CATEGORY       ISBN      AVAILABILITY
HARRY POTTER                    J. K. ROWLING               FANTASY        FICTION        6808      2
MEIN KAMPF                      ADOLF HITLER                HISTORICAL     FICTION        5250      2
13 REASONS WHY                  JAY ASHER                   REALISTIC      FICTION        74        2
THE INVISIBLE MAN               H. G. WELLS                 SCIENCE        FICTION        3659      2
THE HUNGER GAMES                SUZANNE COLLINS             NARRATIVE      NON-FICTION    8931      2
I HAVE A DREAM                  MARTIN LUTHER KING JR.      BIOGRAPHY      NON-FICTION    1273      2
CRIME AND PERIODICALS           NOVA EVERLY                 PERIODICALS    NON-FICTION    7545      2
A NEW EARTH                     ECKHART TOLLE               SELF-HELP      NON-FICTION    879       2
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL             REFERENCE      NON-FICTION    7924      2
```

Figure 26: Sample output 2 - Print method - bookList

The Figure 26 shown above is another sample output for Print method. The user will be entered into the iteration if the given option is not either 1 or 2. The system is allowing user to enter either 1 or 2 to print all the records and to prevent the unclear to the system or making the system to be crashed, so the validation is applied.

3.1.2.2 Print available book title

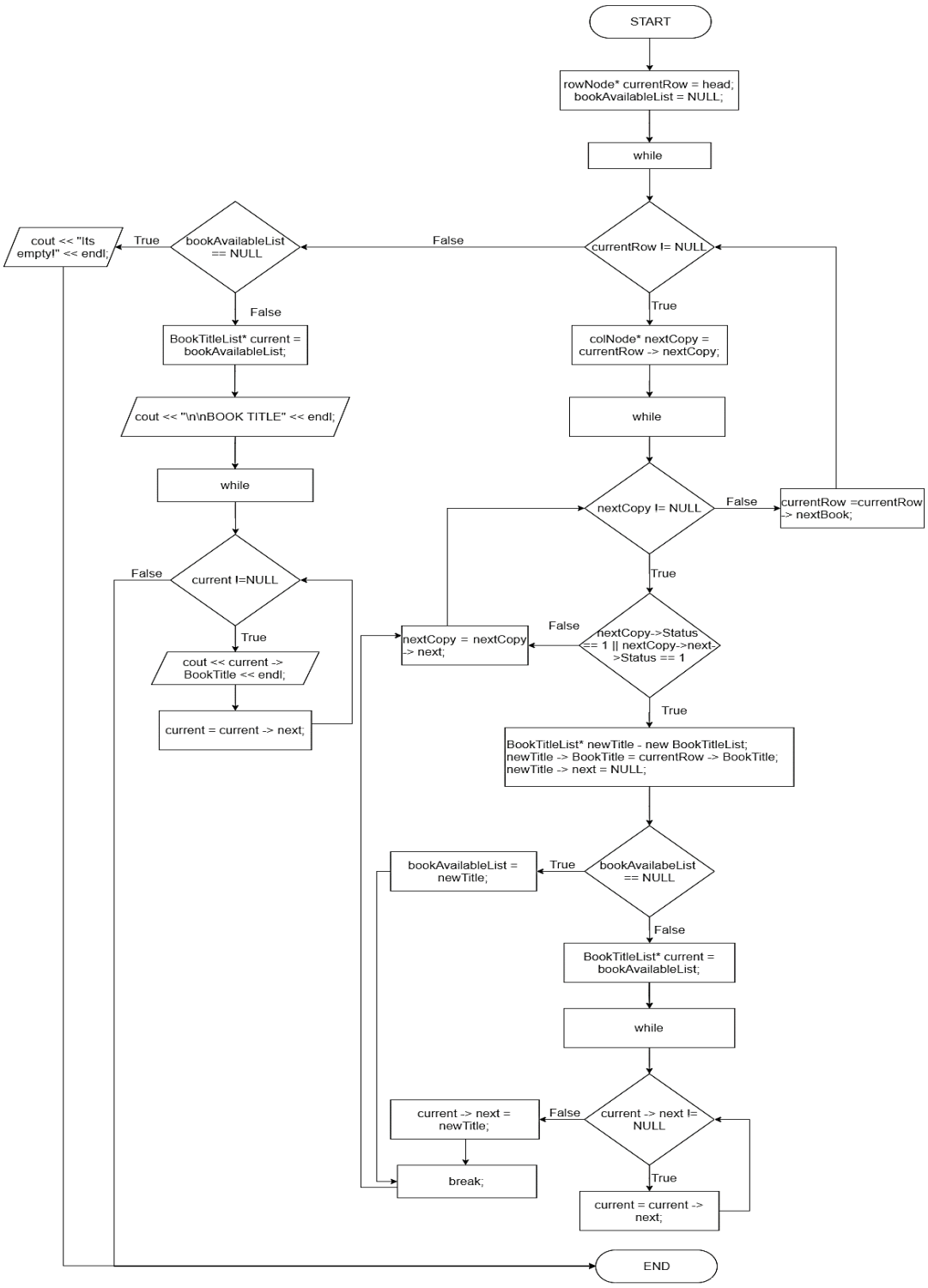


Figure 27: Books - Print available book title flowchart

LINK:
https://drive.google.com/file/d/1_K6JiHtLUs9LMLXaEfUlpjhqGRj708vn/view?usp=sharing

Code Snippet

```
1021 void BookList::PrintAvailableBookTitle() {
1022     rowNode* currentRow = head; //Copy value of head into currentRow
1023     bookAvailableList =  NULL;
1024     while(currentRow != NULL) { //Iterate if the currentRow is not NULL
1025         colNode* nextCopy = currentRow -> nextCopy;
1026         while(nextCopy != NULL) { //Iterate if the nextCopy is not NULL
1027             if(nextCopy -> Status == 1 || nextCopy -> next -> Status == 1) { //Check if either book is available
1028                 BookTitleList* newTitle = new BookTitleList;
1029                 newTitle -> BookTitle = currentRow -> BookTitle;
1030                 newTitle -> next = NULL;
1031                 if(bookAvailableList == NULL){ //Check if the bookAvailableList is NULL
1032                     bookAvailableList = newTitle;
1033                 } else {
1034                     BookTitleList* current = bookAvailableList;
1035                     while (current -> next != NULL) { //Iterate if the next of current is not NULL
1036                         current = current -> next; //Move to the next BookTitleList
1037                     }
1038                     current -> next = newTitle; //Assign the title to the next BookTitleList
1039                 }
1040                 break; //Break the iteration
1041             }
1042             nextCopy = nextCopy -> next; //Move to the next copy
1043         }
1044         currentRow = currentRow -> nextBook; //Move to the next book
1045     }
1046 
1047     if(bookAvailableList == NULL) { //Check if the bookAvailableList is empty
1048         cout << "Its empty!";
1049         cout << endl;
1050     } else {
1051         BookTitleList* current = bookAvailableList; //Copy value of bookAvailableList into current
1052         cout << "\n\nBOOK TITLE" << endl;
1053         while (current != NULL) { //Iterate if the current is not NULL
1054             cout << current -> BookTitle <<endl; //Print the record
1055             current = current -> next; //Move to the next book title
1056         }
1057         cout << endl;
1058     }
1059 };
```

Figure 28: PrintAvailableBookTitle method - bookList

The Figure 28 shown above is the PrintAvailableBookTitle method. Line 1022 to 1023 are the variable declarations. Line 1025 to 1044 will be executed only if the book is not pointing to NULL in Line 1024. Line 1025 is the a pointer declaration which to point to another dimensional linked list. Line 1027 to 1042 will be executed only if the second dimensional linked list is not pointing to NULL as shown in Line 1026. Line 1028 to 1041 will be executed only if the condition in Line 1027 is true, which is to check one of the status of both book copies is available. If either one is available, then create a new pointer variable (BookTitleList) to store the book title as shown in Line 1028 to 1040. Line 1031 to 1039 is to add the created node to the head or the last node of BookTitleList linked list. If BookTitleList is not pointing to NULL, then Line 1035 to 1037 is to travel the BookTitleList linked list until the last node, and store the created node. Line 1042 is to move to the next copy. Line 1044 is to move to the next book. Line 1047 is responsible to print the book title by checking if the bookAvailableList is pointing to NULL or not. If it points to NULL, then Line 1048 to 1049 will be executed. Otherwise, Line 1051 to 1057 will be executed. Line 1053 to 1056 is to print all the records in bookAvailableList.

Result

```
BOOK MENU:
1 – Add a new book
2 – Search
3 – Print
4 – Update book information
5 – Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 3
1 – Print all book title
2 – Print Available book title
Your option is: 1


BOOK TITLE
HARRY POTTER
MEIN KAMPF
13 REASONS WHY
THE INVISIBLE MAN
THE HUNGER GAMES
I HAVE A DREAM
CRIME AND PERIODICALS
A NEW EARTH
THE HERO WITH A THOUSAND FACES
This is a new book
```

Figure 29: Sample output 1 - PrintAvailableBookTitle method - bookList

The Figure 29 shown above is the sample output for PrintAvailableBookTitle method. The user will need to enter 3 to enter print book section. The user then will require to enter either 1 or 2 only to print the records of book linked list.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 3
1 - Print all book title
2 - Print Available book title
Your option is: 3
Please enter 1-2 ONLY.
1 - Print all book title
2 - Print Available book title
If you wish to quit Print Section, you may enter 0 *
Your option is: 1


BOOK TITLE
HARRY POTTER
MEIN KAMPF
13 REASONS WHY
THE INVISIBLE MAN
THE HUNGER GAMES
I HAVE A DREAM
CRIME AND PERIODICALS
A NEW EARTH
THE HERO WITH A THOUSAND FACES
```

Figure 30: Sample output 2 - PrintAvailableBookTitle method - bookList

The Figure 30 shown above is another sample output for PrintAvailableBookTitle method. The user will be entered into the iteration if the given option is not either 1 or 2. The system is allowing user to enter only either 1 or 2 to print all the records and to prevent the unclear input into the system or making the system to be crashed, the validation therefore is applied.

3.1.3 Search book

3.1.3.1 Based on Category

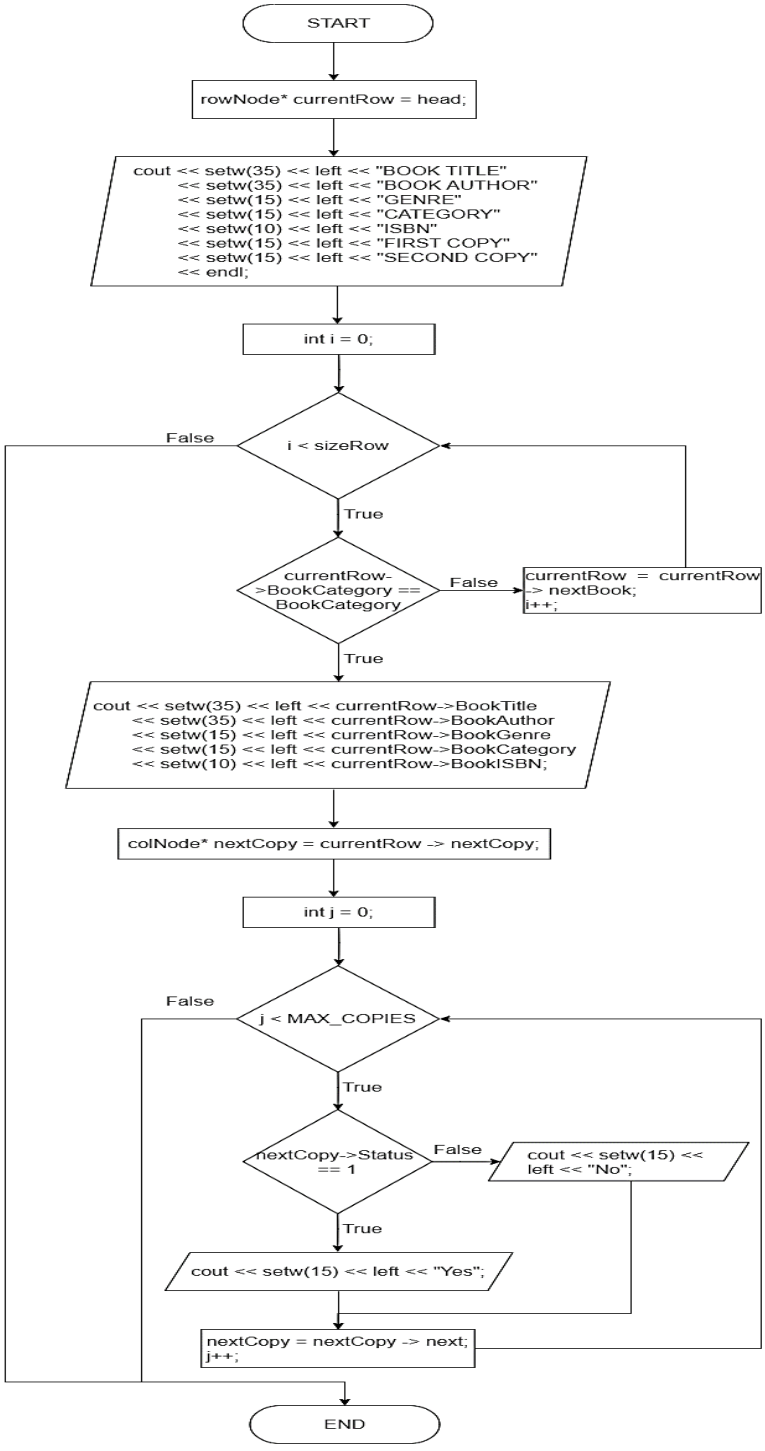


Figure 31: Books - Search book (Category) flowchart

LINK:
https://drive.google.com/file/d/1klRdNRsZClRQ7bH3Tu9ouNu45NxcNqkn/view?usp=sharing

Code Snippet

```
1061 void BookList::Search_Category(string BookCategory) {
1062     //Copy value of head into currentRow
1063     rowNode* currentRow = head;
1064     cout << endl << endl;
1065 
1066     cout << setw(35) << left << "BOOK TITLE" << setw(35) << left << "BOOK AUTHOR" << setw(15) << left << "GENRE" << setw(15) <<
         left << "CATEGORY" << setw(10) << left << "ISBN" << setw(15) << left << "FIRST COPY" << setw(15) << left << "SECOND COPY"
         <<endl;
1067 
1068     for(int i = 0; i < sizeRow; i++) { //Iterate all the books
1069         //Check if the given book category is matched with the book category of book list
1070         if(currentRow -> BookCategory == BookCategory) {
1071             cout << setw(35) << left << currentRow -> BookTitle << setw(35) << left << currentRow -> BookAuthor << setw(15) << left
                 << currentRow -> BookGenre << setw(15) << left << currentRow -> BookCategory << setw(10) << left << currentRow ->
                 BookISBN;
1072             colNode* nextCopy = currentRow -> nextCopy;
1073             for(int j = 0; j < MAX_COPIES; j++) { //Iterate all the copies
1074                 if(nextCopy -> Status == 1) { //Check if the Status is 1
1075                     cout << setw(15) << left << "Yes";
1076                 } else {
1077                     cout << setw(15) << left << "No";
1078                 }
1079                 nextCopy = nextCopy -> next; //Move to the next copy
1080             }
1081             cout << endl;
1082         }
1083 
1084         currentRow = currentRow -> nextBook; //Move to the next book
1085     }
1086     cout << endl;
1087 };
```

Figure 32: Search_Category method - bookList

The Figure 32 shown above is the Search_Category method. Line 1063 is the variable declaration. Line 1066 is to print the header, which will be showing what are the columns to be displayed. Line 1070 to 1084 will be executed only if the condition in Line 1068 is true, which is the variable of i is smaller than the size of books (sizeRow). Line 1071 to 1079 will be executed only if the given book category is matched with the book category in the book linked list as shown in Line 1070. Line 1071 is to print all the records of book. Line 1072 is a pointer declaration. Line 1074 to 1079 will be executed only if the condition in Line 1073 is true, which is the variable of j is smaller than MAX_COPIES, MAX_COPIES is a constant value, which contains a value of 2. Line 1074 will be check the status of each copy and execute either Line 1075 or Line 1077. If the status is equal to 1 then Line 1075 will be executed, otherwise Line 1077 will be executed. Line 1079 is to move to the next copy of the book, and increment 1 to j variable. Line 1084 is to move to the next book, and increment 1 to i variable.

Result

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 2
You want to search a book based on:
1 - Search based on Category
2 - Search based on Genre
3 - Search based on Title
Your option is: 4
INVALID INPUT
```

Figure 33: Sample output 1 - Search - bookList

The Figure 33 shown above is the sample output for Search section. The user is required to enter 2 to enter search section. The user will then be required to enter a number from 1 to 3 as input to print the records accordingly, if the user enters an invalid input such as 4, the system will then print INVALID INPUT since 4 is not within the options.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 2
You want to search a book based on:
1 - Search based on Category
2 - Search based on Genre
3 - Search based on Title
Your option is: 1
Fiction or Non-Fiction? (1 - Fiction / 0 - Non-Fiction): 1

BOOK TITLE                BOOK AUTHOR              GENRE        CATEGORY   ISBN   FIRST COPY   SECOND COPY
HARRY POTTER              J. K. ROWLING            FANTASY      FICTION    6808   Yes          Yes
MEIN KAMPF                ADOLF HITLER             HISTORICAL   FICTION    5250   Yes          Yes
13 REASONS WHY            JAY ASHER                REALISTIC    FICTION    74     Yes          Yes
THE INVISIBLE MAN         H. G. WELLS              SCIENCE      FICTION    3659   Yes          Yes
This is a new book        This has no author       FANTASY      FICTION    7710   Yes          Yes
```

Figure 34: Sample output 1 - Search_Category method - bookList

The Figure 34 shown above is the sample output for Search_Category method. After entering the search print book section, the user is required to enter a number from 1 to 3 to search the book. If the user wants to search based on category, then the user can enter 1 as the input. The system will then show if the user want to search based on fiction or non-fiction category. The user will just need to enter either 1 or 0 to search the category.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 2
You want to search a book based on:
1 - Search based on Category
2 - Search based on Genre
3 - Search based on Title
Your option is: 1
Fiction or Non-Fiction? (1 - Fiction / 0 - Non-Fiction): 2
Please enter 1 or 0 ONLY.
Fiction or Non-Fiction? (1 - Fiction / 0 - Non-Fiction): 1

BOOK TITLE                      BOOK AUTHOR                     GENRE          CATEGORY       ISBN      FIRST COPY     SECOND COPY
HARRY POTTER                    J. K. ROWLING                   FANTASY        FICTION        6808      Yes            Yes
MEIN KAMPF                      ADOLF HITLER                    HISTORICAL     FICTION        5250      Yes            Yes
13 REASONS WHY                  JAY ASHER                       REALISTIC      FICTION        74        Yes            Yes
THE INVISIBLE MAN               H. G. WELLS                     SCIENCE        FICTION        3659      Yes            Yes
A THOUSAND LOVE                 JOHN DOE                        FANTASY        FICTION        7710      Yes            Yes
```

Figure 35: Sample output 2 - Search_Category method - bookList

The Figure 35 shown above is another sample output for Search_Category method. The user is required to enter either 1 or 0 to search the category but the user enters a number other than 1 or 0, then the user is required to enter again before proceeding to the next operation.

3.1.3.2 Based on Genre

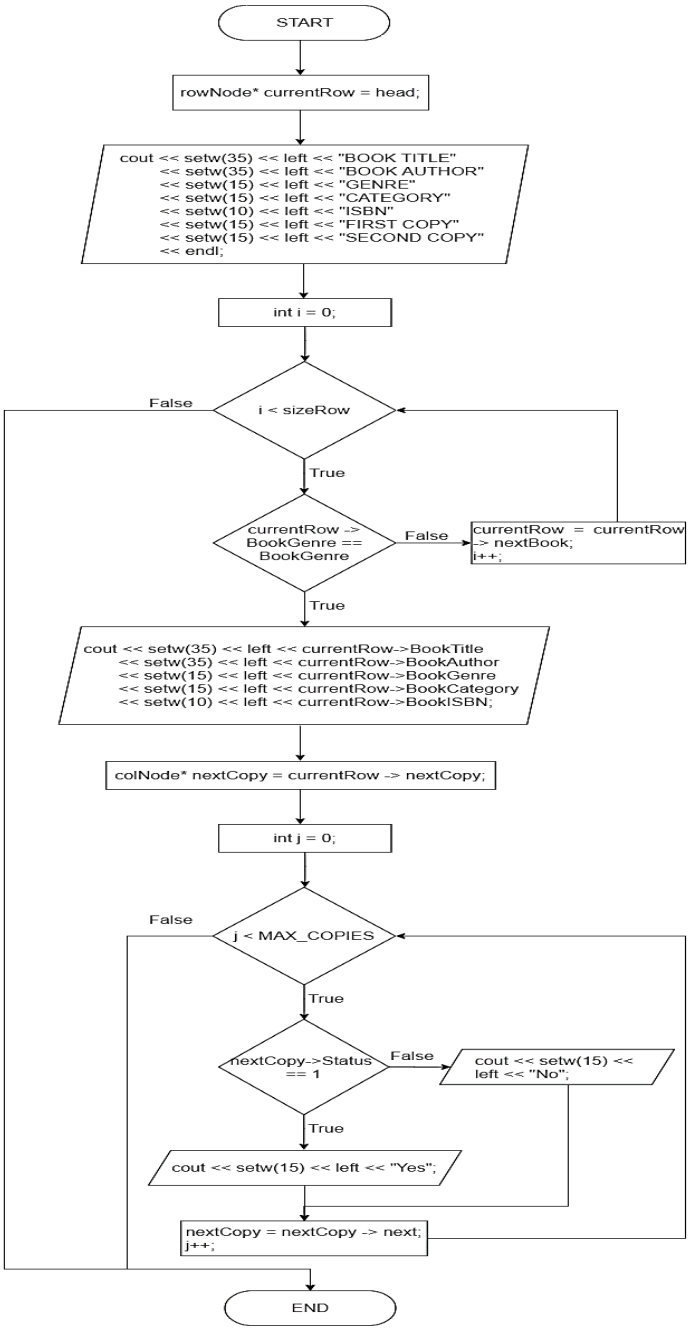


Figure 36: Books - Search book (Genre) flowchart

The Figure 36 flowchart above is the flow of searching a book genre in LMS. This algorithms flowchart is also applicable to the book category which is currentRow -> BookGenre == BookGenre change to currentRow -> BookCategory == BookCategory.

LINK:

https://drive.google.com/file/d/1jh0l-oP-Yg5RlnZ-ajUW7R1MWYGGFHH5/view?usp=sharing

Code Snippet

```
1089 void BookList::Search_Genre(string BookGenre) {
1090     rowNode* currentRow = head; //Copy value of head into currentRow
1091     cout << endl << endl;
1092 
1093     cout << setw(35) << left << "BOOK TITLE" << setw(35) << left << "BOOK AUTHOR" << setw(15) << left << "GENRE" << setw(15) <<
         left << "CATEGORY" << setw(10) << left << "ISBN" << setw(15) << left << "FIRST COPY" << setw(15) << left << "SECOND COPY"
         <<endl;
1094 
1095     for(int i = 0; i < sizeRow; i++) { //Iterate all the books
1096         if(currentRow -> BookGenre == BookGenre) { //Check if the given book genre is matched with the book genre of book list
1097             cout << setw(35) << left << currentRow -> BookTitle << setw(35) << left << currentRow -> BookAuthor << setw(15) << left
                 << currentRow -> BookGenre << setw(15) << left << currentRow -> BookCategory << setw(10) << left << currentRow ->
                 BookISBN;
1098             colNode* nextCopy = currentRow -> nextCopy;
1099             for(int j = 0; j < MAX_COPIES; j++) { //Iterate all the copies
1100                 if(nextCopy -> Status == 1) { //Check if the status is 1
1101                     cout << setw(15) << left << "Yes";
1102                 } else {
1103                     cout << setw(15) << left << "No";
1104                 }
1105                 nextCopy = nextCopy -> next; //Move to  the next copy
1106             }
1107             cout << endl;
1108         }
1109         currentRow = currentRow -> nextBook; //Move to the next book
1110     }
1111     cout << endl;
1112 };
```

Figure 37: Search_Genre method - bookList

The Figure 37 shown above is the Search_Genre method. Line 1090 is the variable declaration. Line 1093 is to print the header, which will be showing what are the columns to be displayed. Line 1096 to 1109 will be executed only if the variable of i is smaller than the size of book in Line 1095. Line 1097 to 1107 will be executed only if the given book genre is matched with the book genre in the book linked list as shown in Line 1096. Line 1097 is to print the data based on the current pointer. Line 1098 is the pointer variable declaration. Line 1100 to 1105 will be executed only if the variable of j is smaller than the MAX_COPIES, MAX_COPIES is a constant value, which contains value of 2. Line 1100 will check if the status is equal to 1, then Line 1101 will be executed, otherwise, Line 1103 will be executed. Line 1105 is to move to the next copy, and add 1 as increment to j variable. Line 1109 is to move to the next book, and add 1 as increment to i variable.

Result

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 2
You want to search a book based on:
1 - Search based on Category
2 - Search based on Genre
3 - Search based on Title
Your option is: 2
1 - Fantasy
2 - Science
3 - Historical
4 - Realistic
5 - Fan
6 - Narrative
7 - Biography
8 - Periodicals
9 - Self-Help
10 - Reference
Please enter genre: 1

BOOK TITLE                    BOOK AUTHOR                   GENRE          CATEGORY       ISBN      FIRST COPY     SECOND COPY
HARRY POTTER                  J. K. ROWLING                 FANTASY        FICTION        6808      Yes            Yes
This is a new book            This has no author            FANTASY        FICTION        7710      Yes            Yes
```

Figure 38: Sample output 1 - Search_Genre method - bookList

The Figure 38 shown above is the sample output for Search_Genre method. After entering the search print book section, the user is required to enter a number from 1 to 3 to search the book. If the user wants to search based on book genre, then the user can enter 2 as the input. The system will then show if the user want to search based on which book genre. The user will just need to enter a number from 1 or 10 to search the category.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 2
You want to search a book based on:
1 - Search based on Category
2 - Search based on Genre
3 - Search based on Title
Your option is: 2
1 - Fantasy
2 - Science
3 - Historical
4 - Realistic
5 - Fan
6 - Narrative
7 - Biography
8 - Periodicals
9 - Self-Help
10 - Reference
Please enter genre: 11
INVALID INPUT
```

Figure 39: Sample output 2 - Search_Genre method - bookList

The Figure 39 shown above is another sample output for Search_Genre method. The user is required to enter a number from 1 to 10 to search the genre but the user enters a number other than 1 to 10, then the user is required to enter again before proceeding to the next operation.

3.1.3.3 Based on Title

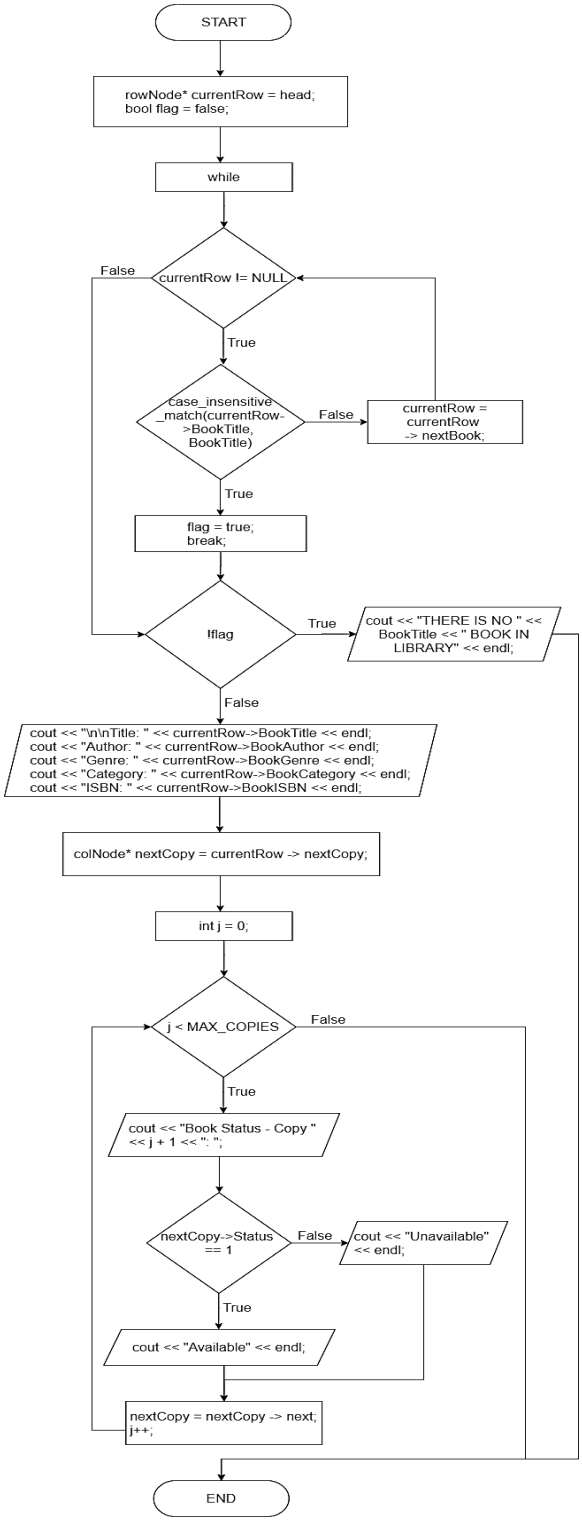


Figure 40: Books - Search book (Title) flowchart

LINK:
https://drive.google.com/file/d/1G161o2JUS855ZOJJCfhtZ2JLsaQAmECb/view?usp=sharing

Code Snippet

```
1114 void BookList::Search_Title(string BookTitle) {
1115     rowNode* currentRow = head; //Copy value  of head into currentRow
1116     bool flag = false;
1117     while(currentRow != NULL) { //Iterate if the currentRow is not NULL
1118         //Check if the given BookTitle is matched the BookTitle of book list
1119         if(case_insensitive_match(currentRow -> BookTitle, BookTitle)) {
1120             flag = true; //Set the flag to be true
1121             break; //Break the iteration
1122         }
1123         currentRow = currentRow -> nextBook; //Move to the next book
1124     }
1125 
1126     if(!flag) { //If no books found
1127         cout << "THERE IS NO " << BookTitle << " BOOK IN LIBRARY";
1128         cout << endl;
1129     } else {
1130         cout << "\n\nTitle: " << currentRow -> BookTitle <<endl;
1131         cout << "Author: " << currentRow -> BookAuthor << endl;
1132         cout << "Genre: " << currentRow -> BookGenre << endl;
1133         cout << "Category: " << currentRow -> BookCategory << endl;
1134         cout << "ISBN: " << currentRow -> BookISBN << endl;
1135 
1136         colNode* nextCopy = currentRow -> nextCopy;
1137         for(int j = 0; j < MAX_COPIES; j++) { //Iterate all the copies
1138             cout << "Book Status - Copy " << j+1 << ": ";
1139             if(nextCopy -> Status == 1) { //Check if the status is 1
1140                 cout << "Available" << endl;
1141             } else {
1142                 cout << "Unavailable" << endl;
1143             }
1144             nextCopy = nextCopy -> next; //Move to the next copy
1145         }
1146         cout <<endl;
1147     }
1148 };
```

Figure 41: Search_Title method - bookList

The Figure 41 shown above is the Search_Title method. Line 1115 to 1116 are the variable declaration. Line 1117 to 1124 is to iterate the whole book linked list to check if the given book title is existed. Line 1119 to 1123 will be executed only if the book is not pointing to NULL. Line 1119 is to check if the given book title is matched with the book title in book linked list by converting into lower cases, if the book title is found, set flag variable to be true and break the iteration as shown in Line 1120 to 1121. Line 1123 is to move to the next book. Line 1127 to 1128 will be executed only If the flag is false value, which means the book is not found in the Line 1117 to 1124. Line 1130 to 1146 will be executed if the flag value is true. Line 1136 is another pointer declaration. Line 1138 to 1144 will be executed if the value of j is smaller than MAX_COPIES, where MAX_COPIES is a constant, and contains a value of 2. Line 1139 is to check if the book status is equal to 1, Line 1140 will be executed, otherwise, Line 1142 will be executed. Line 1144 is to move to the next copy and add 1 as increment to j variable.

Result

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 2
You want to search a book based on:
1 - Search based on Category
2 - Search based on Genre
3 - Search based on Title
Your option is: 3
Please enter the book title: harry potter


Title: HARRY POTTER
Author: J. K. ROWLING
Genre: FANTASY
Category: FICTION
ISBN: 6808
Book Status - Copy 1: Available
Book Status - Copy 2: Available
```

Figure 42: Sample output 1 - Search_Title method - bookList

The Figure 42 shown above is the sample output for Search_Title method. After entering the search print book section, the user is required to enter a number from 1 to 3 to search the book. If the user wants to search based on title, then the user can enter 3 as the input. The user will then be required to enter the book title, if the book title is found within the book linked list, then the result will be shown directly.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 2
You want to search a book based on:
1 - Search based on Category
2 - Search based on Genre
3 - Search based on Title
Your option is: 3
Please enter the book title: HARRY
THERE IS NO HARRY BOOK IN LIBRARY
```

Figure 43: Sample output 2 - Search_Title method - bookList

The Figure 43 shown above is another sample output for Search_Title method. After entering the search print book section, the user is required to enter a number from 1 to 3 to search the book. If the user wants to search based on title, then the user can enter 3 as the input. The user will then be required to enter the book title, if the book title is not found within the book linked list, then the error message will be shown below.

3.1.4 Update books' information

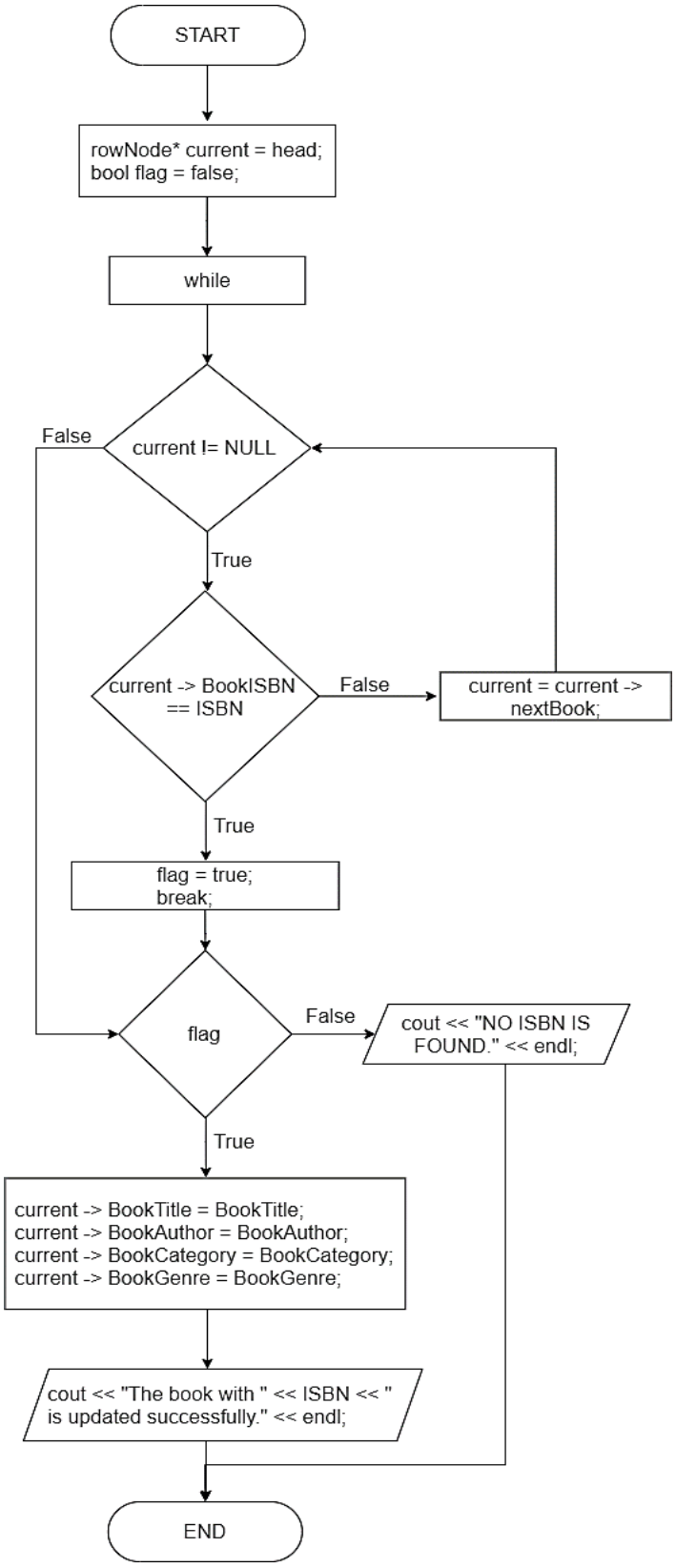


Figure 44: Books - Update book's information flowchart

Code Snippet

```
1150 void BookList::UpdateBookInfo(string BookTitle, string BookAuthor, string BookCategory, string BookGenre, int ISBN) {
1151     rowNode* current = head; //Copy value of head into current
1152     bool flag = false;
1153     while(current != NULL) { //Iterate if the current is not NULL
1154         if(current -> BookISBN == ISBN) { //Check if the given ISBN is matched with the ISBN of book list
1155             flag = true; //Set flag to be true
1156             break; //Break the iteration
1157         }
1158         current = current -> nextBook; //Move to the next book
1159     }
1160 
1161     if(flag) { //If the book is found
1162         current -> BookTitle = BookTitle;
1163         current -> BookAuthor = BookAuthor;
1164         current -> BookCategory = BookCategory;
1165         current -> BookGenre = BookGenre;
1166         cout << "The book with " << ISBN << " is updated successfully.";
1167         cout << endl;
1168     } else {
1169         cout << "NO ISBN IS FOUND.";
1170         cout << endl;
1171     }
1172 };
```

Figure 45: UpdateBookInfo method - bookList

The Figure 45 shown above is the UpdateBookInfo method. Line 1151 to 1152 are the variable declaration. Line 1153 to 1159 is to check if the given ISBN is matched within the book linked list. Line 1154 to 1158 will be executed if the book is not pointing to NULL. Line 1154 is to check if the given ISBN is matched with the book ISBN in linked list. Assign a true value to flag and break the iteration as shown in Line 1155 and 1156. Line 1158 is to move to the next book. Line 1162 to 1167 will be executed only if the flag value is true, where the given ISBN is found in the book linked list as shown in Line 1153 to 1158. Assigning the value to node by referencing to the given parameters. Line 1169 to 1170 will be executed only if the flag value is false.

Result

```
BOOK MENU:
1 – Add a new book
2 – Search
3 – Print
4 – Update book information
5 – Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 4


BOOK TITLE                      BOOK AUTHOR                     ISBN
HARRY POTTER                    J. K. ROWLING                   6808
MEIN KAMPF                      ADOLF HITLER                    5250
13 REASONS WHY                  JAY ASHER                       74
THE INVISIBLE MAN               H. G. WELLS                     3659
THE HUNGER GAMES                SUZANNE COLLINS                 8931
I HAVE A DREAM                  MARTIN LUTHER KING JR.          1273
CRIME AND PERIODICALS           NOVA EVERLY                     7545
A NEW EARTH                     ECKHART TOLLE                   879
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL                 7924
This is a new book              This has no author              7710

Please enter a book ISBN: 7710
Please enter the book title: A thousand love
Please enter the book author: John Doe
Is the new book Fiction? (1 – YES / 0 – NO): 1
1 – Fantasy
2 – Science
3 – Historical
4 – Realistic
5 – Fan
Your option is: 3
The book with 7710 is updated successfully.
```

Figure 46: Sample output 1 - UpdateBookInfo method - bookList

The Figure 46 shown above is the sample output for UpdateBookInfo method. The user is required to enter 4 to do an update for book. After entering 4, the user is required to provide an ISBN of the book to make update, the ISBN should be within the book linked list as shown in the picture. If the ISBN is found within the book linked list, then the user can modify towards the book.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 4

BOOK TITLE                      BOOK AUTHOR                 ISBN
HARRY POTTER                    J. K. ROWLING               6808
MEIN KAMPF                      ADOLF HITLER                5250
13 REASONS WHY                  JAY ASHER                   74
THE INVISIBLE MAN               H. G. WELLS                 3659
THE HUNGER GAMES                SUZANNE COLLINS             8931
I HAVE A DREAM                  MARTIN LUTHER KING JR.      1273
CRIME AND PERIODICALS           NOVA EVERLY                 7545
A NEW EARTH                     ECKHART TOLLE               879
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL             7924
A THOUSAND LOVE                 JOHN DOE                    7710

Please enter a book ISBN: 6888
No ISBN is found.
```

Figure 47: Sample output 2 - UpdateBookInfo method - bookList

The Figure 47 shown above is another sample output for UpdateBookInfo method. The user is required to enter 4 to do an update for book. After entering 4, the user is required to provide an ISBN of the book to make update, the ISBN should be within the book linked list as shown in the picture. If the ISBN is not found within the book linked list, then the error message will be shown directly to the user saying that the given ISBN is not found in the book linked list.

3.1.5 Delete a book

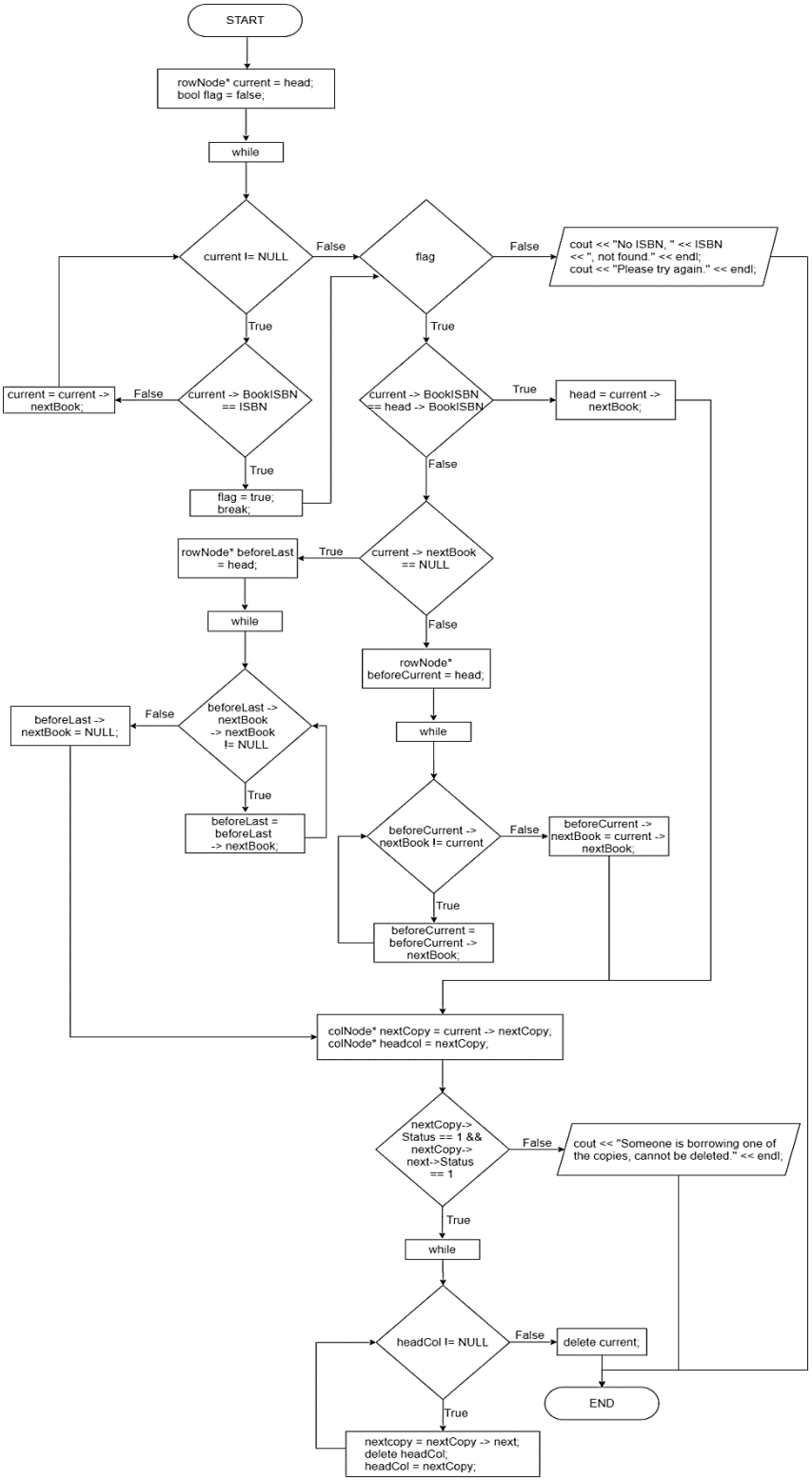


Figure 48: Books - Delete book flowchart

LINK:

https://drive.google.com/file/d/1-MvuXVhndy1rhjVvrPcZil9yRNDpblk3/view?usp=sharing

Code Snippet

```
1205 void BookList::DeleteBook(int ISBN) {
1206     rowNode* current = head; //Copy value of head into current
1207     bool flag = false;
1208     while(current != NULL) { //Iterate if the current is not NULL
1209         if(current -> BookISBN == ISBN) { //Check if the given ISBN is matched with the Book ISBN of book list
1210             flag = true; //Set the flag to be true
1211             break; //Break the iteration
1212         }
1213         current = current -> nextBook; //Move to the next book
1214     }
1215 
1216     if(flag) { //If the book ISBN is found
1217         if(current-> BookISBN == head -> BookISBN) {
1218             head = current->nextBook;
1219         } else if (current->nextBook == NULL) {
1220             rowNode* beforelast = head;
1221             while(beforelast -> nextBook -> nextBook != NULL) {
1222                 beforelast = beforelast -> nextBook;
1223             }
1224             beforelast->nextBook = NULL;
1225         } else {
1226             rowNode* beforeCurrent = head;
1227             while(beforeCurrent->nextBook != current) {
1228                 beforeCurrent = beforeCurrent -> nextBook;
1229             }
1230             beforeCurrent -> nextBook = current -> nextBook;
1231         }
1232 
1233         colNode* nextCopy = current -> nextCopy;
1234         colNode* headCol = nextCopy;
1235         if(nextCopy->Status == 1 && nextCopy -> next->Status == 1) { //Check if both copies are available
1236             //Delete all the copies
1237             while(headCol != NULL) {
1238                 nextCopy = nextCopy -> next;
1239                 delete headCol;
1240                 headCol = nextCopy;
1241             }
1242             delete current;
1243             cout << endl;
1244         } else {
1245             cout << "Someone is borrowing one of the copies, cannot be deleted.";
1246             cout <<endl;
1247         }
1248 
1249     } else {
1250         cout << "No ISBN, " << ISBN << ", not found." << endl;
1251         cout << "Please try again." << endl;
1252     }
1253 
1254 };
```

Figure 49: DeleteBook method - bookList

The Figure 49 shown above is the DeleteBook method. Line 1206 to 1207 are the variable declaration. Line 1208 to 1214 is to check if the given ISBN is found within the book linked list. Line 1209 to 1213 will be executed only if the book is not pointing to NULL. Line 1209 is to check if the given ISBN is matched with the book ISBN in book linked list, set flag to be true and break the iteration as shown in Line 1210 and 1211. Line 1213 is to move to the next book. Line 1217 to 1243 will be executed only if the flag value is true, where the given ISBN is found in the book linked list as shown in Line 1208 to 1214. Line 1217 to 1231 is to validate if the position of the given ISBN. If the given ISBN is at the first node of book linked list, then 1218 will be executed, pointing head to the second node. If the given ISBN is pointing to last node of book linked list, then iterate the book linked list to the second last, and assigning NULL value to the second last node. If the first two conditions do not match, which means the given ISBN is at arbitrary index, then Line 1227 to 1229 is to iterate until the arbitrary index and then make the changes as shown in Line 1230. Line 1233 to 1234 are pointer declaration. Line 1235 to 1246 is to check if the copies of the given ISBN has been borrowed by patron, by checking

statuses of two copies as shown in Line 1235. If both are available (1), which means both copies are not borrowed by patron, then Line 1237 to 1242 is to delete the copy, and Line 1242 is to delete the book. If either book copies is 0, then Line 1245 to 1246 will be executed. Line 1250 to 1251 will be executed if the flag value is false, which represents the given ISBN is not found within the book linked list.

Result

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 5

BOOK TITLE                      BOOK AUTHOR                     ISBN
HARRY POTTER                    J. K. ROWLING                   6808
MEIN KAMPF                      ADOLF HITLER                    5250
13 REASONS WHY                  JAY ASHER                       74
THE INVISIBLE MAN               H. G. WELLS                     3659
THE HUNGER GAMES                SUZANNE COLLINS                 8931
I HAVE A DREAM                  MARTIN LUTHER KING JR.          1273
CRIME AND PERIODICALS           NOVA EVERLY                     7545
A NEW EARTH                     ECKHART TOLLE                   879
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL                 7924
A thousand love                 John Doe                        7710

Please enter a book ISBN that you wish to delete: 7710

BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 3
1 - Print all book title
2 - Print Available book title
Your option is: 2

BOOK TITLE                      BOOK AUTHOR                     GENRE          CATEGORY       ISBN      AVAILABILITY
HARRY POTTER                    J. K. ROWLING                   FANTASY        FICTION        6808      2
MEIN KAMPF                      ADOLF HITLER                    HISTORICAL     FICTION        5250      2
13 REASONS WHY                  JAY ASHER                       REALISTIC      FICTION        74        2
THE INVISIBLE MAN               H. G. WELLS                     SCIENCE        FICTION        3659      2
THE HUNGER GAMES                SUZANNE COLLINS                 NARRATIVE      NON-FICTION    8931      2
I HAVE A DREAM                  MARTIN LUTHER KING JR.          BIOGRAPHY      NON-FICTION    1273      2
CRIME AND PERIODICALS           NOVA EVERLY                     PERIODICALS    NON-FICTION    7545      2
A NEW EARTH                     ECKHART TOLLE                   SELF-HELP      NON-FICTION    879       2
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL                 REFERENCE      NON-FICTION    7924      2
```

Figure 50: Sample output 1 - DeleteBook method - bookList

The Figure 50 shown above is the sample output for DeleteBook method. The user can enter 5 to delete a book based on the ISBN. The ISBN entered by user should be within the given book list. After giving the specific ISBN, the book will then be deleted directly, but the status of both copies should be available, then the operation will be successful. Otherwise, the error message will be shown.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
If you wish to quit Book Section, you may enter 0 *
Your option is: 5

BOOK TITLE                      BOOK AUTHOR                     ISBN
HARRY POTTER                    J. K. ROWLING                   6808
MEIN KAMPF                      ADOLF HITLER                    5250
13 REASONS WHY                  JAY ASHER                       74
THE INVISIBLE MAN               H. G. WELLS                     3659
THE HUNGER GAMES                SUZANNE COLLINS                 8931
I HAVE A DREAM                  MARTIN LUTHER KING JR.          1273
CRIME AND PERIODICALS           NOVA EVERLY                     7545
A NEW EARTH                     ECKHART TOLLE                   879
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL                 7924
A THOUSAND LOVE                 JOHN DOE                        7710

Please enter a book ISBN that you wish to delete: 7770
No ISBN, 7770, not found.
Please try again.
```

Figure 51: Sample output 2 - DeleteBook method - bookList

The Figure 51 shown above is another sample output for DeleteBook method. The user can enter 5 to delete a book based on the ISBN. The ISBN entered by user should be within the given book list. If the user enters an ISBN which is not within the book linked list, the error message will then be shown to the user directly.

```
BOOK MENU:
1 - Add a new book
2 - Search
3 - Print
4 - Update book information
5 - Delete a book
Your option is: 5

BOOK TITLE                      BOOK AUTHOR                     ISBN
HARRY POTTER                    J. K. ROWLING                   6808
MEIN KAMPF                      ADOLF HITLER                    5250
13 REASONS WHY                  JAY ASHER                       74
THE INVISIBLE MAN               H. G. WELLS                     3659
THE HUNGER GAMES                SUZANNE COLLINS                 8931
I HAVE A DREAM                  MARTIN LUTHER KING JR.          1273
CRIME AND PERIODICALS           NOVA EVERLY                     7545
A NEW EARTH                     ECKHART TOLLE                   879
THE HERO WITH A THOUSAND FACES  JOSEPH CAMPBELL                 7924

Please enter a book ISBN that you wish to delete: 74
Someone is borrowing one of the copies, cannot be deleted.
```

Figure 52: Sample output 3 - DeleteBook method - bookList

The Figure 52 shown above is another sample output for DeleteBook method. The user can enter 5 to delete a book based on the ISBN. The ISBN entered by user should be within the given book list. If the user enters an ISBN which is within the book linked list, but one of the copies is currently borrowed by patron, then the error message will be shown to the user directly.

3.2 Patron list

3.2.1 Patron registration

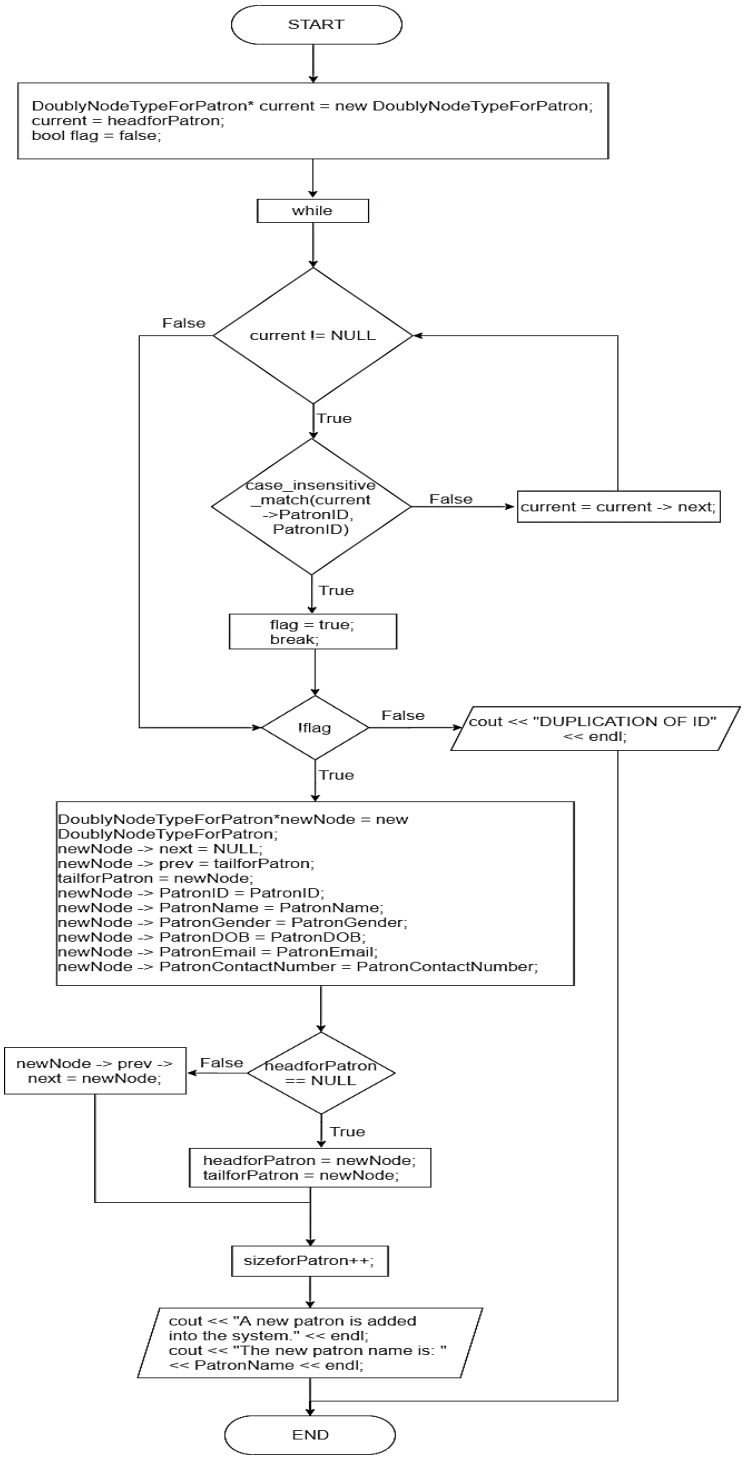


Figure 53: Patrons - Patron registration flowchart

LINK:

https://drive.google.com/file/d/1dOndsPsVqtBKeKSg7rwiHIneJTpE_kqg/view?usp=sharing

Code Snippet

```
183 void PatronList::AddPatron(string PatronID, string PatronName, string PatronGender, string PatronDOB, string PatronEmail, string
        PatronContactNumber) {
184
185     //Copy value of headforPatron into current
186     DoublyNodeTypeForPatron* current = new DoublyNodeTypeForPatron;
187     current = headforPatron;
188     bool flag = false;
189
190     //Iterate it if the current traversal is not NULL
191     while (current != NULL) {
192         //Matching the patron link list and the given patron ID
193         if (case_insensitive_match(current->PatronID, PatronID)) {
194             //Set true value to flag to indicate the patron ID is found, and break the iteration
195             flag = true;
196             break;
197         }
198         current = current->next;
199     }
200
201     //If no duplicate patron ID is detected
202     if (!flag) {
203         //Create a new node for patron
204         DoublyNodeTypeForPatron* newNode = new DoublyNodeTypeForPatron;
205         newNode->next = NULL;
206         newNode->prev = tailforPatron;
207         tailforPatron = newNode;
208         newNode->PatronID = PatronID;
209         newNode->PatronName = PatronName;
210         newNode->PatronGender = PatronGender;
211         newNode->PatronDOB = PatronDOB;
212         newNode->PatronEmail = PatronEmail;
213         newNode->PatronContactNumber = PatronContactNumber;
214          //If the headforPatron is pointing to NULL
215          if (headforPatron == NULL) {
216              headforPatron = newNode;
217              tailforPatron = newNode;
218
219          }
220          else {
221              newNode->prev->next = newNode;
222          }
223          //Increase the size of Patron list
224          sizeforPatron++;
225          cout << "A new patron is added into the system." << endl;
226          cout << "The new patron name is: " << PatronName << endl;
227     }
228     //If detected
229     else {
230         cout << "DUPLICATION OF ID";
231         cout << endl;
232     }
233 };
```

Figure 54: AddPatron method - PatronList

The Figure 54 shown above is the AddPatron method. Line 186 to 188 are the variable declaration. Line 191 to 199 is to check if the given patron ID is exactly the same as the data that stored in patron linked list by converting into lower cases. Set the flag to be true to indicate the same patron ID is detected and then break the iteration. If the Line 193 does not meet, then the flag is false as default. Line 204 to 226 will be executed only if the given patron ID is not detected as the same patron ID. Line 216 to 223 are to add the new node into patron linked list, by checking if the head pointer is pointing to a node. If the head pointer is pointing to NULL, then Line 217 to 218 will be executed, assign the new node to head and tail pointers. Otherwise, assign the new node to the last node of patron linked list. Line 230 will be executed only if the same patron ID is detected in Line 193 to 196.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
Your option is: 1
Please enter your Patron ID: JD001
Please enter your Patron Name: John Doe
Is the patron Male? (1 – YES / 0 – NO): 1
Please enter your DOB: 01/01/2020
Please enter your Email: Johndoe@gmail.com
Please enter your Contact Number: 60165001990
A new patron is added into the system.
The new patron name is: John Doe
```

Figure 55: Sample output 1 - AddPatron method - PatronList

The Figure 55 is the sample output for AddPatron method. The user will need to enter 1 to add a new patron. Patron ID, Patron Name, Data of Birth, Email and Contact Number can be in string types, and the Patron Gender can be entered by using numeric which is either 1 or 0. After entering all the details, then the successful message will be prompted to the user saying that a new patron is added into the system.

```
PATRON ID          PATRON NAME                GENDER         DATE OF BIRTH      PATRON EMAIL                TEL. NUMBER
HKJ001             HEN KIAN JUN               MALE           29/09/1997         KJHEN0929@GMAIL.COM         60139338783
LWY002             LEE WAI YEN                FEMALE         13/10/1998         LWYEN1013@GMAIL.COM         6014848592
LBK003             LOW BOON KIAT              MALE           11/22/1999         BOONKIAT1122@GMAIL.COM      6012395858
NZJ004             NG ZHENG JUE               MALE           20/06/1999         ZHENGJUE1999@GMAIL.COM      6019483838
LYK005             LIM YI KANG                MALE           23/03/1999         YIKANG123@GMAIL.COM         6012394859
YMW006             YONG MUN WEI               MALE           17/10/1997         YONGMUNWEI@GMAIL.COM        6019273828
TKM007             TEOH KHEN MENG             MALE           17/10/1997         KHENMENG1997@GMAIL.COM      6019212328
TYJ008             LIM AH BENG                MALE           24/10/1998         JIFFRED1998@GMAIL.COM       6019272322

1 - Print from beginning
2 - Print from behind
If you wish to quit Print Section, you may enter 0 *
Your option is: 0

PATRON MENU:
1 - Add a new patron
2 - Search
3 - Print
4 - Update a patron information
5 - Get patron with active book borrow
6 - Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 1
Please enter your Patron ID: HKJ001
Please enter your Patron Name: JOHN SMITH
Is the patron Male? (1 - YES / 0 - NO): 1
Please enter your DOB: 29/09/1994
Please enter your Email: JOHNSMITH@GMAIL.COM
Please enter your Contact Number: 60139332837
DUPLICATION OF ID
```

Figure 56: Sample output 2 - AddPatron method - PatronList

The Figure 56 shown above is another sample output for AddPatron method. This is to show that if the user enters the same patron ID into the system. To prevent duplicate patron ID store into the system, an error message will be prompted to the user indicating that the new added patron is similar to the patron existed in the system.

3.2.2 Display the last 10 books borrowed by a patron

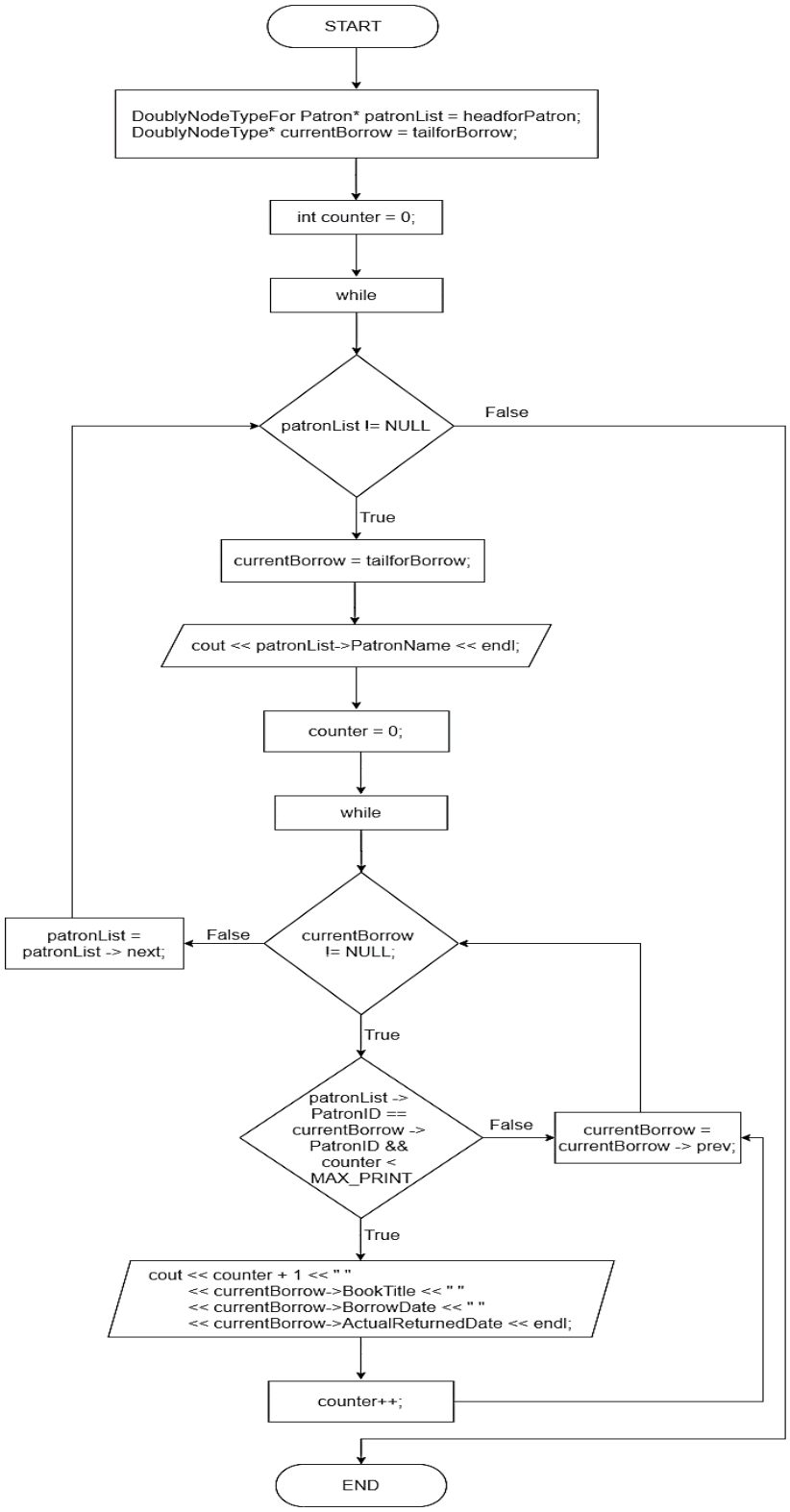


Figure 57: Patrons - Display the last 10 books borrowed by a patron flowchart

MAX_PRINT is a constant variable with the value of 10, therefore this flowchart will be printing only the maximum of 10 records of book borrowed by a patron.

Code Snippet

```
234 void PatronList::PrintLast10BooksByAPatron() {
235     //Copy value of headforPatron into patronList
236     DoublyNodeTypeForPatron* patronList = headforPatron;
237     //Copy value of tailforBorrow into currentBorrow
238     DoublyNodeTypeForBorrow* currentBorrow = tailforBorrow;
239 
240     //Instantiated counter to be 0
241     int counter = 0;
242 
243     //Iterate it if the patronlist is not NULL
244     while (patronList != NULL) {
245         currentBorrow = tailforBorrow;
246         cout << patronList->PatronName << endl;
247         counter = 0;
248         //Iterate it if the currentBorrow is not NULL
249         while (currentBorrow != NULL) {
250             //If the Patron ID of patronlist is matched to Patron IID of currentBorrow and the counter is below 10
251             if (patronList->PatronID == currentBorrow->PatronID && counter < MAX_PRINT) {
252                 //Print the records
253                 cout << counter + 1 << " " << currentBorrow->BookTitle << " " << currentBorrow->BorrowDate << " " <<
                        currentBorrow->ActualReturnedDate << endl;
254                 //Add 1 to counter
255                 counter++;
256             }
257             //Move to the previous borrow
258             currentBorrow = currentBorrow->prev;
259         }
260         cout << endl;
261         //Move to the next patron
262         patronList = patronList->next;
263     }
264 }
```

Figure 58: PrintLast10BooksByAPatron method - patronList

The Figure 58 shown above is PrintLast10BooksByAPatron method. Line 236 to 241 are variable declarations. Line 244 to 263 is to print all the patron list, but the Line 249 to 259 is to print all the borrow list, the borrow list will be printed only if the patron IDs in patron list are matched with the patron IDs in the borrow list and the counter is below 10 in the Line 251. If Line 251 is true, then add 1 as increment to counter. Line 258 is to move the pointer to the previous node of borrow list, while the Line 262 is to move the pointer to the next patron. Therefore, the borrow records will be printed in descending orders while the patron name will be printed in ascending orders.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
Your option is: 6
HEN KIAN JUN
1 THE HERO WITH A THOUSAND FACES 30/8/2020 NULL
2 I HAVE A DREAM 30/8/2020 30/8/2020
3 MEIN KAMPF 30/8/2020 30/8/2020
4 13 REASONS WHY 30/8/2020 NULL

LEE WAI YEN
1 THE INVISIBLE MAN 30/8/2020 NULL

LOW BOON KIAT
1 THE INVISIBLE MAN 30/8/2020 NULL

NG ZHENG JUE
1 HARRY POTTER 30/8/2020 NULL

LIM YI KANG
1 A NEW EARTH 30/8/2020 NULL

YONG MUN WEI

TEOH KHEN MENG
1 HARRY POTTER 30/8/2020 NULL

LIM AH BENG
1 MEIN KAMPF 30/8/2020 NULL

John Doe
1 A NEW EARTH 30/8/2020 NULL
2 13 REASONS WHY 30/8/2020 NULL
```

Figure 59: Sample output 1 - PrintLast10BooksByAPatron method - patronList

The Figure 59 shown above is the sample output for PrintLast10BooksByAPatron method. The user will need to enter 6 to proceed to the print last 10 books section.

3.2.3 View all patrons with active book borrowed

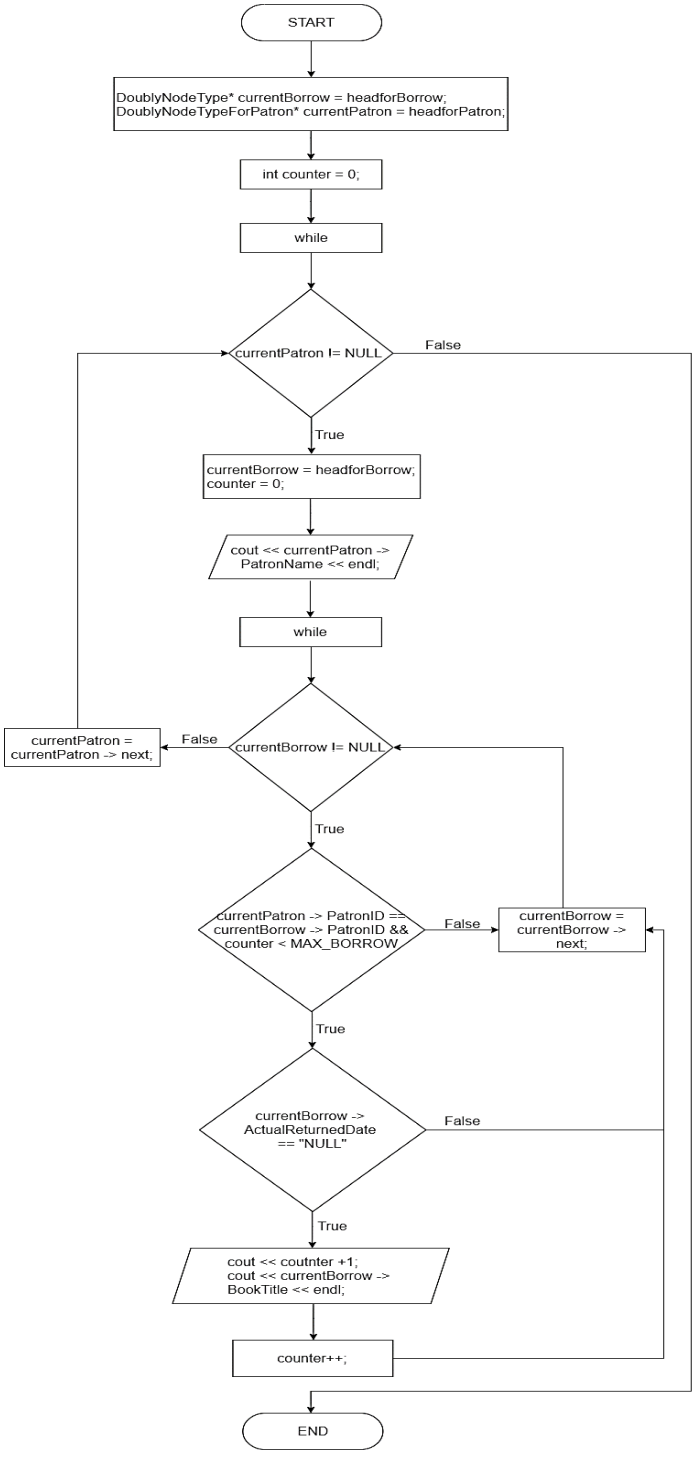


Figure 60: Patrons - View all patrons with active book borrowed flowchart

MAX_BORROW is a constant variable with the value of 3, therefore this flowchart will be printing only the maximum of 3 records of patron is currently borrowing.

LINK:

https://drive.google.com/file/d/1B0GyVZG5mS8E6KoIAz9vDqrPpMC4LJCq/view?usp=sharing

Code Snippet

```
266 void PatronList::getPatronwithActiveBookBorrow() {
267     //Copy value of headforBorrow into currentBorrow
268     DoublyNodeTypeForBorrow* currentBorrow = headforBorrow;
269     //Copy value of headforPatron into currentPatron
270     DoublyNodeTypeForPatron* currentPatron = headforPatron;
271 
272     //Instantiated counter to be 0
273     int counter = 0;
274 
275     //Iterate if the currentPatron is not NULL
276     while (currentPatron != NULL) {
277         currentBorrow = headforBorrow;
278         counter = 0;
279         cout << currentPatron->PatronName << endl;
280         //Iterate if the currentBorrow is not NULL
281         while (currentBorrow != NULL) {
282             //Check if the patron ID of currentPatron with Patron ID of currentBorrow and the counter is below 10
283             if (currentPatron->PatronID == currentBorrow->PatronID && counter < MAX_BORROW) {
284                 //Check if the actual returned date is NULL
285                 if (currentBorrow->ActualReturnedDate == "NULL") {
286                     cout << counter + 1 << " " << currentBorrow->BookTitle << endl;
287                     //Increment 1 to counter
288                     counter++;
289                 }
290             }
291             //Move to the next borrow
292             currentBorrow = currentBorrow->next;
293         }
294         cout << endl;
295         //Move to the next patron
296         currentPatron = currentPatron->next;
297     }
298 };
```

Figure 61: getPatronwithActiveBookBorrow method - patronList

The Figure 61 shown above is the getPatronwithActiveBookBorrow method. Line 268 to 273 are the variable declarations. Line 277 to 296 will be executed only if the patron linked list is not pointing to NULL. Line 283 to 292 will be executed only if the borrow linked list is not pointing to NULL. Before executing Line 286 to 288, Line 285 will check if the actual returned date of the borrow is NULL. The actual returned date is to check if the book has been returned. Line 292 is to move to the next borrow list, while Line 296 is to move to the next patron.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 5
HEN KIAN JUN
1 13 REASONS WHY
2 THE HERO WITH A THOUSAND FACES

LEE WAI YEN
1 THE INVISIBLE MAN

LOW BOON KIAT
1 THE INVISIBLE MAN

NG ZHENG JUE
1 HARRY POTTER

LIM YI KANG
1 A NEW EARTH

YONG MUN WEI

TEOH KHEN MENG
1 HARRY POTTER

LIM AH BENG
1 MEIN KAMPF

John Doe
1 13 REASONS WHY
2 A NEW EARTH
```

Figure 62: Sample output 1 - getPatronwithActiveBookBorrow method - patronList

The Figure 62 shown above is another sample output for getPatronwithActiveBookBorrow method. The user will need to enter 5 to proceed to get patron with active book borrow section.

3.2.4 Search patron

3.2.4.1 Based on ID

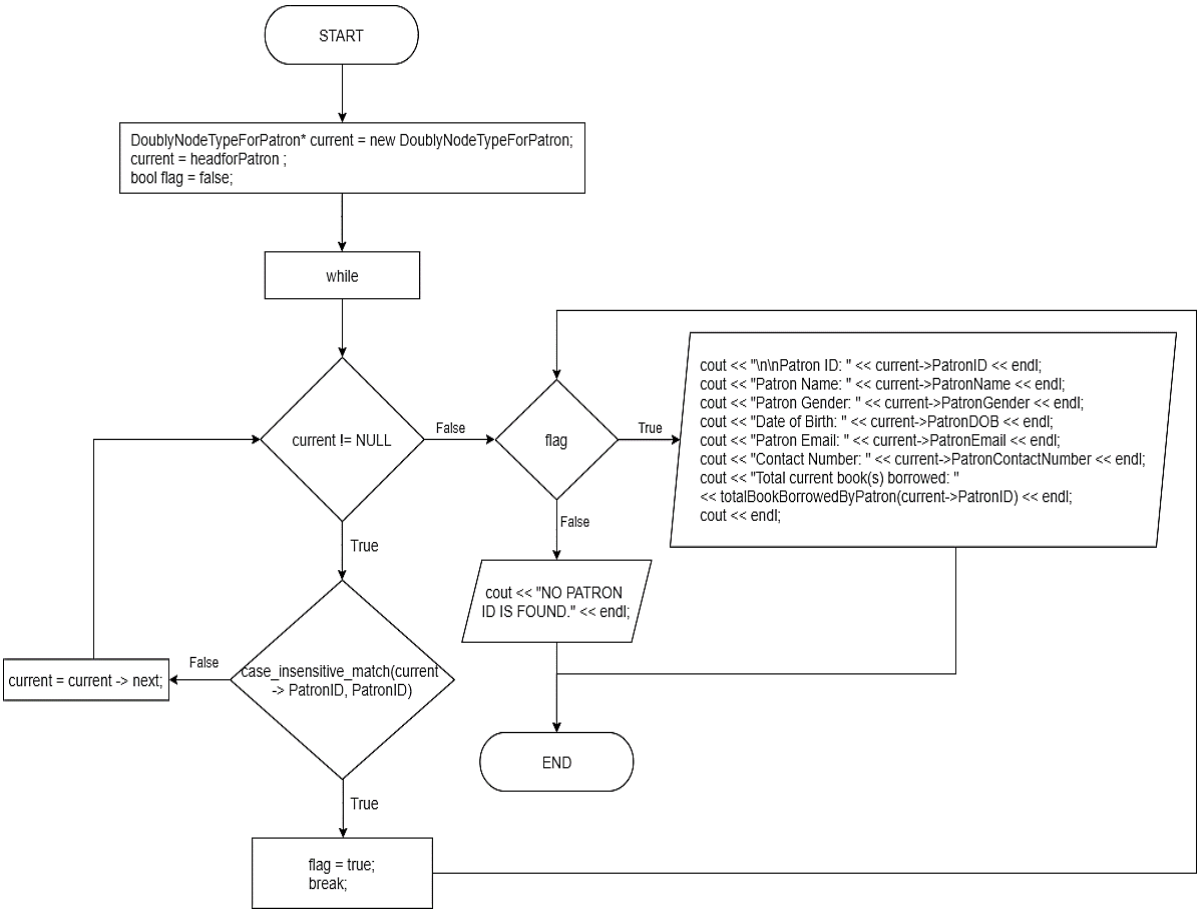


Figure 63: Patrons - Search patron (ID) flowchart

The Figure 63 flowchart above is the flow of searching a patron based on patron ID in LMS. However, the algorithms flowchart is also applicable to the search patron name.

case_insensitive_match(string, string) is a method in which will be passing this two values to do comparison without the case sensitive.

Code Snippet

```
321 void PatronList::SearchPatron_ID(string PatronID) {
322     //Copy value of headforPatron into current
323     DoublyNodeTypeForPatron* current = new DoublyNodeTypeForPatron;
324     current = headforPatron;
325 
326     bool flag = false;
327 
328     //Iterate if the current is not NULL
329     while (current != NULL) {
330         //Check if the given Patron ID is matched the Patron ID in Patron link list
331         if (case_insensitive_match(current->PatronID, PatronID)) {
332             //Set true value to flag and break the iteration
333             flag = true;
334             break;
335         }
336         //Move to the next patron if not found
337         current = current->next;
338     }
339 
340     //If flag equals to true
341     if (flag) {
342         //Print the records
343         cout << "\n\nPatron ID: " << current->PatronID << endl;
344         cout << "Patron Name: " << current->PatronName << endl;
345         cout << "Patron Gender: " << current->PatronGender << endl;
346         cout << "Date of Birth: " << current->PatronDOB << endl;
347         cout << "Patron Email: " << current->PatronEmail << endl;
348         cout << "Contact Number: " << current->PatronContactNumber << endl;
349         //Call method to return the value of book borrowed by passing patron ID
350         cout << "Total current book(s) borrowed: " << totalBookBorrowedByPatron(current->PatronID) << endl;
351         cout << endl;
352     }
353     //If flag equals to false, means the given patron ID is not found
354     else {
355         cout << "NO PATRON ID IS FOUND.";
356         cout << endl;
357     }
358 }
```

Figure 64: SearchPatron_ID method - patronList

The Figure 64 shown above is the SearchPatron_ID method. Line 323 to 326 are the variable declarations. Line 331 to 337 will be executed only if the patron is not pointing to NULL. Line 331 is to compare the given patron ID with the patron ID in patron linked list by converting into lower cases, set flag to be true and break the iteration if the given ID is the same as the data that stored in patron linked list. Line 337 is to move to the next patron, however, if the Line 331 is true, then the whole while loop will no longer iterate. Line 343 to 351 will be executed only if the given patron ID is existed in the patron linked list, which is based on the flag value. Line 350 is to call another method to obtain the number of book borrowed by that patron by passing patron ID. Line 355 will be executed only if the flag value is false, flag value is false refers to the criteria does not meet in the Line 331, which means the given patron ID is not found.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 2
You want to search a patron based on:
1 – Search based on Patron ID
2 – Search based on Patron Name
Your option is: 1
Please enter patron ID: hkj001


Patron ID: HKJ001
Patron Name: HEN KIAN JUN
Patron Gender: MALE
Date of Birth: 29/09/1997
Patron Email: KJHEN0929@GMAIL.COM
Contact Number: 60139338783
Total current book(s) borrowed: 2
```

Figure 65: Sample output 1 - SearchPatron_ID method - patronList

The Figure 65 shown above is the sample output for SearchPatron_ID method. After entering the search print patron section, the user is required to enter a number from either 1 or 2 to search the patron. If the user wants to search the patron based on patron ID, then the user can enter 1 as the input. The user will then is required to enter the patron ID, if the patron ID is found within the patron list, then the result will be shown directly.

```
PATRON ID          PATRON NAME                GENDER         DATE OF BIRTH      PATRON EMAIL                TEL. NUMBER
HKJ001             HEN KIAN JUN               MALE           29/09/1997         KJHEN0929@GMAIL.COM         60139338783
LWY002             LEE WAI YEN                FEMALE         13/10/1998         LWYEN1013@GMAIL.COM         6014848592
LBK003             LOW BOON KIAT              MALE           11/22/1999         BOONKIAT1122@GMAIL.COM      6012395858
NZJ004             NG ZHENG JUE               MALE           20/06/1999         ZHENGJUE1999@GMAIL.COM      6019483838
LYK005             LIM YI KANG                MALE           23/03/1999         YIKANG123@GMAIL.COM         6012394859
YMW006             YONG MUN WEI               MALE           17/10/1997         YONGMUNWEI@GMAIL.COM        6019273828
TKM007             TEOH KHEN MENG             MALE           17/10/1997         KHENMENG1997@GMAIL.COM      6019212328
TYJ008             LIM AH BENG                MALE           24/10/1998         JIFFRED1998@GMAIL.COM       6019272322

1 - Print from beginning
2 - Print from behind
If you wish to quit Print Section, you may enter 0 *
Your option is: 0

PATRON MENU:
1 - Add a new patron
2 - Search
3 - Print
4 - Update a patron information
5 - Get patron with active book borrow
6 - Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 2
You want to search a patron based on:
1 - Search based on Patron ID
2 - Search based on Patron Name
Your option is: 1
Please enter patron ID: HKJ002
NO PATRON ID IS FOUND.
```

Figure 66: Sample output 2 - SearchPatron_ID method - patronList

The Figure 66 shown above is another sample output for SearchPatron_ID method. After entering the search print patron section, the user is required to enter a number either 1 or 2 to search the patron. If the user wants to search based on patron ID, then the user can enter 1 as the input. The user will then is required to enter the patron ID, if the patron ID is not found within the patron list, then the error message will be shown below.

3.2.4.2 Based on Name

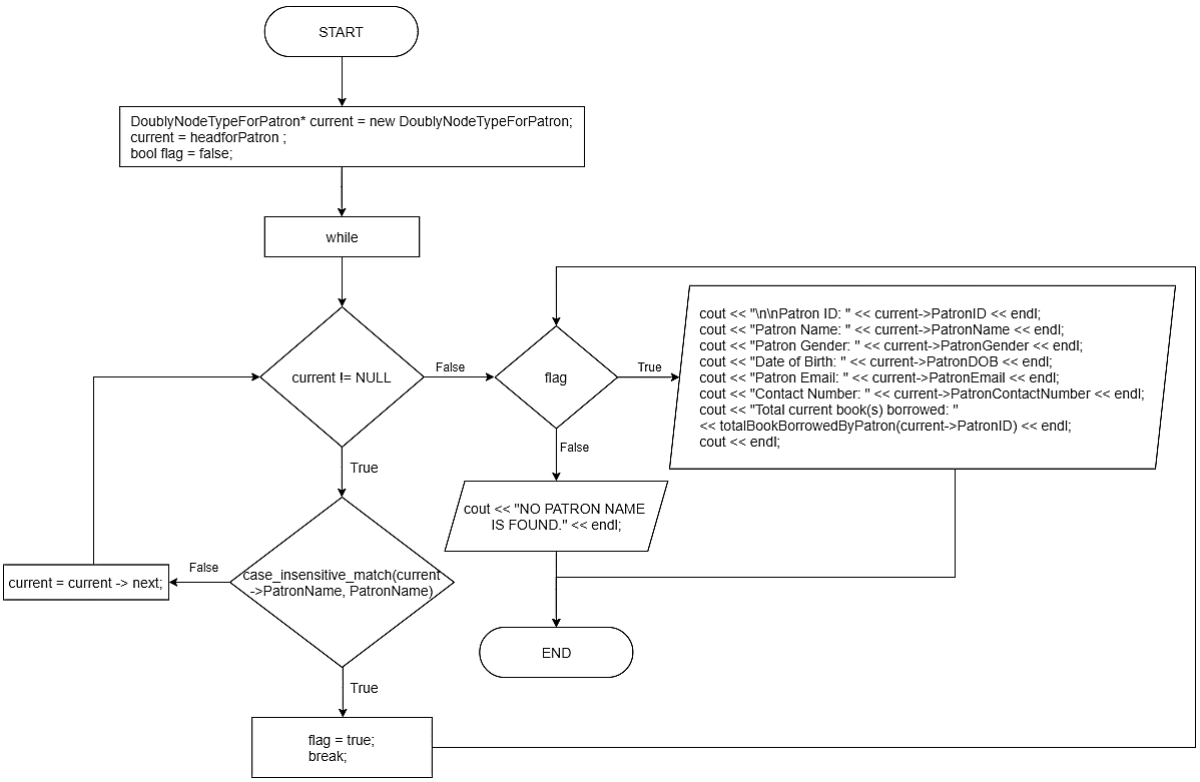


Figure 67: Patrons - Search patron (NAME) flowchart

Code Snippet

```
360 void PatronList::SearchPatron_NAME(string PatronName) {
361     //Copy value of headforPatron into current
362     DoublyNodeTypeForPatron* current = new DoublyNodeTypeForPatron;
363     current = headforPatron;
364 
365     bool flag = false;
366 
367     //Iterate if the current is not NULL
368     while (current != NULL) {
369         //Check if the given patron name is matched to the patron name of patron link list
370         if (case_insensitive_match(current->PatronName, PatronName)) {
371             //Set a true value to flag and break the iteration
372             flag = true;
373             break;
374         }
375         //Move to the next patron if not found
376         current = current->next;
377     }
378 
379     //If flag equals to true
380     if (flag) {
381         cout << "\n\nPatron ID: " << current->PatronID << endl;
382         cout << "Patron Name: " << current->PatronName << endl;
383         cout << "Patron Gender: " << current->PatronGender << endl;
384         cout << "Date of Birth: " << current->PatronDOB << endl;
385         cout << "Patron Email: " << current->PatronEmail << endl;
386         cout << "Contact Number: " << current->PatronContactNumber << endl;
387         //Call method to return the value of book borrowed by passing patron ID
388         cout << "Total current book(s) borrowed: " << totalBookBorrowedByPatron(current->PatronID) << endl;
389         cout << endl;
390     }
391     //If flag equals to false, means the given patron name is not found
392     else {
393         cout << "NO PATRON NAME IS FOUND.";
394         cout << endl;
395     }
396 }
```

Figure 68: SearchPatron_NAME method - patronList

The Figure 68 shown above is the SearchPatron_NAME method. Line 362 to 365 are the variable declarations. Line 370 to 376 will be executed only if the patron is not pointing to NULL. Line 370 is to compare the given patron name with the patron name in patron linked list by converting into lower cases, set flag to be true and break the iteration if the given name is the same as the data that stored in patron linked list. Line 376 is to move to the next patron, however, if the Line 370 is true, then the whole while loop will no longer iterate. Line 381 to 389 will be executed only if the given patron name is existed in the patron linked list, which is based on the flag value. Line 388 is to call another method to obtain the number of book borrowed by that patron by passing patron ID. Line 393 will be executed only if the flag value is false, flag value is false refers to the criteria does not meet in the Line 370, which means the given patron name is not found.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 2
You want to search a patron based on:
1 – Search based on Patron ID
2 – Search based on Patron Name
Your option is: 2
Please enter patron Name: john doe


Patron ID: JD001
Patron Name: John Doe
Patron Gender: MALE
Date of Birth: 01/01/2020
Patron Email: Johndoe@gmail.com
Contact Number: 60165001990
Total current book(s) borrowed: 2
```

Figure 69: Sample output 1 - SearchPatron_NAME method - patronList

The Figure 69 shown above is the sample output for SearchPatron_NAME method. After entering the search print patron section, the user is required to enter a number either 1 or 2 to search the patron. If the user wants to search based on name, then the user can enter 2 as the input. The user will then be required to enter the patron name, if the patron name is found within the patron list, then the result will be shown directly.

```
PATRON ID          PATRON NAME                GENDER         DATE OF BIRTH      PATRON EMAIL                TEL. NUMBER
HKJ001             HEN KIAN JUN               MALE           29/09/1997         KJHEN0929@GMAIL.COM         60139338783
LWY002             LEE WAI YEN                FEMALE         13/10/1998         LWYEN1013@GMAIL.COM         6014848592
LBK003             LOW BOON KIAT              MALE           11/22/1999         BOONKIAT1122@GMAIL.COM      6012395858
NZJ004             NG ZHENG JUE               MALE           20/06/1999         ZHENGJUE1999@GMAIL.COM      6019483838
LYK005             LIM YI KANG                MALE           23/03/1999         YIKANG123@GMAIL.COM         6012394859
YMW006             YONG MUN WEI               MALE           17/10/1997         YONGMUNWEI@GMAIL.COM        6019273828
TKM007             TEOH KHEN MENG             MALE           17/10/1997         KHENMENG1997@GMAIL.COM      6019212328
TYJ008             LIM AH BENG                MALE           24/10/1998         JIFFRED1998@GMAIL.COM       6019272322

1 - Print from beginning
2 - Print from behind
If you wish to quit Print Section, you may enter 0 *
Your option is: 0

PATRON MENU:
1 - Add a new patron
2 - Search
3 - Print
4 - Update a patron information
5 - Get patron with active book borrow
6 - Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 2
You want to search a patron based on:
1 - Search based on Patron ID
2 - Search based on Patron Name
Your option is: 2
Please enter patron Name: HEN KIAN JAN
NO PATRON NAME IS FOUND.
```

Figure 70: Sample output 2 - SearchPatron_NAME method - patronList

The Figure 70 shown above is another sample output for SearchPatron_NAME method. After entering the search print patron section, the user is required to enter a number either 1 or 2 to search the patron. If the user wants to search based on name, then the user can enter 2 as the input. The user will then be required to enter the patron name, if the patron name is not found within the patron list, then the error message will be shown below.

3.2.5 Print patron list

3.2.5.1 Print from beginning

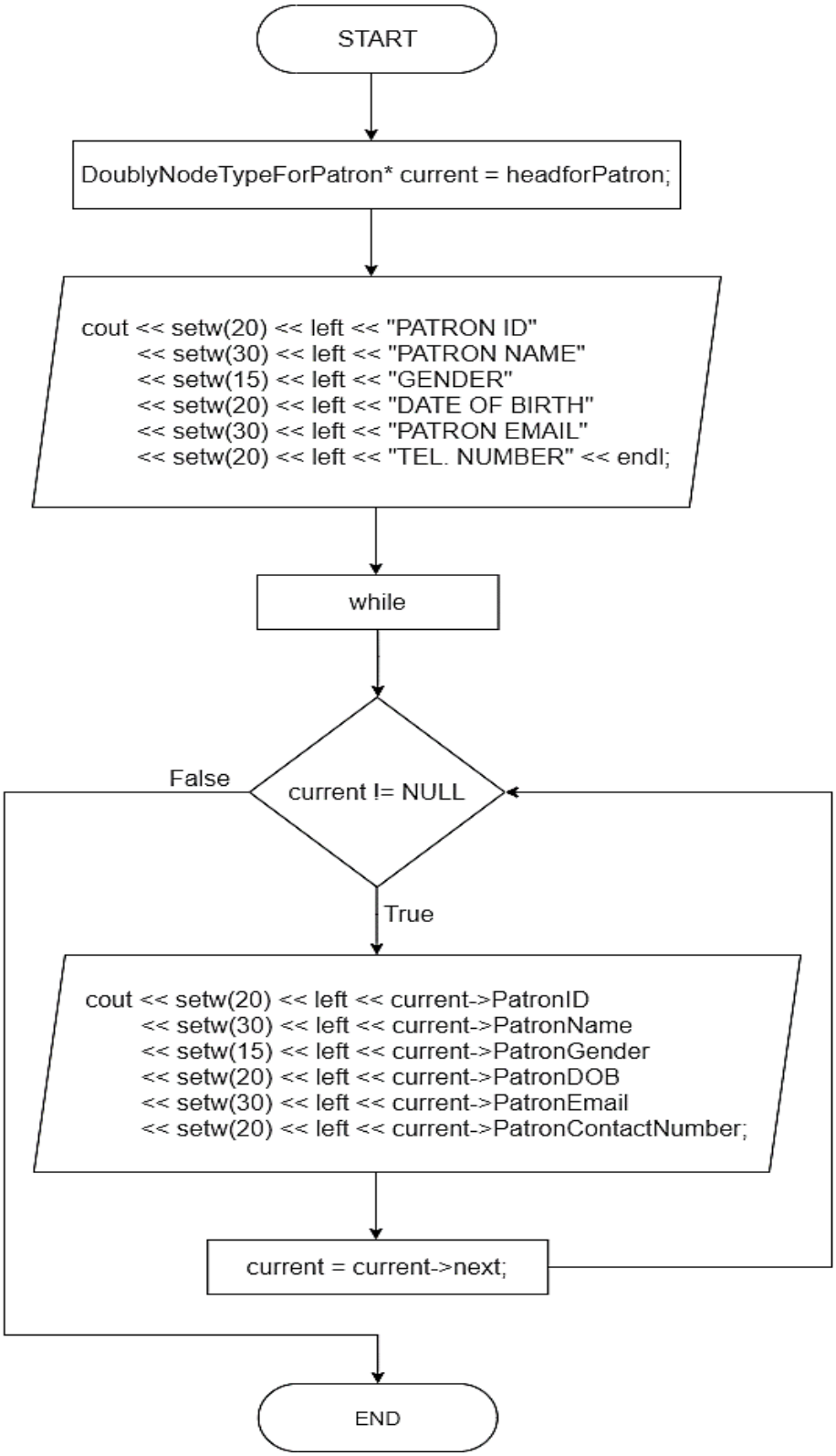


Figure 71: Patrons - Print patron list from beginning flowchart

Code Snippet

```
398 void PatronList::PrintfromBeginning() {
399     //Copy value of headforPatron into current
400     DoublyNodeTypeForPatron* current = headforPatron;
401     cout << endl << endl;
402     cout << setw(20) << left << "PATRON ID" << setw(30) << left << "PATRON NAME" << setw(15) << left << "GENDER" << setw(20) <<
            left << "DATE OF BIRTH" << setw(30) << left << "PATRON EMAIL" << setw(20) << left << "TEL. NUMBER" << endl;
403     //Iterate if the current is not NULL
404     while (current != NULL) {
405         //Print all the records of patron list
406         cout << setw(20) << left << current->PatronID << setw(30) << left << current->PatronName << setw(15) << left <<
                current->PatronGender << setw(20) << left << current->PatronDOB << setw(30) << left << current->PatronEmail << setw(20)
                << left << current->PatronContactNumber;
407         //Move to the next patron
408         current = current->next;
409         cout << endl;
410     }
411     cout << endl;
412 };
```

Figure 72: PrintfromBeginning method - patronList

The Figure 72 shown above is the PrintfromBeginning method. Line 400 is the variable declarations. Line 406 to 409 will be executed only if the patron is not pointing to NULL in Line 404. Line 408 is to move to the next patron.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 3
1 – Print from beginning
2 – Print from behind
Your option is: 1

PATRON ID       PATRON NAME             GENDER      DATE OF BIRTH   PATRON EMAIL                TEL. NUMBER
HKJ001          HEN KIAN JUN            MALE        29/09/1997      KJHEN0929@GMAIL.COM         60139338783
LWY002          LEE WAI YEN             FEMALE      13/10/1998      LWYEN1013@GMAIL.COM         6014848592
LBK003          LOW BOON KIAT           MALE        11/22/1999      BOONKIAT1122@GMAIL.COM      6012395858
NZJ004          NG ZHENG JUE            MALE        20/06/1999      ZHENGJUE1999@GMAIL.COM      6019483838
LYK005          LIM YI KANG             MALE        23/03/1999      YIKANG123@GMAIL.COM         6012394859
YMW006          YONG MUN WEI            MALE        17/10/1997      YONGMUNWEI@GMAIL.COM        6019273828
TKM007          TEOH KHEN MENG          MALE        17/10/1997      KHENMENG1997@GMAIL.COM      6019212328
TYJ008          LIM AH BENG             MALE        24/10/1998      JIFFRED1998@GMAIL.COM       6019272322
JD001           John Doe                MALE        01/01/2020      Johndoe@gmail.com           60165001990
```

Figure 73: Sample output 1 - PrintfromBeginning method - patronList

The Figure 73 shown above is the sample output for PrintfromBeginning method. After entering the print patron section, the user is required to enter a number from either 1 or 2 to print the patron. If the user wants to print the patron from beginning, then the user can enter 1 as the input to view the result.

```
PATRON MENU:
1 - Add a new patron
2 - Search
3 - Print
4 - Update a patron information
5 - Get patron with active book borrow
6 - Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 3
1 - Print from beginning
2 - Print from behind
Your option is: 3
Please enter 1-2 ONLY.
1 - Print from beginning
2 - Print from behind
If you wish to quit Print Section, you may enter 0 *
Your option is: 1

PATRON ID          PATRON NAME                GENDER         DATE OF BIRTH      PATRON EMAIL                TEL. NUMBER
HKJ001             HEN KIAN JUN               MALE           29/09/1997         KJHEN0929@GMAIL.COM         60139338783
LWY002             LEE WAI YEN                FEMALE         13/10/1998         LWYEN1013@GMAIL.COM         6014848592
LBK003             LOW BOON KIAT              MALE           11/22/1999         BOONKIAT1122@GMAIL.COM      6012395858
NZJ004             NG ZHENG JUE               MALE           20/06/1999         ZHENGJUE1999@GMAIL.COM      6019483838
LYK005             LIM YI KANG                MALE           23/03/1999         YIKANG123@GMAIL.COM         6012394859
YMW006             YONG MUN WEI               MALE           17/10/1997         YONGMUNWEI@GMAIL.COM        6019273828
TKM007             TEOH KHEN MENG             MALE           17/10/1997         KHENMENG1997@GMAIL.COM      6019212328
TYJ008             LIM AH BENG                MALE           24/10/1998         JIFFRED1998@GMAIL.COM       6019272322
```

Figure 74: Sample output 2 - PrintfromBeginning method - patronList

The Figure 74 shown above is another sample output for PrintfromBeginning method. After entering the print patron section, the user is required to enter a number from either 1 or 2 to print the patron. If the user entered a number which neither 1 nor 2, then an error message will be shown below and required the user to enter the correct number.

3.2.5.2 Print from behind

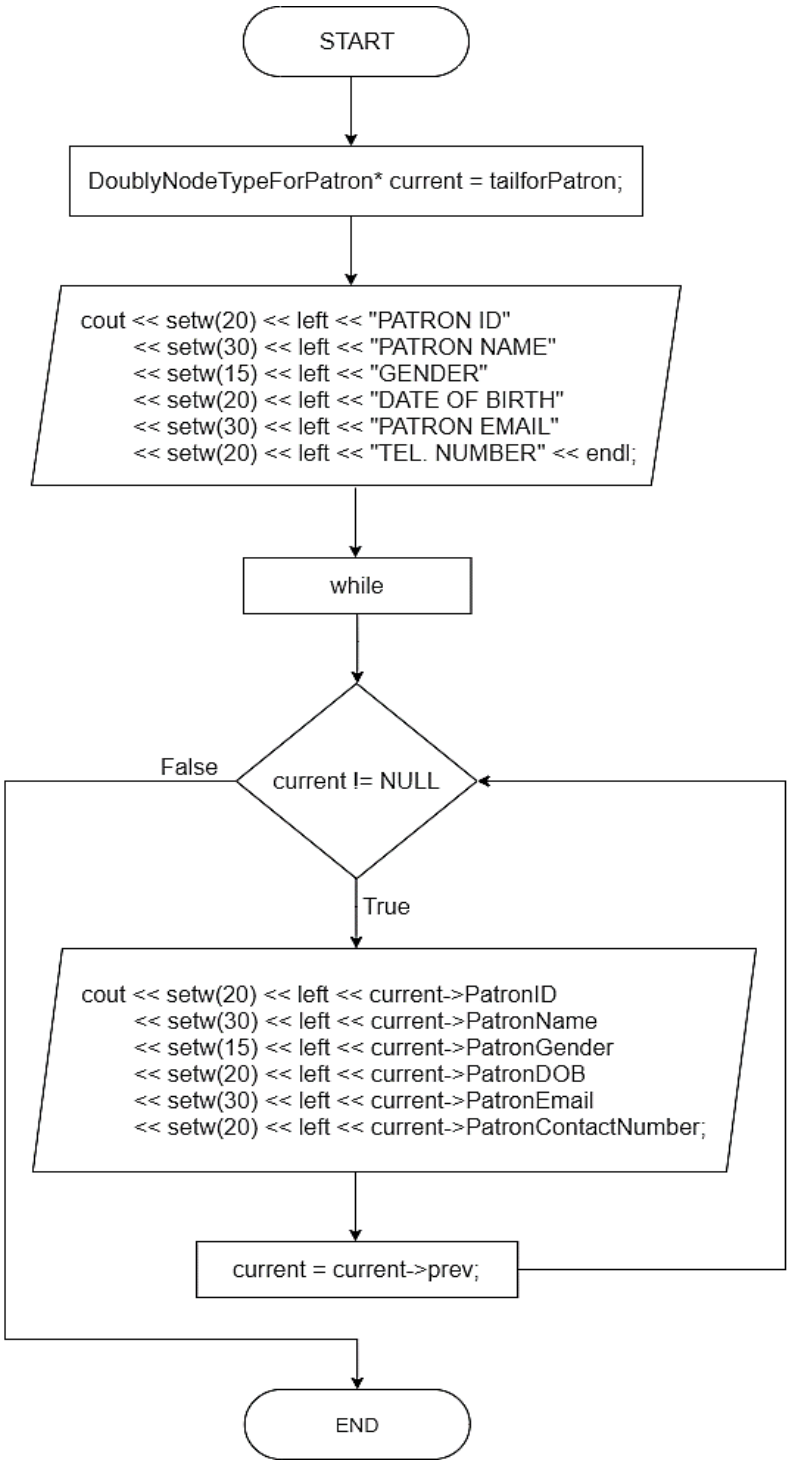


Figure 75: Patrons - Print patron list from behind flowchart

Code Snippet

```
414 void PatronList::PrintfromBehind() {
415     //Copy value of headforPatron into current
416     DoublyNodeTypeForPatron* current = tailforPatron;
417     cout << endl << endl;
418     cout << setw(20) << left << "PATRON ID" << setw(30) << left << "PATRON NAME" << setw(15) << left << "GENDER" << setw(20) <<
            left << "DATE OF BIRTH" << setw(30) << left << "PATRON EMAIL" << setw(20) << left << "TEL. NUMBER" << endl;
419     //Iterate if the current is not NULL
420     while (current != NULL) {
421         //Print all the records
422         cout << setw(20) << left << current->PatronID << setw(30) << left << current->PatronName << setw(15) << left <<
                current->PatronGender << setw(20) << left << current->PatronDOB << setw(30) << left << current->PatronEmail << setw(20)
                << left << current->PatronContactNumber;
423         //Move to the previous patron
424         current = current->prev;
425         cout << endl;
426     }
427     cout << endl;
428 };
```

Figure 76: PrintfromBehind method - patronList

The Figure 76 shown above is the PrintfromBehind method. Line 416 is the variable declarations. Line 422 to 425 will be executed only if the patron is not pointing to NULL in Line 420. Line 424 is to move to the next patron.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 3
1 – Print from beginning
2 – Print from behind
Your option is: 2

PATRON ID       PATRON NAME             GENDER      DATE OF BIRTH   PATRON EMAIL                TEL. NUMBER
JD001           John Doe                MALE        01/01/2020      Johndoe@gmail.com           60165001990
TYJ008          LIM AH BENG             MALE        24/10/1998      JIFFRED1998@GMAIL.COM       6019272322
TKM007          TEOH KHEN MENG          MALE        17/10/1997      KHENMENG1997@GMAIL.COM      6019212328
YMW006          YONG MUN WEI            MALE        17/10/1997      YONGMUNWEI@GMAIL.COM        6019273828
LYK005          LIM YI KANG             MALE        23/03/1999      YIKANG123@GMAIL.COM         6012394859
NZJ004          NG ZHENG JUE            MALE        20/06/1999      ZHENGJUE1999@GMAIL.COM      6019483838
LBK003          LOW BOON KIAT           MALE        11/22/1999      BOONKIAT1122@GMAIL.COM      6012395858
LWY002          LEE WAI YEN             FEMALE      13/10/1998      LWYEN1013@GMAIL.COM         6014848592
HKJ001          HEN KIAN JUN            MALE        29/09/1997      KJHEN0929@GMAIL.COM         60139338783
```

Figure 77: Sample output 1 - PrintfromBehind method - patronList

The Figure 77 shown above is the sample output for PrintfromBehind method. After entering the print patron section, the user is required to enter a number from either 1 or 2 to print the patron. If the user wants to print the patron from behind, then the user can enter 2 as the input to view the result.

3.2.6 Update patrons' information

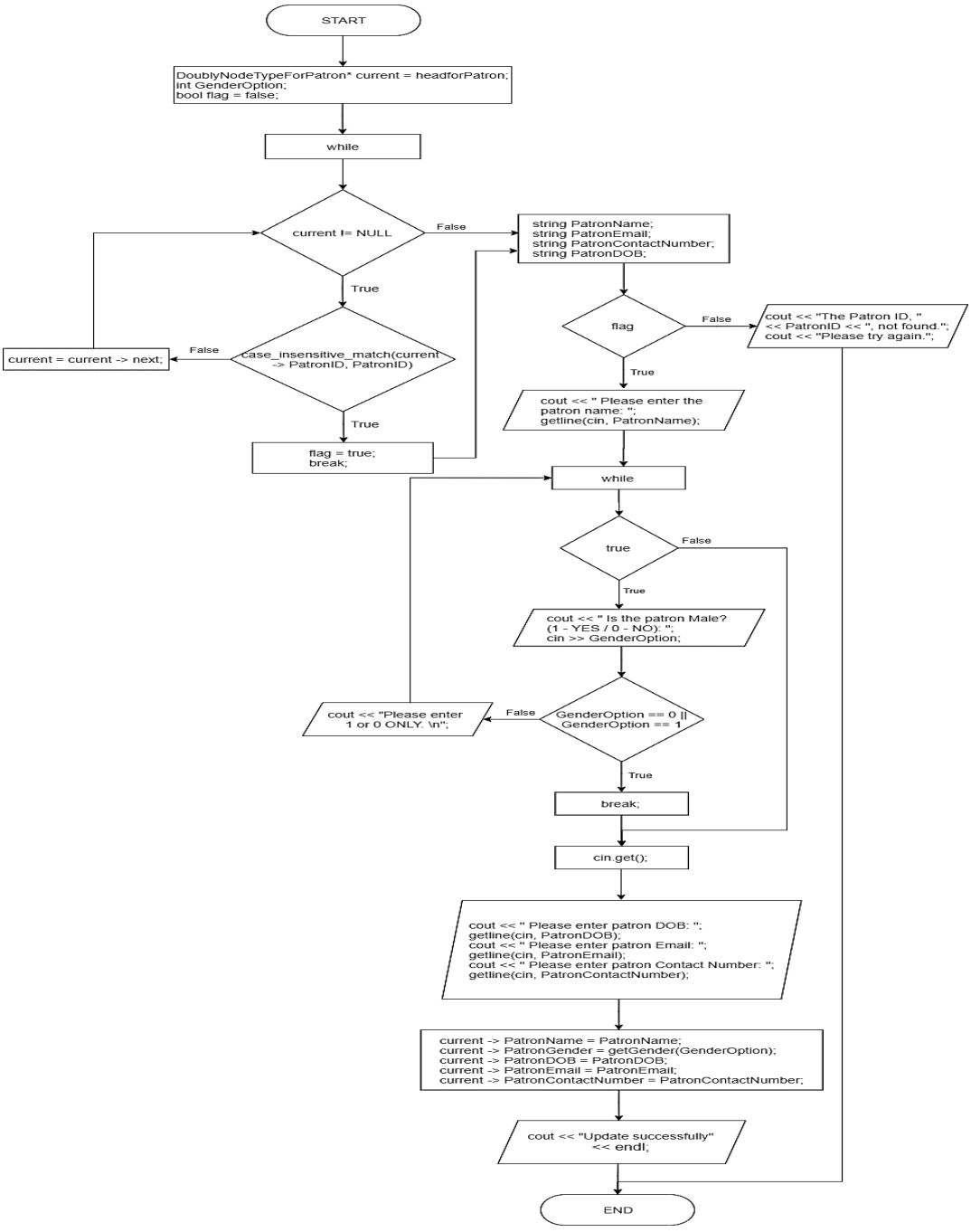


Figure 78: Patrons - Update patron's information flowchart

case_insensitive_match(string, string) is a method in which will be passing this two values to do comparison without the case sensitive.

LINK:

https://drive.google.com/file/d/1lUwthHc8mqYTMhQMl8KSd_74RDlVowkU/view?usp=sharing

Code Snippet

```
430 void PatronList::UpdatePatronInformation(string PatronID) {
431     //Copy value of headforPatron into current
432     DoublyNodeTypeForPatron* current = headforPatron;
433     int GenderOption;
434     bool flag = false;
435     //Iterate if the current is not NULL
436     while (current != NULL) {
437         //Check if the given patron ID is matched with the patron ID in patron list
438         if (case_insensitive_match(current->PatronID, PatronID)) {
439             //Set a true value to flag and break the iteration, if found
440             flag = true;
441             break;
442         }
443         //Move to the next patron
444         current = current->next;
445     }
446 
447     string PatronName;
448     string PatronEmail;
449     string PatronContactNumber;
450     string PatronDOB;
451 
452     //If patron ID is found
453     if (flag) {
454         //Get patron details
455         cout << "Please enter the patron name: ";
456         getline(cin, PatronName);
457 
458         //Iterate if the given input is not either 1 or 0
459         while(true) {
460             cout << "Is the patron Male? (1 - YES / 0 - NO): ";
461             cin >> GenderOption;
462             if(GenderOption == 0 || GenderOption == 1) {
463                 //Break the iteration if the given input is 1 or 0
464                 break;
465             } else {
466                 cout << "Please enter 1 or 0 ONLY. \n";
467             }
468         }
469         cin.get();
470         //Get the patron details
471         cout << "Please enter patron DOB: ";
472         getline(cin, PatronDOB);
473         cout << "Please enter patron Email: ";
474         getline(cin, PatronEmail);
475         cout << "Please enter patron Contact Number: ";
476         getline(cin, PatronContactNumber);
477         //Instatiate the updated patron details to the current node type
478         current->PatronName = PatronName;
479         current->PatronGender = getGender(GenderOption);
480         current->PatronDOB = PatronDOB;
481         current->PatronEmail = PatronEmail;
482         current->PatronContactNumber = PatronContactNumber;
483         cout << "Update successfully";
484         cout << endl;
485     }
486     else {
487         cout << "The Patron ID, " << PatronID << ", not found.";
488         cout << "Please try again.";
489     }
490 }
```

Figure 79: UpdatePatronInformation method - patronList

The Figure 79 shown above is the UpdatePatronInformation method. Line 432 to 434 are the variable declarations. Line 436 to 445 will be iterated the whole patron list to check if the given patron ID is the same as the patron ID in patron list. Line 438 is responsible to check the given patron ID and the patron ID in patron list by converting into lower cases, set the flag to be true and break the iteration if the patron ID is matched. Line 444 is to move to the next patron. Line 447 to 450 are the variable declarations. Line 455 to 484 will be executed only if the flag value is true, which means the patron ID is found in the Line 436 to 445. Line 459 to 468 will be iterated until the user enters either 1 or 0 for gender option, the iteration break happens in Line 462 and 463. Line 471 to 482 is to replace the data with the updated patron information. If the patron ID not found, Line 487 will be executed.

Result

```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 4


Patron Name                  Patron ID
HEN KIAN JUN                 HKJ001
LEE WAI YEN                  LWY002
LOW BOON KIAT                LBK003
NG ZHENG JUE                 NZJ004
LIM YI KANG                  LYK005
YONG MUN WEI                 YMW006
TEOH KHEN MENG               TKM007
LIM AH BENG                  TYJ008
John Doe                     JD001

Please enter your Patron ID: jd001
Please enter the patron name: JAMES DOE
Is the patron Male? (1 – YES / 0 – NO): 0
Please enter patron DOB: 29/09/1999
Please enter patron Email: JamesDoe@gmail.com
Please enter patron Contact Number: 601788229021
Update successfully
```

Figure 80: Sample output 1 - UpdatePatronInformation method - patronList

The Figure 80 shown above is the sample output for UpdatePatronInformation method. The user is required to enter 4 to do an update for the patron. After entering 4, the user is required to provide the patron ID to make update, the patron ID should be within the patron list as shown in the picture. If the patron ID is found within the patron list, then the user can modify the patron's information.

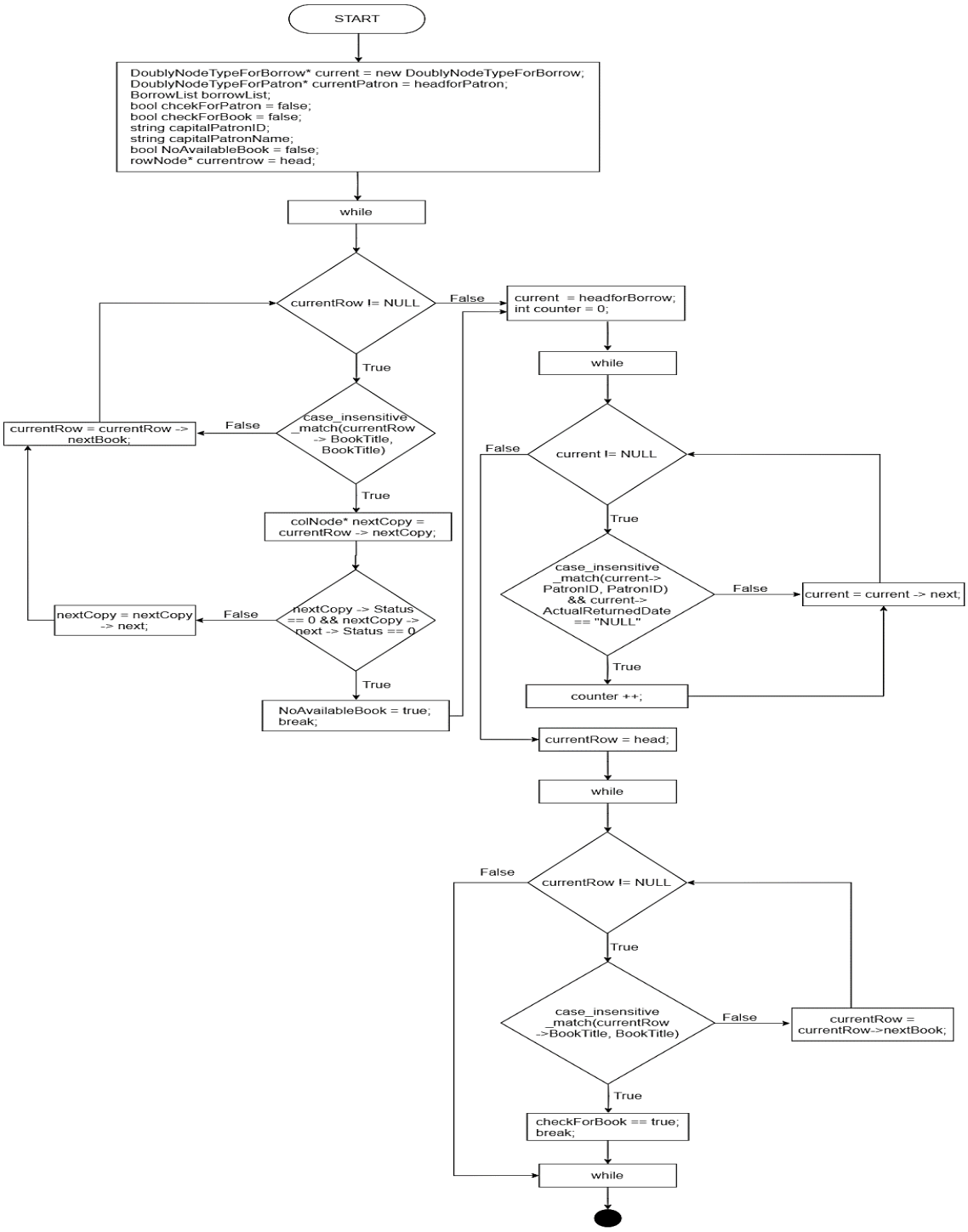
```
PATRON MENU:
1 – Add a new patron
2 – Search
3 – Print
4 – Update a patron information
5 – Get patron with active book borrow
6 – Print the last 10 books borrowed by patron
If you wish to quit Patron Section, you may enter 0 *
Your option is: 4


Patron Name                  Patron ID
HEN KIAN JUN                 HKJ001
LEE WAI YEN                  LWY002
LOW BOON KIAT                LBK003
NG ZHENG JUE                 NZJ004
LIM YI KANG                  LYK005
YONG MUN WEI                 YMW006
TEOH KHEN MENG               TKM007
LIM AH BENG                  TYJ008

Please enter your Patron ID: HKJ002
The Patron ID, HKJ002, not found.Please try again.
```

Figure 81: Sample output 2 - UpdatePatronInformation method - patronList

The Figure 81 shown above is another sample output for UpdatePatronInformation method. The user is required to enter 4 to do an update for the patron. After entering 4, the user is required to provide the patron ID to make update, the patron ID should be within the patron list as shown in the picture. If the patron ID is not found within the patron list, then the error message will be shown directly to the user saying that the given patron ID is not found within the patron list.

3.3 Borrow list

3.3.1 Add borrow books transaction



LINK:

https://drive.google.com/file/d/1qDsBaQIQ_PZZxZflrIwREgOUCmMLRZ9h/view?usp=sharing

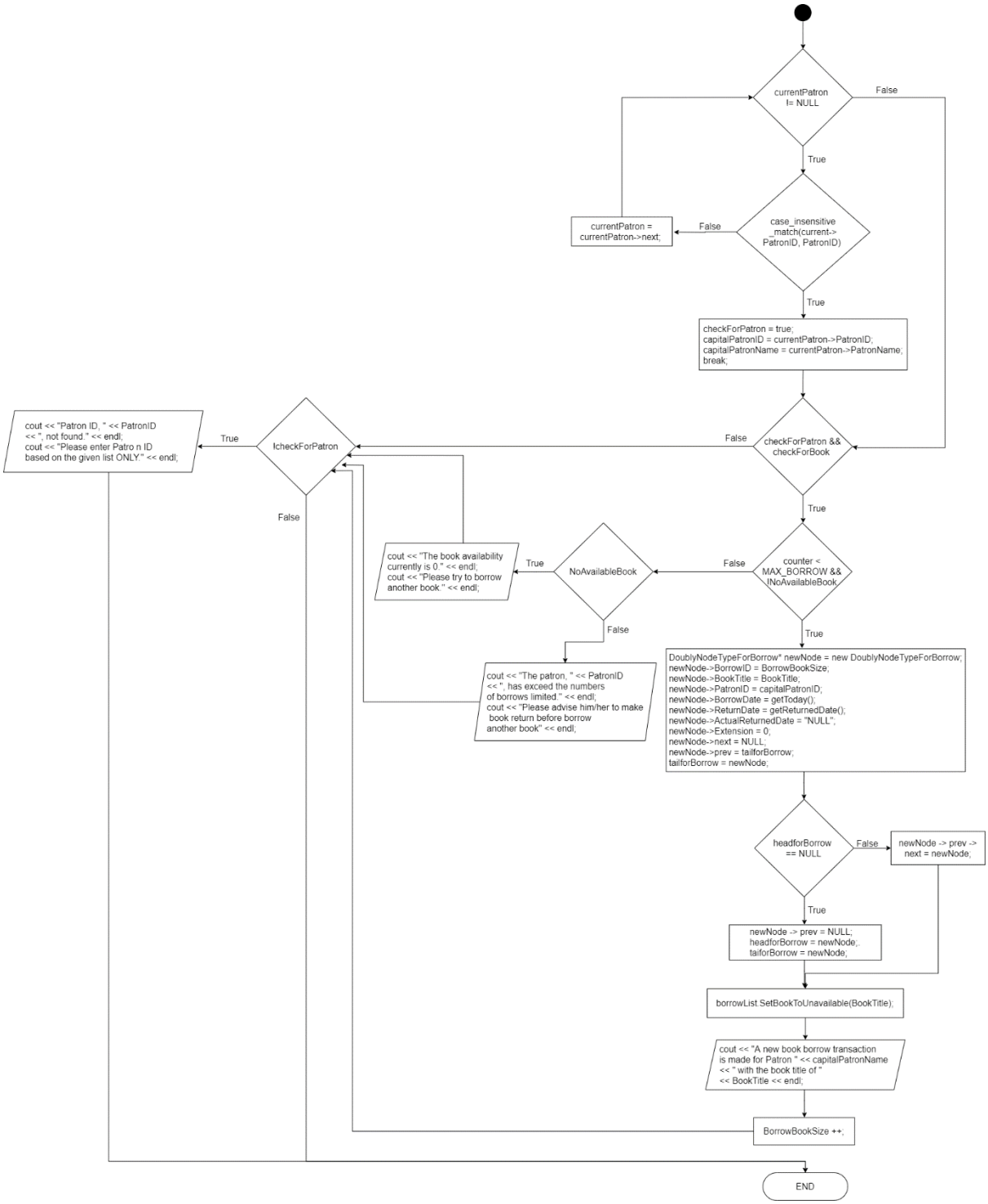


Figure 82: Borrow Books - Add borrow books transaction flowchart

LINK:

https://drive.google.com/file/d/1nRZIyG_iqHj61Hf9bPJwSyTT8scZRpzu/view?usp=sharing

Code Snippet

```
void BorrowList::AddBorrow(string BookTitle, string PatronID) {
    //Create a current as a pointer
    DoublyNodeTypeForBorrow* current = new DoublyNodeTypeForBorrow;
    //Copy value of headforPatron into currentPatron
    DoublyNodeTypeForPatron* currentPatron = headforPatron;
    BorrowList borrowList;

    bool checkForPatron = false;
    bool checkForBook = false;

    string capitalPatronID;
    string capitalPatronName;

    bool NoAvailableBook = false;
    //Copy value of head into currentRow
    rowNode* currentRow = head;
    //Iterate if the currentRow is not NULL
    while(currentRow != NULL) {
        //Check if the given BookTitle is matched with the BookTitle of Book list
        if(case_insensitive_match(currentRow -> BookTitle, BookTitle)) {
            //Copy value of nextCopy into nextCopy pointer
            colNode* nextCopy = currentRow -> nextCopy;
            //Check if both books are 0
            if(nextCopy -> Status == 0 && nextCopy -> next -> Status == 0) {
                //Set a true value to NoAvailableBook and break the iteration
                NoAvailableBook = true;
                break;
            }
            //Move to the next copy
            nextCopy = nextCopy -> next;
        }
        //Move to the next book
        currentRow = currentRow -> nextBook;
    }

    //Copy value of headforBorrow into current
    current = headforBorrow;
    int counter = 0;
    //Iterate if the current is not NULL
    while (current != NULL) {
        //Check if the given patron ID is matched with the patron ID of borrow list and actual returned date is NULL
        if (case_insensitive_match(current->PatronID, PatronID) && current->ActualReturnedDate == "NULL") {
            //Increment 1 to the counter
            counter++;
        }
        current = current->next;
    }

    //Copy value of head into currentRow
    currentRow = head;
    //Iterate if the currentRow is not NULL
    while(currentRow != NULL) {
        //Check if the given BookTitle is matched with the BookTitle of book list
        if(case_insensitive_match(currentRow -> BookTitle, BookTitle)) {
            //Set a true value to checkForBook and break the iteration
            checkForBook = true;
            break;
        }
        //Move to the next book
        currentRow = currentRow -> nextBook;
    }

    //Iterate if the currentPatron is not NULL
    while(currentPatron != NULL) {
        //Check if the given patron ID is matched with the patron ID of patron list
        if(case_insensitive_match(currentPatron -> PatronID, PatronID)) {
            //Set the true value to the checkForPatron and break the iteration
            checkForPatron = true;
            capitalPatronID = currentPatron -> PatronID;
            capitalPatronName = currentPatron -> PatronName;
            break;
        }
        //Move to the next patron
        currentPatron = currentPatron -> next;
    }

    //If both checkForPatron and checkForBook are true value
    if(checkForPatron && checkForBook) {
        //Check if the counter is below 3 and the book is available
        if (counter < MAX_BORROW && !NoAvailableBook) {
            //Create a new node
            DoublyNodeTypeForBorrow* newNode = new DoublyNodeTypeForBorrow;
            newNode->BorrowID = BorrowBookSize;
            newNode->BookTitle = BookTitle;
            newNode->PatronID = capitalPatronID;
            newNode->BorrowDate = getToday();
            newNode->ReturnDate = getReturnedDate();
            newNode->ActualReturnedDate = "NULL";
            newNode->Extension = 0;
            newNode->next = NULL;
            newNode->prev = tailforBorrow;
            tailforBorrow = newNode;
            //Check if the headforBorrow is not pointing to any
            if (headforBorrow == NULL) {
```

```
605                 newNode->prev = NULL;
606                 headforBorrow = newNode;
607                 tailforBorrow = newNode;
608             }
609             else {
610                 newNode->prev->next = newNode;
611             }
612 
613             //Set the copy of the selected title to be unavailable
614             borrowList.SetBookToUnavailable(BookTitle);
615             cout << "A new book borrow transaction is made for Patron " << capitalPatronName << " with the book title of " <<
                    BookTitle <<endl;
616             //Incremental 1 to size of borrow book
617             BorrowBookSize++;
618         } else if (NoAvailableBook) { //NoAvailableBook is true only if both copies of book is 0
619             cout << "The book availability currently is 0." << endl;
620             cout << "Please try to borrow another book." << endl;
621         }
622         else { //If the book is available but the patron has reached its maximum
623             cout << "The patron, " << PatronID << ", has exceed the numbers of borrows limited." << endl;
624             cout << "Please advise him/her to make book return before borrow another book" << endl;
625         }
626     }
627      if (!checkForPatron) { //If the given patron ID is not found
628          cout << "Patron ID, " << PatronID << ", not found." << endl;
629          cout << "Please enter Patron ID based on the given list ONLY." << endl;
630      }
631 };
```

Figure 83: AddBorrow method - borrowList

The Figure 83 shown above is the AddBorrow method. Line 513 to 526 are the variable declaration. Line 528 to 544 is to check if the given book title is existed in book linked list. Line 530 to 543 will be executed only if the book pointer is not pointing to NULL. Line 530 is to check if the given book title is matched with the book title in book linked list by converting into lower case, if it does, then Line 532 to 540 will be executed. Line 532 is another pointer declaration. Line 534 is to check if the statuses of both copies are equal to 0, if it does, then Line 536 and 537 will be executed. Line 536 is to set a true value to variable of NoAvailableBook and Line 537 is to break the iteration. Line 540 is to move to the next copy, and Line 543 is to move to the next book. Line 547 to 548 are the variable declaration. Line 550 is to check if the given patron ID is existed in the borrow linked list and its actual returned date is equal to NULL. Line 552 to 556 will be executed only if the pointer is not pointing to NULL as shown in Line 550. Line 552 is to check if the given patron ID is matched with the patron ID in the borrow linked list and its actual returned date is equal to NULL, if it does, then add 1 as increment to counter. Line 556 is to move to the next borrow record. Line 564 to 570 will be executed only if the book pointer is not pointing to NULL as shown in 562. Line 564 is to check if the given book title is matched with the book title in book linked list by converting into lower case, if it does, then Line 566 to 567 will be executed. Line 566 is to set a true value to variable of checkForBook, and Line 567 is to break the iteration. Line 570 is to move to the next book. Line 574 to 585 is to check if the given patron ID is existed in the patron linked list and assign the value to the variables of capitalPatronID and capitalPatronName. Line 576 to 584 will be executed only if the patron pointer is not pointing to NULL as shown in Line 574.

Line 576 is to check if the given patron ID is matched with the patron ID in patron linked list by converting to lower cases, if it does, then Line 578 to 581 will be executed. Line 578 is to set a true value to checkForPatron, and Line 581 is to break the iteration. Line 584 is to move to the next patron.

Line 590 to 625 will be executed only if the checkForPatron and checkForBook are true values, where it is set in Line 566 and 578. Line 590 is to check if the book is available and the counter is lesser than MAX_BORROW, MAX_BORROW is a constant variable, which contains value of 3, then Line 592 to 602 will be executed. Line 592 is the pointer declaration, Line 596 is to call getToday method to obtain the today date based on system and assign to BorrowDate data member, and Line 597 is to call getReturnedDate to obtain the returned date which is 15 days after from today and assign to ReturnDate data member. Line 604 to 611 is to add into borrow linked list. Line 604 is to check if the book linked list is pointing NULL, if it does, then Line 605 to 607 will be executed, else Line 610 will be executed. Line 614 will be executed to call SetBookToUnavailable method to make the book copy to be unavailable. Line 617 is to add 1 as increment to size of book borrow. Line 619 to 620 will be executed if the NoAvailableBook is true value, where it is detected in Line 528 to 544, it is indicating the both copies of the book are unavailable. Line 623 to 624 will be executed only if the counter is greater than MAX_BORROW, which indicates the patron has borrowed up to a maximum of 3 books. Line 627 to 630 will be executed if the patron ID is not found in Line 574 to 585.

Result

```
BORROW MENU:
1 – Add a new book borrow transaction
2 – Print
3 – Make Extension
4 – Return a book
Your option is: 1


Patron Name                  Patron ID
HEN KIAN JUN                 HKJ001
LEE WAI YEN                  LWY002
LOW BOON KIAT                LBK003
NG ZHENG JUE                 NZJ004
LIM YI KANG                  LYK005
YONG MUN WEI                 YMW006
TEOH KHEN MENG               TKM007
LIM AH BENG                  TYJ008
JAMES DOE                    JD001

Please enter patron ID: lwy002


BOOK TITLE                        BOOK AUTHOR                      ISBN
HARRY POTTER                      J. K. ROWLING                    6808
MEIN KAMPF                        ADOLF HITLER                     5250
13 REASONS WHY                    JAY ASHER                        74
THE INVISIBLE MAN                 H. G. WELLS                      3659
THE HUNGER GAMES                  SUZANNE COLLINS                  8931
I HAVE A DREAM                    MARTIN LUTHER KING JR.           1273
CRIME AND PERIODICALS             NOVA EVERLY                      7545
A NEW EARTH                       ECKHART TOLLE                    879
THE HERO WITH A THOUSAND FACES    JOSEPH CAMPBELL                  7924

Please enter Book ISBN ONLY: 7545
A new book borrow transaction is made for Patron LEE WAI YEN with the book title of CRIME AND PERIODICALS
```

Figure 84: Sample output 1 - AddBorrow method - borrowList

The Figure 84 shown above is the sample output for AddBorrow method. The user will need to enter 1 to add a new borrow transaction. Then, user is required to base on the patron list shown in the picture to type the patron ID and based on the book list to type the book ISBN. If the patron ID typed is within the patron list and the book ISBN is typed within the book list, then the successful message will be prompted to the user saying that a new borrow transaction is added into the system.

```
BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 1


Patron Name                  Patron ID
HEN KIAN JUN                 HKJ001
LEE WAI YEN                  LWY002
LOW BOON KIAT                LBK003
NG ZHENG JUE                 NZJ004
LIM YI KANG                  LYK005
YONG MUN WEI                 YMW006
TEOH KHEN MENG               TKM007
LIM AH BENG                  TYJ008

Please enter patron ID: HKJ001


BOOK TITLE                        BOOK AUTHOR                     ISBN
HARRY POTTER                      J. K. ROWLING                   6808
MEIN KAMPF                        ADOLF HITLER                    5250
13 REASONS WHY                    JAY ASHER                       74
THE INVISIBLE MAN                 H. G. WELLS                     3659
THE HUNGER GAMES                  SUZANNE COLLINS                 8931
I HAVE A DREAM                    MARTIN LUTHER KING JR.          1273
CRIME AND PERIODICALS             NOVA EVERLY                     7545
A NEW EARTH                       ECKHART TOLLE                   879
THE HERO WITH A THOUSAND FACES    JOSEPH CAMPBELL                 7924

Please enter Book ISBN ONLY: 74
The book availability currently is 0.
Please try to borrow another book.
```

Figure 85: Sample output 2 - AddBorrow method - borrowList

The Figure 85 shown above is the sample output for AddBorrow method. The user will need to enter 1 to add a new borrow transaction. Then, user is required to base on the patron list shown in the picture to type the patron ID and based on the book list to type the book ISBN. Even though, if the patron ID typed is within the patron list however, if the book availability is not available, the error message will be prompted to the user indicating that the book currently is not available.

```
BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 1


Patron Name                  Patron ID
HEN KIAN JUN                 HKJ001
LEE WAI YEN                  LWY002
LOW BOON KIAT                LBK003
NG ZHENG JUE                 NZJ004
LIM YI KANG                  LYK005
YONG MUN WEI                 YMW006
TEOH KHEN MENG               TKM007
LIM AH BENG                  TYJ008

Please enter patron ID: HKJ001


BOOK TITLE                       BOOK AUTHOR                      ISBN
HARRY POTTER                     J. K. ROWLING                    6808
MEIN KAMPF                       ADOLF HITLER                     5250
13 REASONS WHY                   JAY ASHER                        74
THE INVISIBLE MAN                H. G. WELLS                      3659
THE HUNGER GAMES                 SUZANNE COLLINS                  8931
I HAVE A DREAM                   MARTIN LUTHER KING JR.           1273
CRIME AND PERIODICALS            NOVA EVERLY                      7545
A NEW EARTH                      ECKHART TOLLE                    879
THE HERO WITH A THOUSAND FACES   JOSEPH CAMPBELL                  7924

Please enter Book ISBN ONLY: 3659
The patron, HKJ001, has exceed the numbers of borrows limited.
Please advise him/her to make book return before borrow another book
```

Figure 86: Sample output 3 - AddBorrow method - borrowList

The Figure 86 shown above is the sample output for AddBorrow method. The user will need to enter 1 to add a new borrow transaction. Then, user is required to base on the patron list shown in the picture to type the patron ID and based on the book list to type the book ISBN. Even though, if the patron ID typed is within the patron list and the book ISBN typed is within the book list, however, if the patron ID has exceed the borrow limit, the error message will be prompted to the user indicating that the patron has exceed the borrow limit and suggest the patron to return a book before borrow another book.

```
BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 1


Patron Name                  Patron ID
HEN KIAN JUN                 HKJ001
LEE WAI YEN                  LWY002
LOW BOON KIAT                LBK003
NG ZHENG JUE                 NZJ004
LIM YI KANG                  LYK005
YONG MUN WEI                 YMW006
TEOH KHEN MENG               TKM007
LIM AH BENG                  TYJ008

Please enter patron ID: HKJ002


BOOK TITLE                        BOOK AUTHOR                    ISBN
HARRY POTTER                      J. K. ROWLING                  6808
MEIN KAMPF                        ADOLF HITLER                   5250
13 REASONS WHY                    JAY ASHER                      74
THE INVISIBLE MAN                 H. G. WELLS                    3659
THE HUNGER GAMES                  SUZANNE COLLINS                8931
I HAVE A DREAM                    MARTIN LUTHER KING JR.         1273
CRIME AND PERIODICALS             NOVA EVERLY                    7545
A NEW EARTH                       ECKHART TOLLE                  879
THE HERO WITH A THOUSAND FACES    JOSEPH CAMPBELL                7924

Please enter Book ISBN ONLY: 6808
Patron ID, HKJ002, not found.
Please enter Patron ID based on the given list ONLY.
```

Figure 87: Sample output 4 - AddBorrow method - borrowList

The Figure 87 shown above is the sample output for AddBorrow method. The user will need to enter 1 to add a new borrow transaction. Then, user is required to base on the patron list shown in the picture to type the patron ID and based on the book list to type the book ISBN. Even though, if the book ISBN typed is within the book list however, if the patron ID typed is not within the patron list, the error message will be prompted to the user indicating that the patron ID is not found within the patron list.

3.3.2 Make extension date

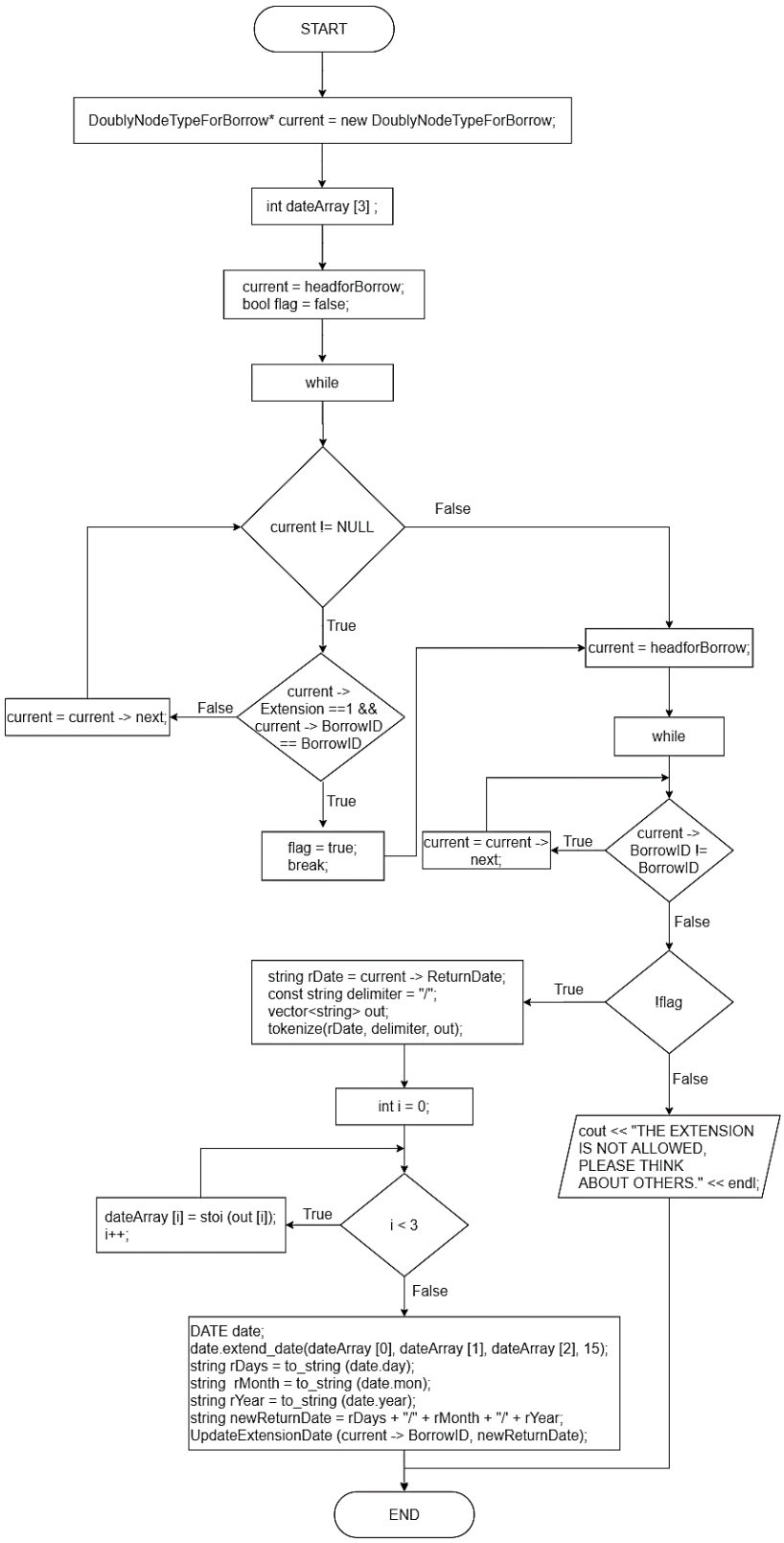


Figure 88: Borrow Books - Make extension date flowchart

Code Snippet

```
753 void BorrowList::MakeExtension(int BorrowID) {
754     //Copy value of headforBorrow into current
755     DoublyNodeTypeForBorrow* current = new DoublyNodeTypeForBorrow;
756     current = headforBorrow;
757 
758     int dateArray[3];
759     bool flag = false;
760     while (current != NULL) { //Iterate if the current is not NULL
761         //If the given borrow ID is matched with the borrow ID of borrow list and its extension is 1
762         if (current->Extension == 1 && current -> BorrowID == BorrowID) {
763             flag = true; //Set a true value to flag
764             break; //Break the iteration
765         }
766         current = current->next; //Move to the next borrow
767     }
768 
769     current = headforBorrow; //Copy value of headforBorrow into current
770     while (current->BorrowID != BorrowID) { //Iterate if the given borrow ID is not matched with the borrow ID of borrow list
771         current = current->next; //Move to the next borrow
772     }
773 
774     if (!flag) { //If the borrow ID does not make any extension before
775         string rDate = current->ReturnDate;
776         const string delimiter = "/";
777         vector<string> out;
778         tokenize(rDate, delimiter, out);
779         for (int i = 0; i < 3; i++) {
780             dateArray[i] = stoi(out[i]);
781         }
782         DATE date;
783         //Pass the day, month, and year as well as number of extension days to extend_date method
784         date.extend_date(dateArray[0], dateArray[1], dateArray[2], 15);
785         string rDays = to_string(date.day); //Convert int into string
786         string rMonth = to_string(date.mon); //Convert int into string
787         string rYear = to_string(date.year); //Convert int into string
788         string newReturnDate = rDays + "/" + rMonth + "/" + rYear; //Concatenate three string with / symbols to a single string
789         UpdateExtensionDate(current->BorrowID, newReturnDate); //Update the return date of the borrow ID to be new returned date
790     }
791     else {
792         cout << "THE EXTENSION IS NOT ALLOWED, PLEASE THINK ABOUT OTHERS.";
793         cout << endl;
794     }
795 }
```

Figure 89: MakeExtension method - borrowList

The Figure 89 shown above is the MakeExtension method. Line 755 to 759 is the variable declaration. Line 760 will iterate if the borrow list is not NULL. Line 762 will check if the given borrow ID is matched with the borrow ID of the borrow list and its extension is 1, if it does, it will set a value true to flag and break the iteration. Line 766 is to move to next borrow. Line 770 will iterate if the given borrow ID is not matched with the borrow ID of borrow list, if it does, it will move to the next borrow. Line 774 is to check if the borrow ID does not make any extension before, then is will calculate and state the date. If borrow ID made extension before, the it will execute Line 784 to 789 which pass the day, month and year and the number of extension days to extend_date method. It will convert the date, month and year into string. Line 789 will call the method to update the return date of the borrow ID to be the new returned date.

Result

```
BORROW MENU:
1 – Add a new book borrow transaction
2 – Print
3 – Make Extension
4 – Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 3


BORROW ID BOOK TITLE                        PATRON ID
1         13 REASONS WHY                    JD001
2         13 REASONS WHY                    HKJ001
3         THE INVISIBLE MAN                 LWY002
4         THE INVISIBLE MAN                 LBK003
5         MEIN KAMPF                        TYJ008
7         HARRY POTTER                      NZJ004
8         HARRY POTTER                      TKM007
9         A NEW EARTH                       LYK005
10        A NEW EARTH                       JD001
12        THE HERO WITH A THOUSAND FACES    HKJ001
13        CRIME AND PERIODICALS             LWY002

Please enter the borrow ID: 2
UPDATE SUCCESSFULLY !

BORROW MENU:
1 – Add a new book borrow transaction
2 – Print
3 – Make Extension
4 – Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 2
1 – Print from beginning
2 – Print from behind
3 – Print last 10 book borrowed by patrons
Your option is: 1


BORROW ID BOOK TITLE                  PATRON ID   BORROW DATE   RETURN DATE   ACTUAL RETURNED DATE   EXTENSION
1         13 REASONS WHY              JD001       30/8/2020     14/9/2020     NULL                   0
2         13 REASONS WHY              HKJ001      30/8/2020     29/9/2020     NULL                   1
3         THE INVISIBLE MAN           LWY002      30/8/2020     14/9/2020     NULL                   0
```

Figure 90: Sample output 1 - MakeExtension method - borrowList

The Figure 90 shown above is the sample output for MakeExtension method. The user is required to enter 3 to make borrow extension. After entering 3, the user is required to provide the borrow ID to make the extension, the borrow ID should be within the borrow list as shown in the picture. If the borrow ID is found within the borrow list, then the borrow extension is then updated successfully.

```
BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 3


BORROW ID BOOK TITLE                          PATRON ID
1         13 REASONS WHY                      HKJ001
2         13 REASONS WHY                      HKJ001

Please enter the borrow ID: 2
UPDATE SUCCESSFULLY !

BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 3


BORROW ID BOOK TITLE                          PATRON ID
1         13 REASONS WHY                      HKJ001
2         13 REASONS WHY                      HKJ001

Please enter the borrow ID: 2
THE EXTENSION IS NOT ALLOWED, PLEASE THINK ABOUT OTHERS.
```

Figure 91: Sample output 2 - MakeExtension method - borrowList

The Figure 91 shown above is another sample output for MakeExtension method. The user is required to enter 3 to make the borrow extension. After entering 3, the user is required to provide the borrow ID to make the extension, the borrow ID should be within the borrow list as shown in the picture. If the borrow ID has exceed the number of borrow extension, then the error message will be show directly to the user saying that extension is not allow for the borrow ID.

3.3.3 Print borrow list

3.3.3.1 Print 10 last borrow list

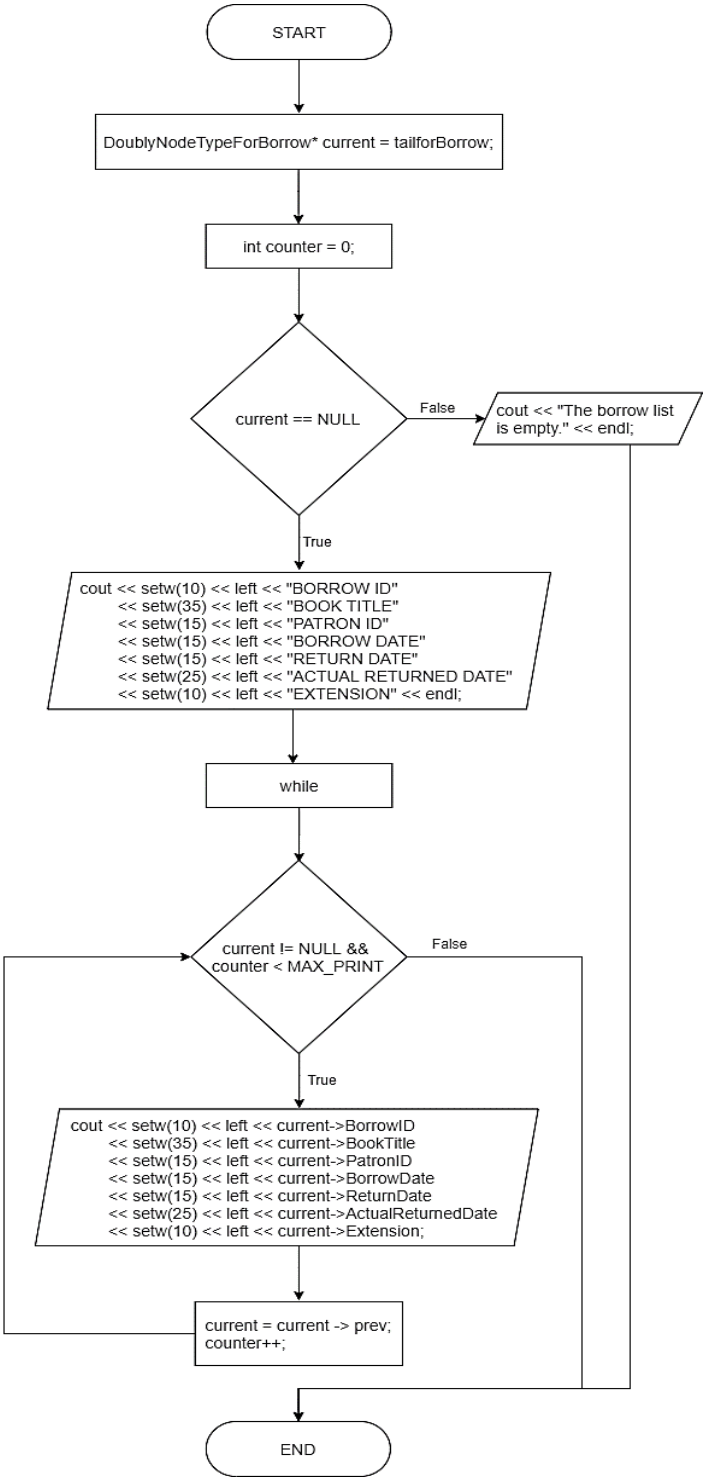


Figure 92: Borrow Books - Display the last 10 books borrowed transaction flowchart

MAX_PRINT is a constant variable with the value of 10, therefore this flowchart will be printing only the maximum of 10 records of book borrowed by a patron.

Code Snippet

```
797 void BorrowList::Print10Last() {
798     //Copy value of tailforBorrow into current
799     DoublyNodeTypeForBorrow* current = tailforBorrow;
800     int counter = 0;
801 
802     cout << endl <<endl;
803     if(current == NULL) { //Check if the current is NULL
804         cout << "The borrow list is empty." << endl;
805     } else {
806         //Print the header of borrow list
807         cout << setw(10) << left << "BORROW ID" << setw(35) << left << "BOOK TITLE" << setw(15) << left << "PATRON ID" << setw(15)
                << left << "BORROW DATE" << setw(15) << left << "RETURN DATE" << setw(25) << left << "ACTUAL RETURNED DATE" << setw(10)
                << left << "EXTENSION" << endl;
808         while (current != NULL && counter < MAX_PRINT) { //Iterate if the current is not NULL and the counter is lesser than 10
809             //Print all records of borrow list
810             cout << setw(10) << left << current->BorrowID << setw(35) << left << current->BookTitle << setw(15) << left <<
                    current->PatronID << setw(15) << left << current->BorrowDate << setw(15) << left << current->ReturnDate << setw(25)
                    << left << current->ActualReturnedDate << setw(10) << left << current->Extension;
811             current = current->prev; //Move to the previous borrow
812             counter++; //Increment 1 to counter
813             cout << endl;
814         }
815     }
816     cout << endl;
817 }
```

Figure 93: Print10Last method - borrowList

The Figure 93 shown above is the Prinit10Last method. Line 799 to 900 are the variable declaration. Line 803 is to check if the tail pointer for borrow is pointing to NULL, if it does, then Line 804 will be executed. Otherwise, the Line 807 to 814 will be executed. Line 804 is to print the header, which is showing what are the columns. Line 810 to 813 will be executed only if the tail pointer is not pointing is not NULL, and the counter variable must be lesser than MAX_PRINT, MAX_PRINT is a constant variable, which contains a value of 10. Line 810 is to print the records based on the pointer. Line 811 is to move to the previous borrow record, Line 812 is to add 1 as increment to counter.

Result

```
BORROW MENU:
1 – Add a new book borrow transaction
2 – Print
3 – Make Extension
4 – Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 2
1 – Print from beginning
2 – Print from behind
3 – Print last 10 book borrowed by patrons
Your option is: 4
INVALID INPUT
```

Figure 94: Sample output 1 - Print10Last method - borrowList

The Figure 94 shown above is another sample output for Print10Last method. After entering the print borrow section, the user is required to enter a number from 1 to 3 to print the patron. However, if the user enters an input which is out of the range between 1 and 3 then an error message will prompt out to indicate the user that the input is invalid.

```
BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 2
1 - Print from beginning
2 - Print from behind
3 - Print last 10 book borrowed by patrons
Your option is: 3

BORROW ID BOOK TITLE                       PATRON ID      BORROW DATE    RETURN DATE    ACTUAL RETURNED DATE     EXTENSION
13        CRIME AND PERIODICALS            LWY002         30/8/2020      14/9/2020      NULL                     0
12        THE HERO WITH A THOUSAND FACES   HKJ001         30/8/2020      14/9/2020      NULL                     0
11        I HAVE A DREAM                   HKJ001         30/8/2020      14/9/2020      30/8/2020                0
10        A NEW EARTH                      JD001          30/8/2020      14/9/2020      NULL                     0
9         A NEW EARTH                      LYK005         30/8/2020      14/9/2020      NULL                     0
8         HARRY POTTER                     TKM007         30/8/2020      14/9/2020      NULL                     0
7         HARRY POTTER                     NZJ004         30/8/2020      14/9/2020      NULL                     0
6         MEIN KAMPF                       HKJ001         30/8/2020      14/9/2020      30/8/2020                0
5         MEIN KAMPF                       TYJ008         30/8/2020      14/9/2020      NULL                     0
4         THE INVISIBLE MAN                LBK003         30/8/2020      14/9/2020      NULL                     0
```

Figure 95: Sample output 2 - Print10Last method - borrowList

The Figure 95 shown above is the sample output for Print10Last method. After entering the print borrow section, the user is required to enter a number from 1 to 3 to print the patron. If the user wants to print the last 10 borrow transaction, then user can enter 3 as input to view the result.

3.3.3.2 Print from Beginning

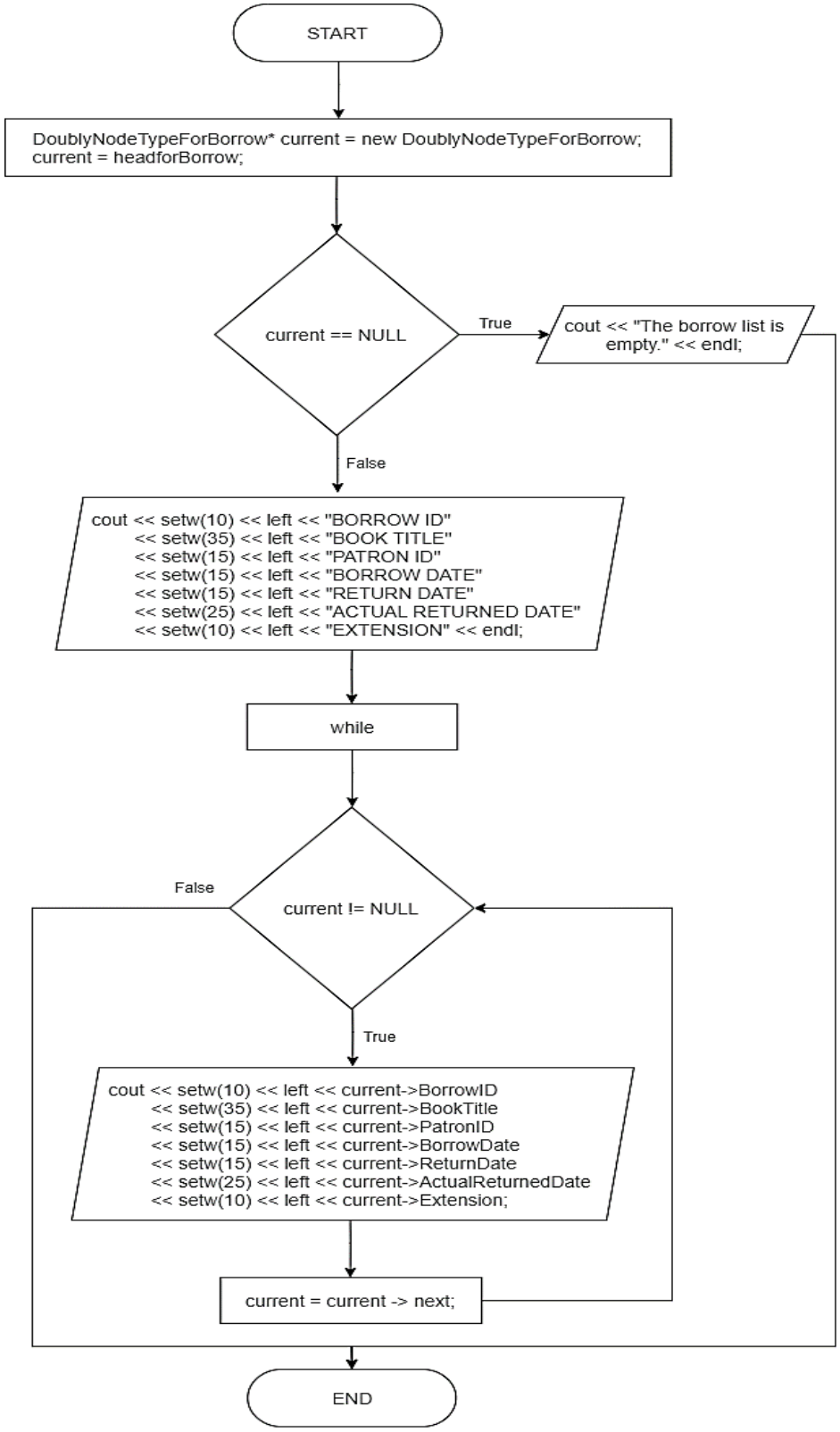


Figure 96: Borrow Books - Display all book borrowed transactions by all patrons (Beginning) flowchart

Code Snippet

```
637 void BorrowList::PrintfromBeginning() {
638     //Copy value of headforBorrow into current
639     DoublyNodeTypeForBorrow* current = new DoublyNodeTypeForBorrow;
640     current = headforBorrow;
641 
642     cout << endl << endl;
643     if(current == NULL) { //Check if the current is not pointing to any
644         cout << "The borrow list is empty." << endl;
645     } else {
646         //Print the header of borrow list
647         cout << setw(10) << left << "BORROW ID" << setw(35) << left << "BOOK TITLE" << setw(15) << left << "PATRON ID" << setw(15)
                << left << "BORROW DATE" << setw(15) << left << "RETURN DATE" << setw(25) << left << "ACTUAL RETURNED DATE" << setw(10)
                << left << "EXTENSION" << endl;
648         while (current != NULL) { //Iterate if the current is not NULL
649             //Print all the records of borrow list
650             cout << setw(10) << left << current->BorrowID << setw(35) << left << current->BookTitle << setw(15) << left <<
                    current->PatronID << setw(15) << left << current->BorrowDate << setw(15) << left << current->ReturnDate << setw(25)
                    << left << current->ActualReturnedDate << setw(10) << left << current->Extension;
651             //Move to the next borrow list
652             current = current->next;
653             cout << endl;
654         }
655     }
656     cout << endl;
657 };
```

Figure 97: PrintfromBeginning method - borrowList

The Figure 97 shown above is the PrintfromBeginning method. Line 639 to 640 are the variable declaration. Line 643 is to check if the borrow is pointing to the NULL, if it does, then Line 644 will be executed. Otherwise, Line 647 to 653 will be executed. Line 647 is to print the headers, which to show what are the columns. Line 650 to 653 will be executed only if the pointing is not pointing to NULL as shown in Line 648. Line 650 is to print the records based on the current pointer. Line 652 is to move to the next borrow record.

Result

```
BORROW MENU:
1 – Add a new book borrow transaction
2 – Print
3 – Make Extension
4 – Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 2
1 – Print from beginning
2 – Print from behind
3 – Print last 10 book borrowed by patrons
Your option is: 1

BORROW ID BOOK TITLE                      PATRON ID      BORROW DATE    RETURN DATE    ACTUAL RETURNED DATE     EXTENSION
1         13 REASONS WHY                  JD001          30/8/2020      14/9/2020      NULL                     0
2         13 REASONS WHY                  HKJ001         30/8/2020      29/9/2020      NULL                     1
3         THE INVISIBLE MAN               LWY002         30/8/2020      14/9/2020      NULL                     0
4         THE INVISIBLE MAN               LBK003         30/8/2020      14/9/2020      NULL                     0
5         MEIN KAMPF                      TYJ008         30/8/2020      14/9/2020      NULL                     0
6         MEIN KAMPF                      HKJ001         30/8/2020      14/9/2020      30/8/2020                0
7         HARRY POTTER                    NZJ004         30/8/2020      14/9/2020      NULL                     0
8         HARRY POTTER                    TKM007         30/8/2020      14/9/2020      NULL                     0
9         A NEW EARTH                     LYK005         30/8/2020      14/9/2020      NULL                     0
10        A NEW EARTH                     JD001          30/8/2020      14/9/2020      NULL                     0
11        I HAVE A DREAM                  HKJ001         30/8/2020      14/9/2020      30/8/2020                0
12        THE HERO WITH A THOUSAND FACES  HKJ001         30/8/2020      14/9/2020      NULL                     0
13        CRIME AND PERIODICALS           LWY002         30/8/2020      14/9/2020      NULL                     0
```

Figure 98: Sample output 1 - PrintfromBeginning method - borrowList

The Figure 98 shown above is the sample output for PrintfromBeginning method. After entering the print borrow transaction section, the user is required to enter a number from 1 to 3 to print the borrow transaction. If the user wants to print the borrow transaction from the beginning, then the user can enter 1 as the input to view the result.

3.3.3.3 Print from Behind

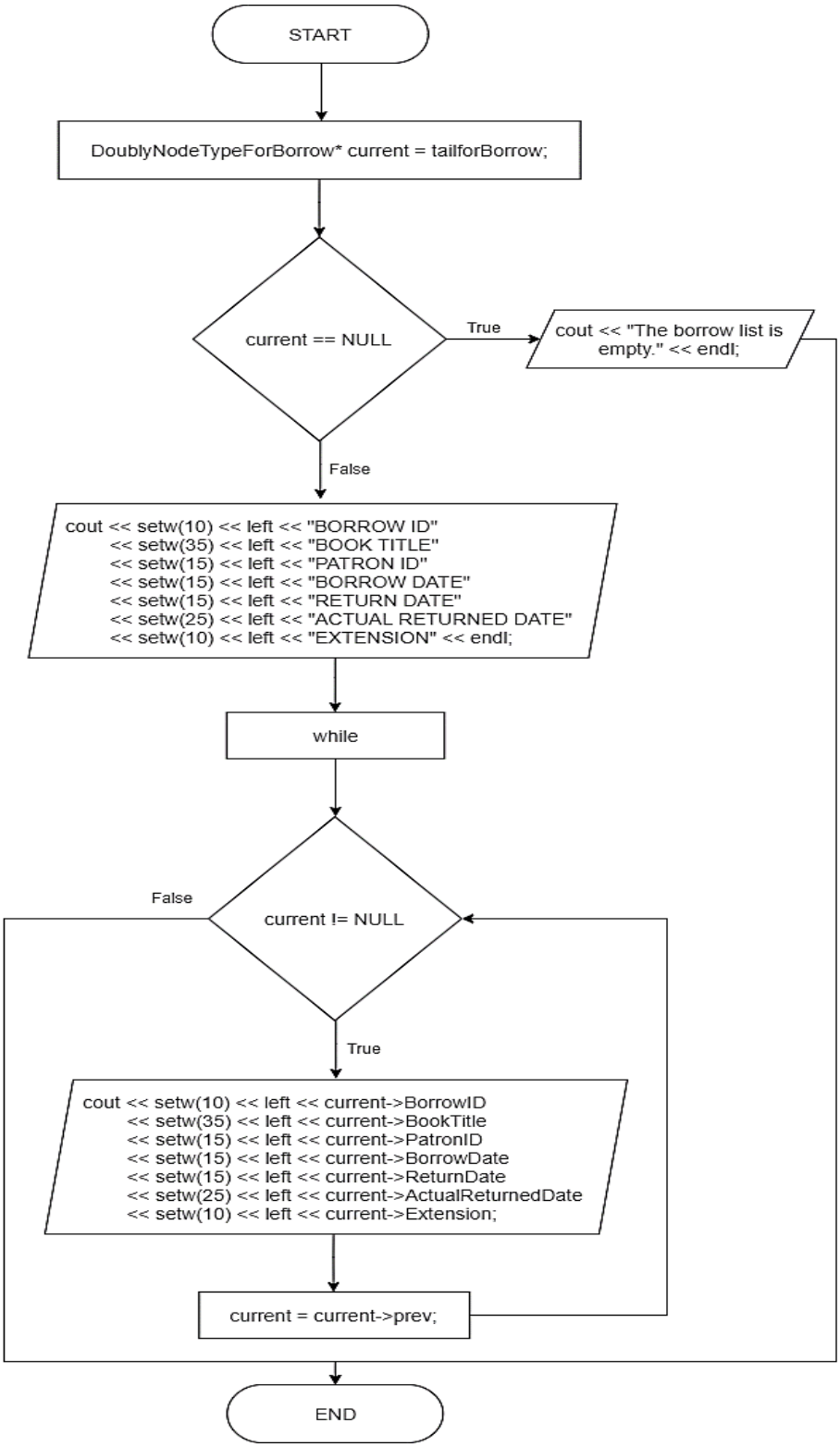


Figure 99: Borrow Books - Display all book borrowed transactions by all patrons (Behind) flowchart

Code Snippet

```
682 void BorrowList::PrintfromBehind() {
683     //Copy value of tailforBorrow into current
684     DoublyNodeTypeForBorrow* current = tailforBorrow;
685 
686     cout << endl << endl;
687     if(current == NULL) { //If the current is not pointing to any
688         cout << "The borrow list is empty." << endl;
689     } else {
690         //Print the header of borrow list
691         cout << setw(10) << left << "BORROW ID" << setw(35) << left << "BOOK TITLE" << setw(15) << left << "PATRON ID" << setw(15)
                << left << "BORROW DATE" << setw(15) << left << "RETURN DATE" << setw(25) << left << "ACTUAL RETURNED DATE" << setw(10)
                << left << "EXTENSION" << endl;
692         while (current != NULL) { //Iterate if the current is not NULL
693             //Print all records of borrow list
694             cout << setw(10) << left << current->BorrowID << setw(35) << left << current->BookTitle << setw(15) << left <<
                    current->PatronID << setw(15) << left << current->BorrowDate << setw(15) << left << current->ReturnDate << setw(25)
                    << left << current->ActualReturnedDate << setw(10) << left << current->Extension;
695             current = current->prev; //Move to the previous borrow record
696             cout << endl;
697         }
698     }
699     cout << endl;
700 
701 };
```

Figure 100: PrintfromBehind method - borrowList

The Figure 100 shown above is the PrintfromBehind method. Line 684 is the variable declaration. Line 687 is to check if the borrow is pointing to the NULL, if it does, then Line 688 will be executed. Otherwise, Line 691 to 696 will be executed. Line 691 is to print the headers, which to show what are the columns. Line 694 to 696 will be executed only if the pointing is not pointing to NULL as shown in Line 692. Line 694 is to print the records based on the current pointer. Line 695 is to move to the next borrow record.

Result

```
BORROW MENU:
1 – Add a new book borrow transaction
2 – Print
3 – Make Extension
4 – Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 2
1 – Print from beginning
2 – Print from behind
3 – Print last 10 book borrowed by patrons
Your option is: 2


BORROW ID BOOK TITLE                      PATRON ID      BORROW DATE    RETURN DATE    ACTUAL RETURNED DATE     EXTENSION
13        CRIME AND PERIODICALS           LWY002         30/8/2020      14/9/2020      NULL                     0
12        THE HERO WITH A THOUSAND FACES  HKJ001         30/8/2020      14/9/2020      NULL                     0
11        I HAVE A DREAM                  HKJ001         30/8/2020      14/9/2020      30/8/2020                0
10        A NEW EARTH                     JD001          30/8/2020      14/9/2020      NULL                     0
9         A NEW EARTH                     LYK005         30/8/2020      14/9/2020      NULL                     0
8         HARRY POTTER                    TKM007         30/8/2020      14/9/2020      NULL                     0
7         HARRY POTTER                    NZJ004         30/8/2020      14/9/2020      NULL                     0
6         MEIN KAMPF                      HKJ001         30/8/2020      14/9/2020      30/8/2020                0
5         MEIN KAMPF                      TYJ008         30/8/2020      14/9/2020      NULL                     0
4         THE INVISIBLE MAN               LBK003         30/8/2020      14/9/2020      NULL                     0
3         THE INVISIBLE MAN               LWY002         30/8/2020      14/9/2020      NULL                     0
2         13 REASONS WHY                  HKJ001         30/8/2020      29/9/2020      NULL                     1
1         13 REASONS WHY                  JD001          30/8/2020      14/9/2020      NULL                     0
```

Figure 101: Sample output 1 - PrintfromBehind method - borrowList

The Figure 101 shown above is the sample output for PrintfromBehind method. After entering the print borrow transaction section, the user is required to enter a number from 1 to 3 to print the borrow transaction. If the user wants to print the borrow transaction from behind, then the user can enter 2 as input to view the result.

3.3.4 Return book

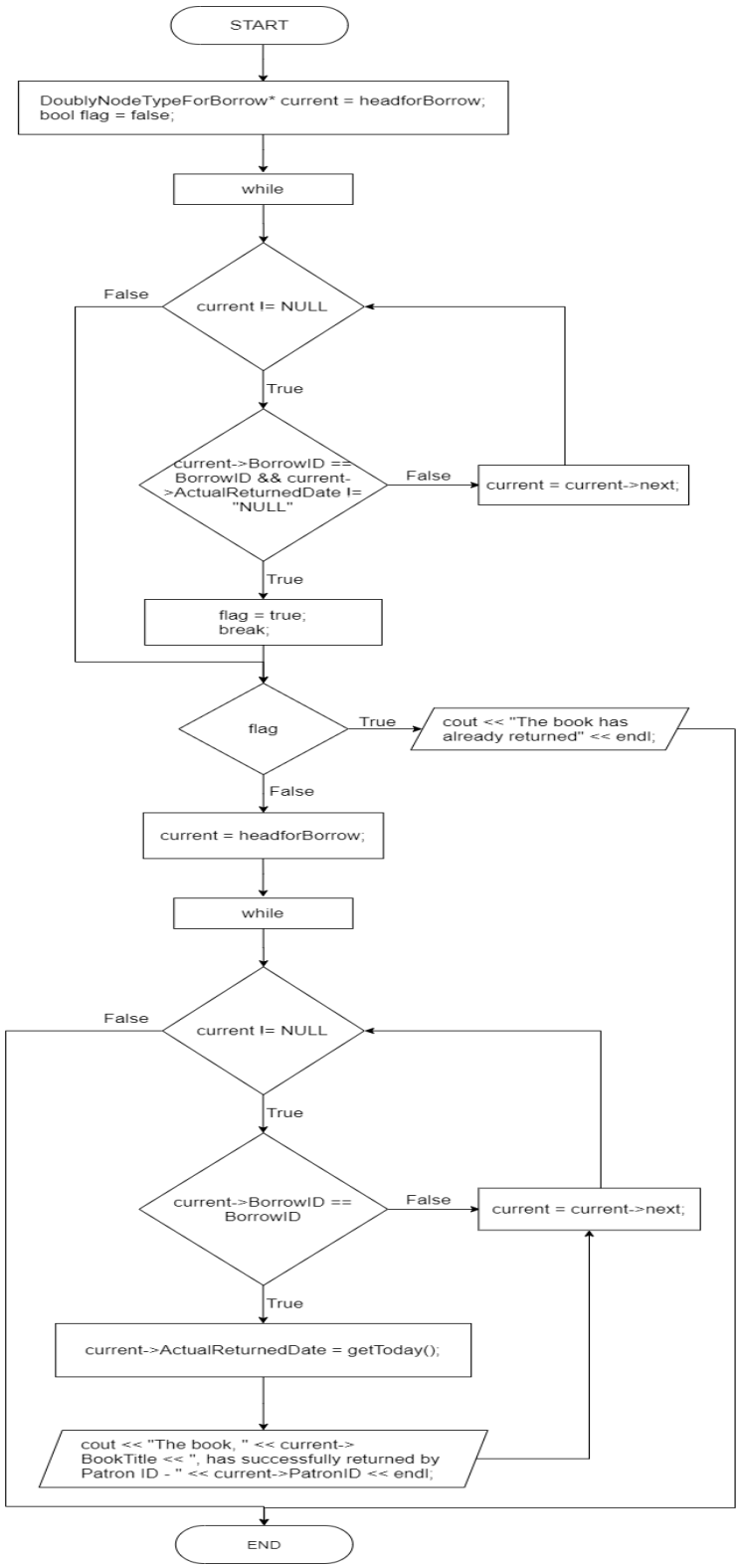


Figure 102: Borrow Books - Return book flowchart

Code Snippet

```
833 void BorrowList::UpdateActualReturnedDate(int BorrowID) {
834     //Copy value of headforBorrow into current
835     DoublyNodeTypeForBorrow* current = headforBorrow;
836     bool flag = false;
837     while (current != NULL) { //Iterate if the current is not NULL
838         //Check if the given borrow ID is matched with the borrow ID of borrow list and its returned date is not NULL
839         if (current->BorrowID == BorrowID && current->ActualReturnedDate != "NULL") {
840             flag = true; //Set true value to flag
841             break; //Break the iteration
842         }
843         current = current->next; //Move to the next borrow
844     }
845     if (flag) { //If the returned date of the given borrow ID is not NULL
846         cout << "The book has already returned" << endl;
847     }
848     else {
849         current = headforBorrow; //Copy value of headforBorrow into current
850         while (current != NULL) { //Iterate if the current is not NULL
851             if (current->BorrowID == BorrowID) { //Check if the borrow ID of borrow list is matched with the given borrow ID
852                 current->ActualReturnedDate = getToday(); //Update the actual returned date to be today
853                 cout << "The book, " << current->BookTitle << ", has successfully returned by Patron ID - " << current->PatronID;
854                 cout << endl;
855             }
856             current = current->next; //Move to the next borrow record
857         }
858 
859     }
860 
861 }
```

Figure 103: UpdateActualReturnedDate method - borrowList

The Figure 103 is the UpdateActualReturnedDate method. Line 835 to 836 are the variable declaration. Line 837 to 844 is to check if the given borrow ID is matched with the borrow list, while the actual return date is equal to NULL. Line 839 to 843 will be executed only if the borrow is not pointing to NULL. Line 839 is to check the given borrow ID is matched with the borrow ID in the borrow list, and its actual returned date is NULL, then Line 840 to 841 will be executed. Line 840 is to set true value to flag, and Line 841 is to break the iteration. Line 843 is to move to the next borrow record. If the condition met in Line 839, then Line 843 will never be executed. Line 845 is to check if the flag value is true, if it does, then Line 846 will be executed, which means the condition is met in the Line 837 to 844. Otherwise, Line 849 to 857 will be executed. Line 849 is a pointer assignment. Line 851 to 856 will be executed only if the borrow is not pointing to NULL. Line 851 is to check if the given borrow ID is matched with the borrow ID in the borrow linked list, if it does, then update the actual returned date by calling getToday method and assign the value to the node. Line 856 is to move to the next copy.

Result

```
BORROW MENU:
1 – Add a new book borrow transaction
2 – Print
3 – Make Extension
4 – Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 4


BORROW ID BOOK TITLE                        PATRON ID
1         13 REASONS WHY                    JD001
2         13 REASONS WHY                    HKJ001
3         THE INVISIBLE MAN                 LWY002
4         THE INVISIBLE MAN                 LBK003
5         MEIN KAMPF                        TYJ008
7         HARRY POTTER                      NZJ004
8         HARRY POTTER                      TKM007
9         A NEW EARTH                       LYK005
10        A NEW EARTH                       JD001
12        THE HERO WITH A THOUSAND FACES    HKJ001
13        CRIME AND PERIODICALS             LWY002

Please enter the borrow ID: 10
The book, A NEW EARTH, has successfully returned by Patron ID – JD001
```

Figure 104: Sample output 1 - UpdateActualReturnedDate method - borrowList

The Figure 104 shown above is the sample output for UpdateActualReturnedDate method. The user is required to enter 4 to return a book. After entering 4, the user is required to provide the borrow ID to return the book, the borrow ID should be within the borrow list as shown in the picture. If the borrow ID is found within the borrow list, then the user able to return the book.

```
BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 4


BORROW ID BOOK TITLE                          PATRON ID
1         13 REASONS WHY                      HKJ001
2         13 REASONS WHY                      HKJ001

Please enter the borrow ID: 2
The book, 13 REASONS WHY, has successfully returned by Patron ID - HKJ001


BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 4


BORROW ID BOOK TITLE                          PATRON ID
1         13 REASONS WHY                      HKJ001

Please enter the borrow ID: 2
The book has already returned
```

Figure 105: Sample output 2 - UpdateActualReturnedDate method - borrowList

PrintPatronID

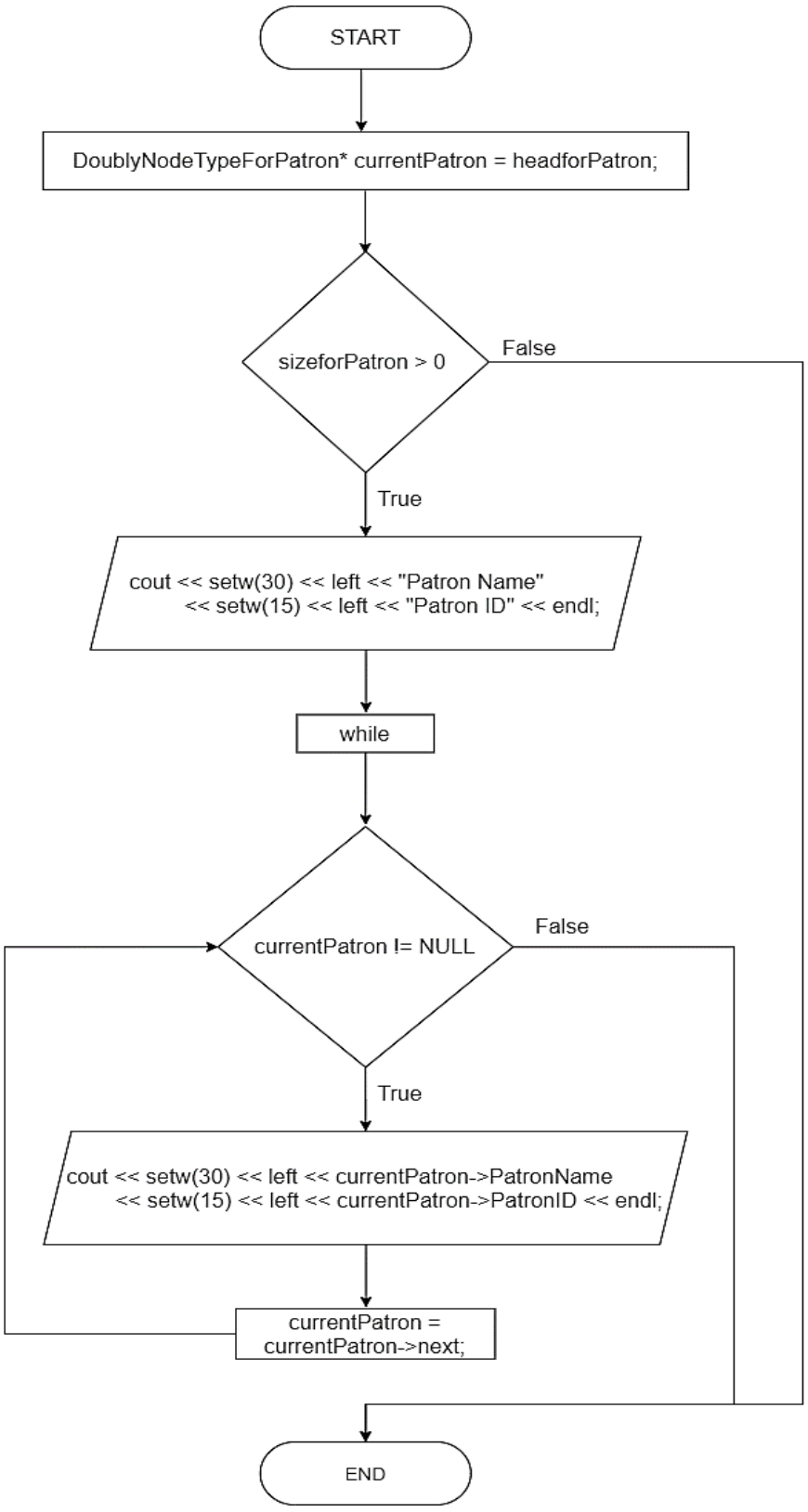


Figure 106: Patrons - Print PatronID method flowchart

Code Snippet

```
492 void PatronList::PrintPatronID() {
493     //Copy value of headforPatron into currentPatron
494     DoublyNodeTypeForPatron* currentPatron = headforPatron;
495     cout << endl << endl;
496     //If size of patron is greater than 0
497     if(sizeforPatron > 0) {
498         //Print the header of patron
499         cout << setw(30) << left << "Patron Name" << setw(15) << left << "Patron ID" << endl;
500         //Iterate if the currentPatron is not NULL
501         while(currentPatron != NULL) {
502             //Print all the records of patron
503             cout << setw(30) << left << currentPatron -> PatronName << setw(15) << left << currentPatron -> PatronID << endl;
504             //Move to the next patron
505             currentPatron = currentPatron -> next;
506         }
507     }
508     cout << endl;
509 };
```

Figure 107: PrintPatronID method - patronList

The Figure 107 shown above is the PrintPatronID method. Line 494 is the variable declaration. Line 497 is to check if the size for patron is greater than 0, which means the patron list will have at least one record. If it does, Line 499 to 505 will be executed. Line 499 is to print the headers, which is showing what are the columns to be displayed. Line 503 to 505 will be executed only if the patron is not pointing to NULL. Line 503 is to print the records based on the current pointer. Line 505 is to move to the next patron.

SetBookToUnavailable

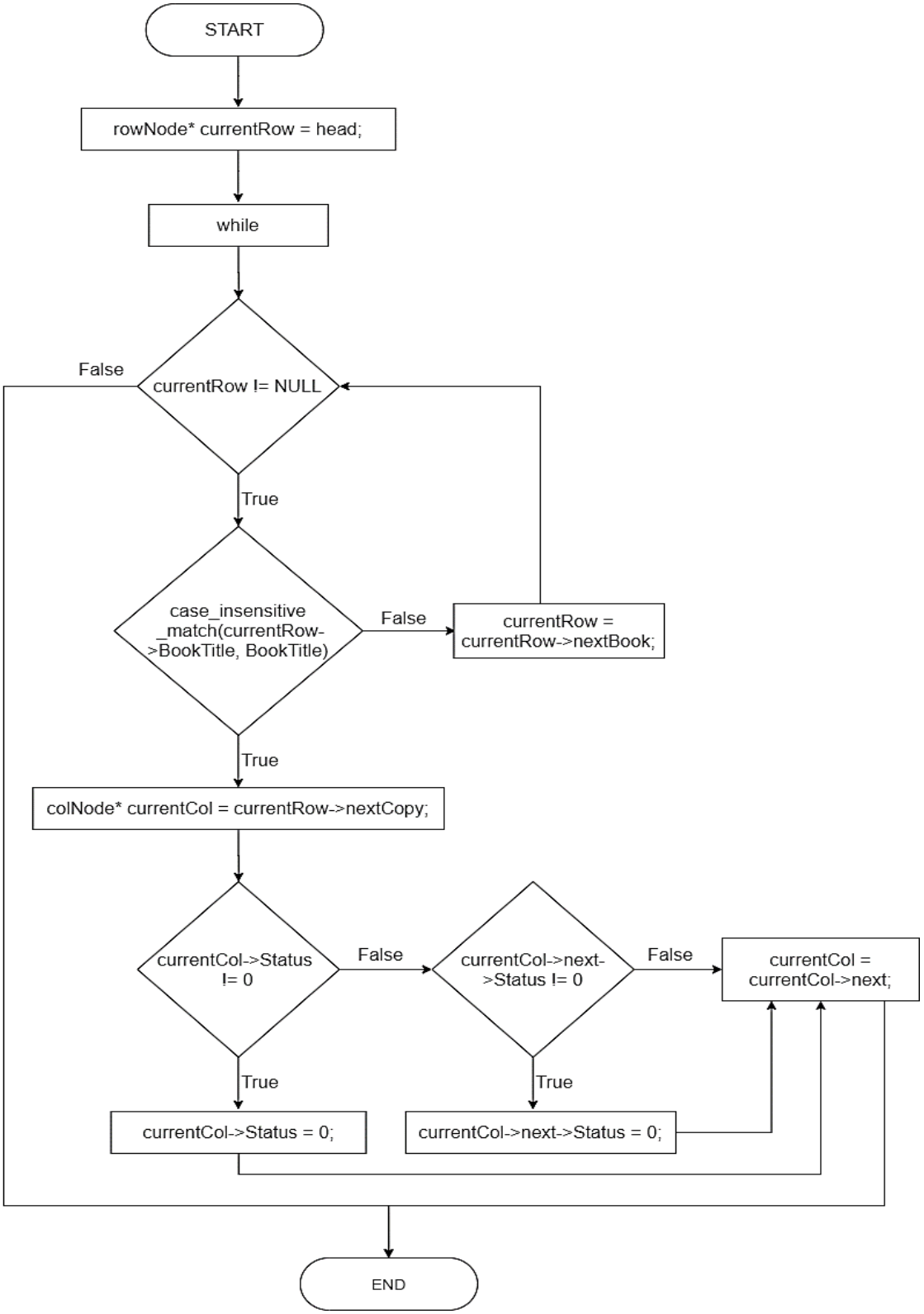


Figure 108: SetBookToUnavailable method flowchart

Code Snippet

```
659 void BorrowList::SetBookToUnavailable(string BookTitle) {
660     //Copy value of head into currentRow
661     rowNode* currentRow = head;
662     while(currentRow != NULL) { //Iterate if the currentRow is not NULL
663         //Check if the given BookTitle is matched with the BookTitle of borrow list
664         if(case_insensitive_match(currentRow -> BookTitle, BookTitle)) {
665             //Copy value of nextCopy of currentRow into currentCol
666             colNode* currentCol = currentRow -> nextCopy;
667             if(currentCol -> Status != 0) { //If the status of first copy is not 0
668                 currentCol -> Status = 0; //Assign 0 to the status
669             } else { //If the status of first copy is 0
670                 if(currentCol -> next -> Status != 0) { //Check the status of second copy is not 0
671                     currentCol -> next -> Status = 0; //Assign 0 to the status
672                 }
673             }
674             //Move to the next copy
675             currentCol = currentCol -> next;
676         }
677         //Move to the next book
678         currentRow = currentRow -> nextBook;
679     }
680 };
```

Figure 109: SetBookToUnavailable method - borrowList

The Figure 109 shown above is the SetBookToUnavailable method. Line 661 is the variable declaration. Line 664 to 678 will be executed only if the book pointer is not pointing to NULL. Line 664 is to check if the given book title is matched with the book title in book linked list by converting to the lower cases, if it matches, then Line 666 to 675 will be executed. Line 666 is another pointer variable declaration. Line 667 to 673 is to check if the status is equal to 0, if not, then assign 0 to the status. Line 675 is to move to the next copy, whereas Line 678 is to move to the next book.

getToday

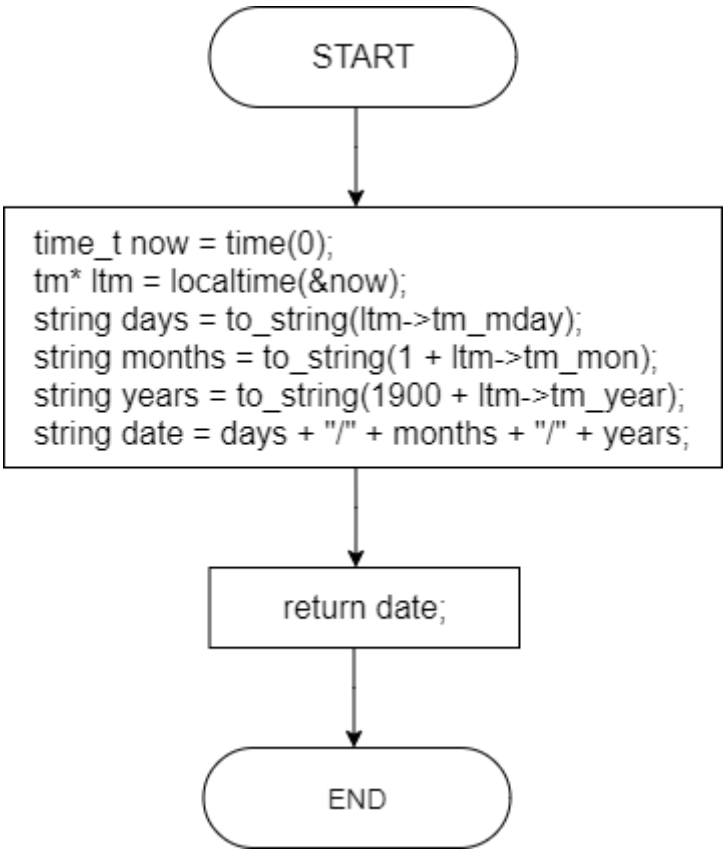


Figure 110: getToday method flowchart

Code Snippet

```
703 string BorrowList::getToday() {
704     //This method is to get today date as string by using its local time
705     time_t now = time(0);
706     tm* ltm = localtime(&now);
707     string days = to_string(ltm->tm_mday);
708     string months = to_string(1 + ltm->tm_mon);
709     string years = to_string(1900 + ltm->tm_year);
710     string date = days + "/" + months + "/" + years;
711     return date;
712 }
```

Figure 111: getToday method - borrowList

The Figure 111 shown above is the getToday method. Line 705 to 706 is utilizing the library where <c.time> is included in the beginning. Line 707 to 709 is to access its day, month, and year, and assign to the variable of days, months, and years respectively. Line 710 is to concatenate the three string variables together by using / as the delimiter. Line 711 is to return the joined string to its method.

tokenize

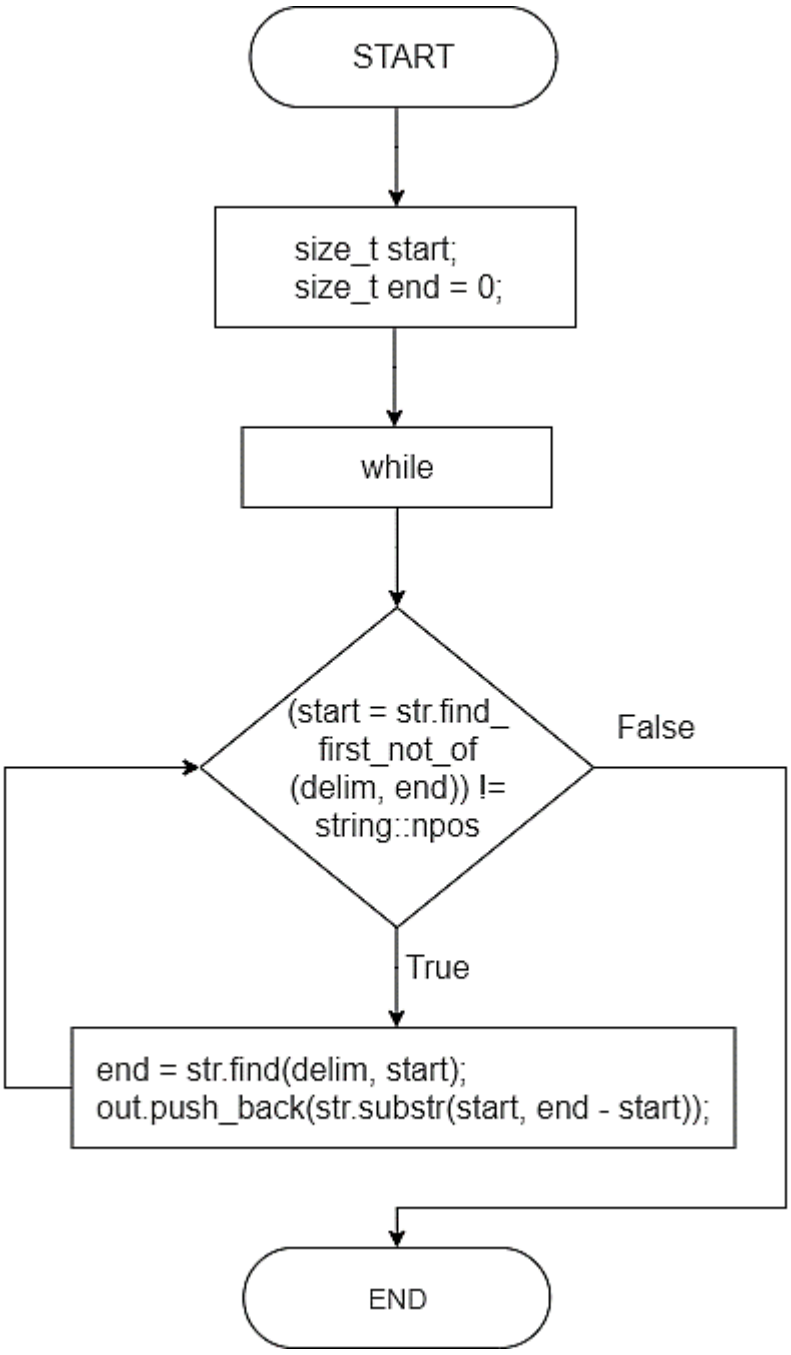


Figure 112: tokenize method flowchart

Code Snippet

```
714 void tokenize(string const& str, const string delim, vector<string>& out)
715 {
716
717     size_t start;
718     size_t end = 0;
719     while ((start = str.find_first_not_of(delim, end)) != string::npos)
720     {
721         end = str.find(delim, start);
722         out.push_back(str.substr(start, end - start));
723     }
724 };
```

Figure 113: Tokenize method

The Figure 113 shown above is the Tokenize method. Line 717 to 718 are the variable declaration. Line 719 to 723 is to keep iterating the given string by splitting the string based on the given delimiters / until the string to be empty as shown in 719. The split string will be directly assigned into the vector (out) based on the referencing until the iteration is stopped as shown Line 721 to 722.

PrintBorrowID

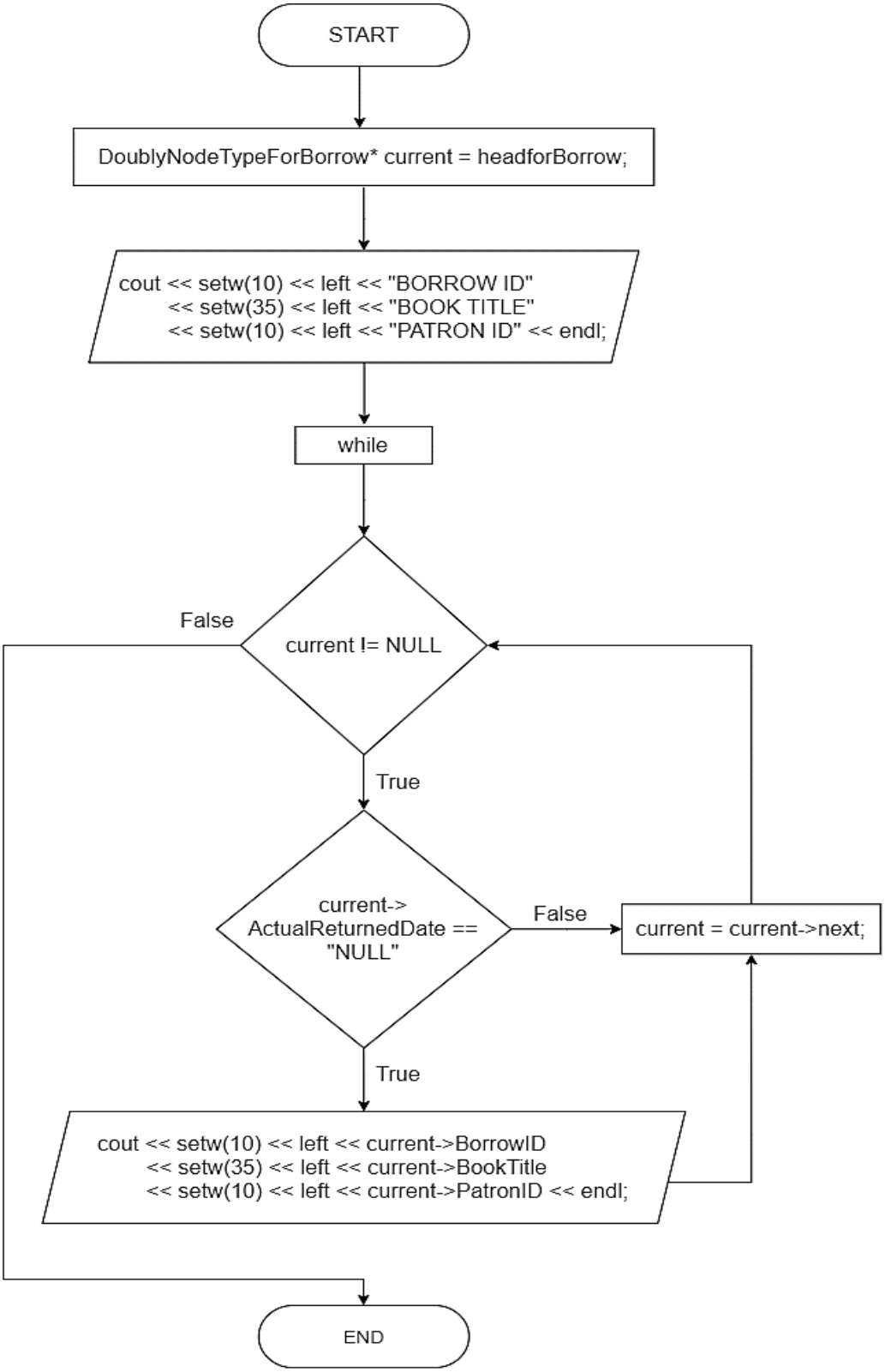


Figure 114: PrintBorrowID method flowchart

Code Snippet

```
726 void BorrowList::PrintBorrowID() {
727     //Copy value of headforBorrow into current
728     DoublyNodeTypeForBorrow* current = headforBorrow;
729     cout << endl << endl;
730     cout << setw(10) << left << "BORROW ID" << setw(35) << left << "BOOK TITLE" << setw(10) << left << "PATRON ID" << endl;
731     while(current != NULL) { //Iterate if the current is not NULL
732         if(current -> ActualReturnedDate == "NULL") { //Check if the actual returned date is not NULL
733             //Print all records of borrow list
734             cout << setw(10) << left << current -> BorrowID << setw(35) << left << current -> BookTitle << setw(10) << left <<
                    current -> PatronID << endl;
735         }
736         current = current -> next; //Move to the next borrow
737     }
738     cout << endl;
739 }
```

Figure 115: PrintBorrowID method

The Figure 115 shown above is the PrintBorrowID method. Line 728 is the variable declaration. Line 730 is to print the headers, which is to show what are the columns to be displayed. Line 731 is to check if the borrow linked list is not pointing to NULL, then Line 732 to 736 will be executed. Line 732 is to check if the actual returned date of the current borrow list is NULL, if it does, then will be printing the record as shown in Line 734. Line 736 is to move to the next borrow record.

UpdateExtensionDate

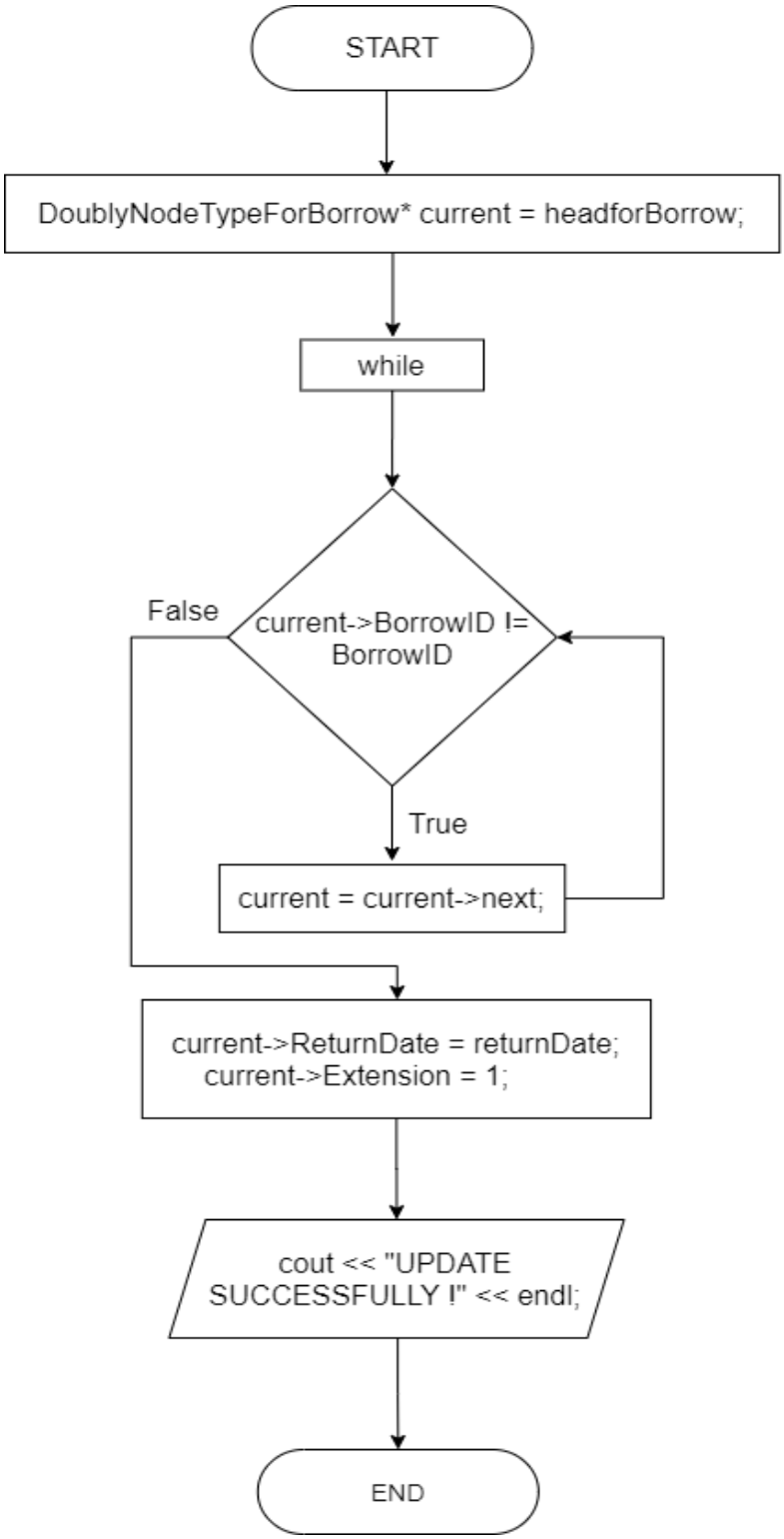


Figure 116: UpdateExtensionDate method flowchart

Code Snippet

```
741 void BorrowList::UpdateExtensionDate(int BorrowID, string returnDate) {
742     //Copy value of headforBorrow into current
743     DoublyNodeTypeForBorrow* current = headforBorrow;
744     while (current->BorrowID != BorrowID) { //Iterate if the given borrow ID is not matched with the borrow ID of borrow list
745         current = current->next; //Move to the next borrow record
746     }
747     current->ReturnDate = returnDate; //Set the new return date
748     current->Extension = 1; //Update the extension status
749     cout << "UPDATE SUCCESSFULLY !";
750     cout << endl;
751 }
```

Figure 117: UpdateExtensionDate method - borrowList

The Figure 117 shown above is the UpdateExtensionDate method. Line 743 is the variable declaration. Line 744 to 746 is to iterate until the given borrow ID is matched with the borrow ID in borrow linked list. Line 747 to 748 is to update the return date and also the extension value, to indicate an extension is made for this borrow record.

getReturnedDate

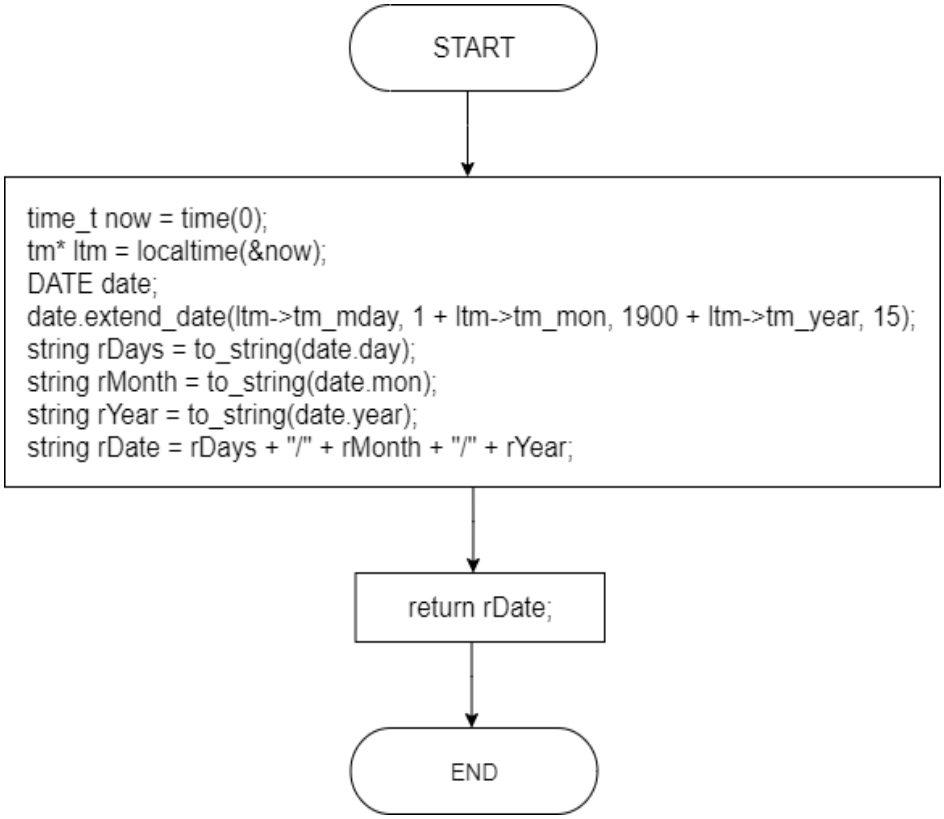


Figure 118: getReturnedDate method flowchart

Code Snippet

```
820 string BorrowList::getReturnedDate() {
821     //This method will return a new returned date which after 15 days
822     time_t now = time(0);
823     tm* ltm = localtime(&now);
824     DATE date;
825     date.extend_date(ltm->tm_mday, 1 + ltm->tm_mon, 1900 + ltm->tm_year, 15);
826     string rDays = to_string(date.day);
827     string rMonth = to_string(date.mon);
828     string rYear = to_string(date.year);
829     string rDate = rDays + "/" + rMonth + "/" + rYear;
830     return rDate;
831 }
```

Figure 119: getReturnedDate method - borrowList

The Figure 119 shown above is the getReturnedDate method. Line 822 to 823 are utilizing the library (<c.time>) plugin function, which already declared in the beginning. Line 824 is an instance of class, and Line 825 is to access the extend_date method of DATE class. line 826 to 828 is to access the day, month, and year, and assign to the variables of rDays, rMonth, rYear respectively. Line 829 to concatenate the three variables by using / as delimiters to store into a rDate variable. Line 830 is to return the joined string to its method.

ExtendDate

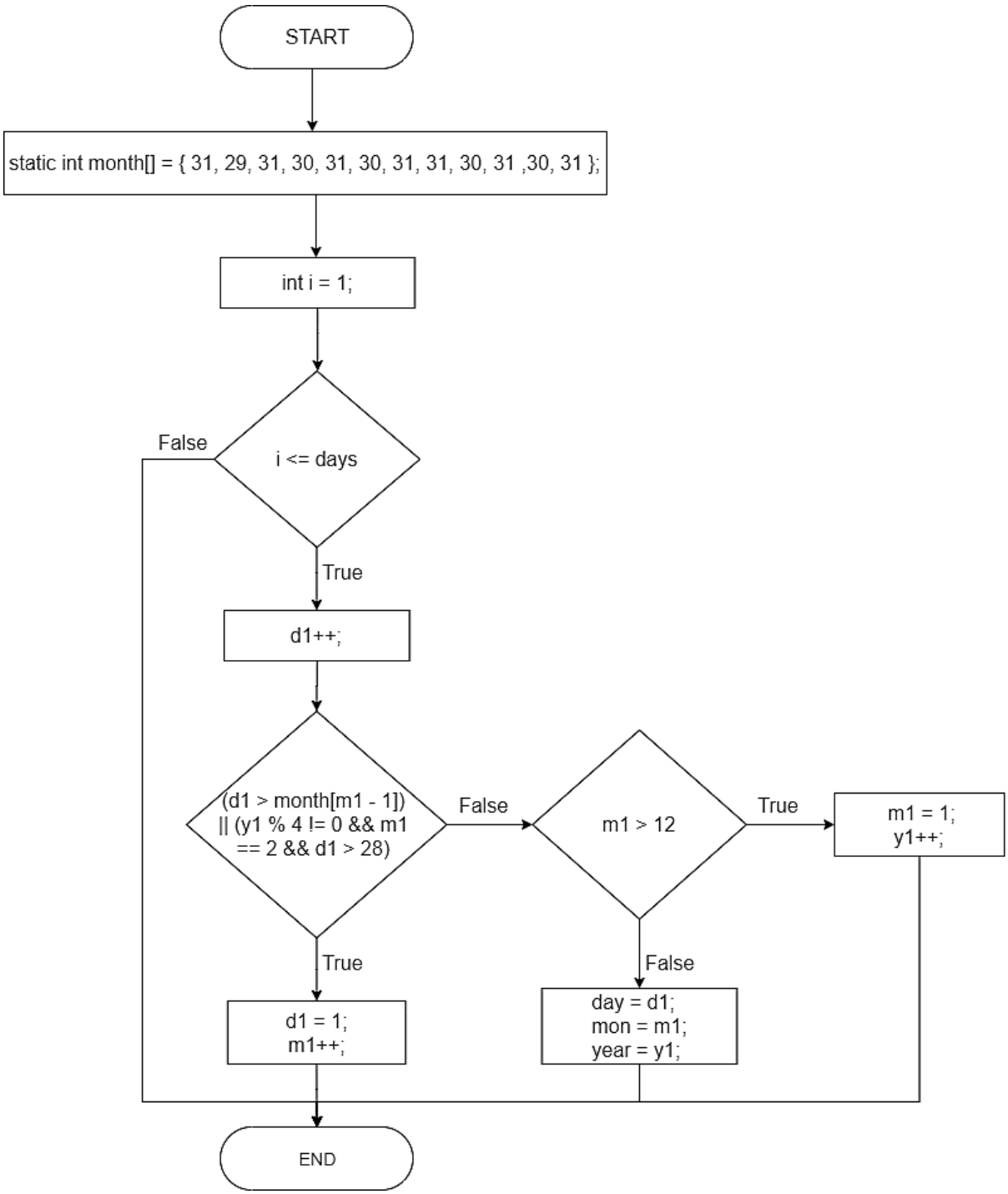


Figure 120: ExtendDate method flowchart

Code Snippet

```
863 void DATE::extend_date(int d1, int m1, int y1, int days) {
864     static int month[] = { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31 ,30, 31 };
865     for (int i = 1; i <= days; i++) {
866         d1++;
867         if ((d1 > month[m1 - 1]) || (y1 % 4 != 0 && m1 == 2 && d1 > 28)) {
868             d1 = 1;
869             m1++;
870         }
871         if (m1 > 12) {
872             m1 = 1;
873             y1++;
874         }
875         day = d1;
876         mon = m1;
877         year = y1;
878     }
879 }
```

Figure 121: Extend_Date method - DATE

The Figure 121 shown above is the Extend_Date method. Line 864 is the array list of the total days for each moth. Line 865 will start to run the for loop. Line 865 is to iterate the loop based on the given days, at the end of the iteration loop, will be increasing 1 as increment to make the iteration stop at the end. Line 867 is to check if the given day (d1) is greater than the days in the month array list, meanwhile, (y1 % 4 != 0 && m1 == 2 && d1 > 28) is to check if the given year is leap year, if it does, Line 868 to 869 will be executed. Line 871 is to check if the given month is greater than 12, if it does, then Line 872 to 873 will be executed, which means to increment 1 to years to indicate the year is next month and set the month to be January (1). Assigning all the value to the variables of day, month, and year respectively as shown in Line 875 to 877.

AddSecondCopy

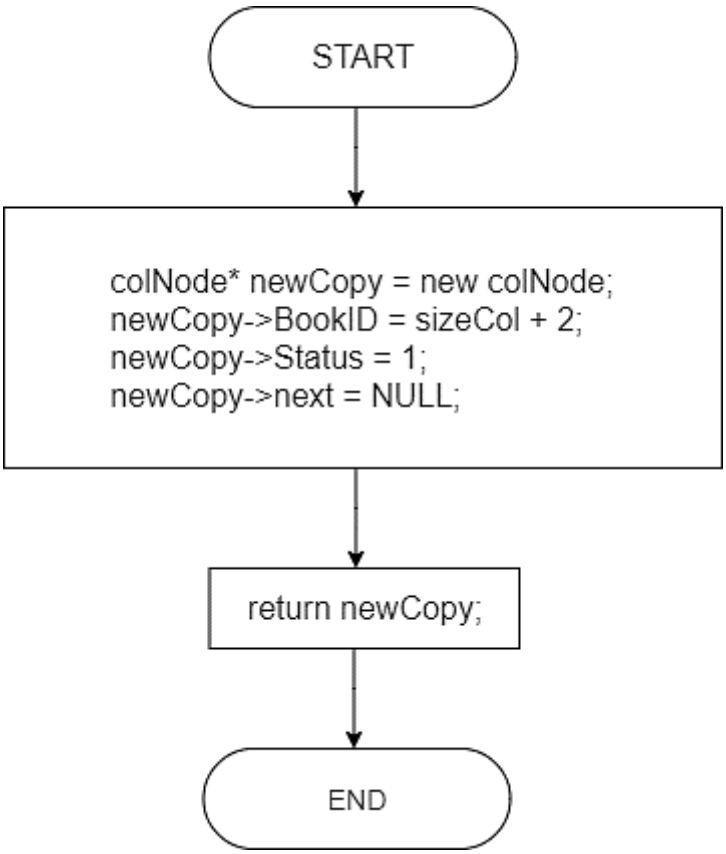


Figure 122: AddSecondCopy method flowchart

Code Snippet

```
882 colNode* BookList::AddSecondCopy() {
883     //This method will create the second copy for the book
884     colNode* newCopy = new colNode;
885     newCopy -> BookID = sizeCol + 2;
886     newCopy -> Status = 1;
887     newCopy -> next = NULL;
888     return newCopy;
889 }
```

Figure 123: AddSecondCopy method - bookList

The Figure 123 shown above is the AddSecondCopy method. Line 884 is the variable declaration. Line 885 to 887 is to create the second copy for the book and set the status for the book as “1” which mean is available and set the next node to NULL . Line 888 will then return the new copy of the book.

AddFirstCopy

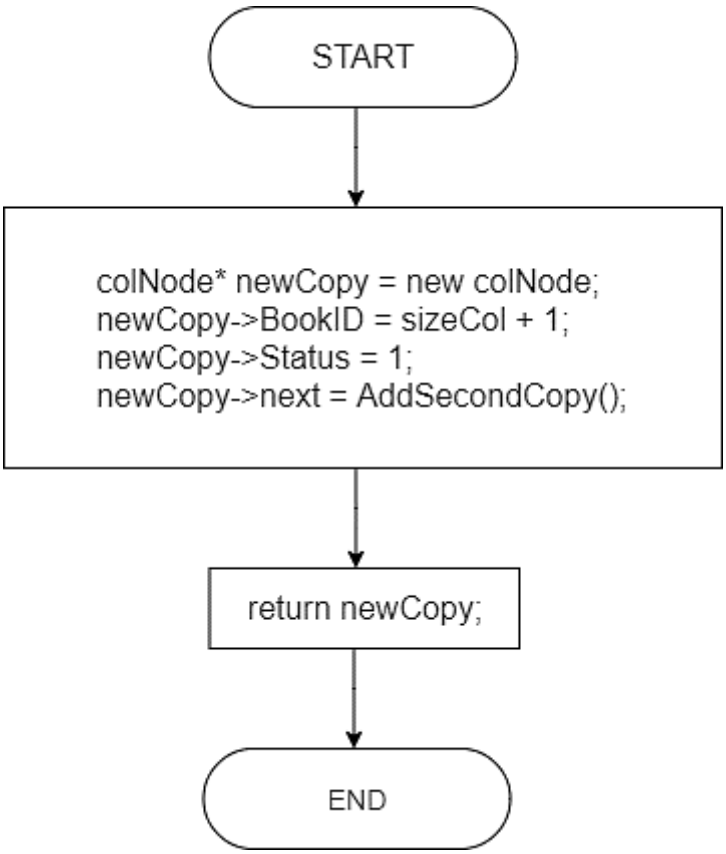


Figure 124: AddFirstCopy method flowchart

Code Snippet

```
891 colNode* BookList::AddFirstCopy() {
892     //This method will create the first copy for the book
893     colNode* newCopy = new colNode;
894     newCopy -> BookID = sizeCol + 1;
895     newCopy -> Status = 1;
896     newCopy -> next = AddSecondCopy(); //Call the method to create the second copy
897     return newCopy;
898 };
```

Figure 125: AddFirstCopy

The Figure 125 shown above is the AddFirstCopy method. Line 893 is the variable declaration. Line 894 to 896 is to create the first copy for the book and set the status for the book as “1” which means s available and set the next node to the AddSecondCopy() method to add the second copy of the book. Line 897 will the return the new copy of the book.

PrintBorrowList

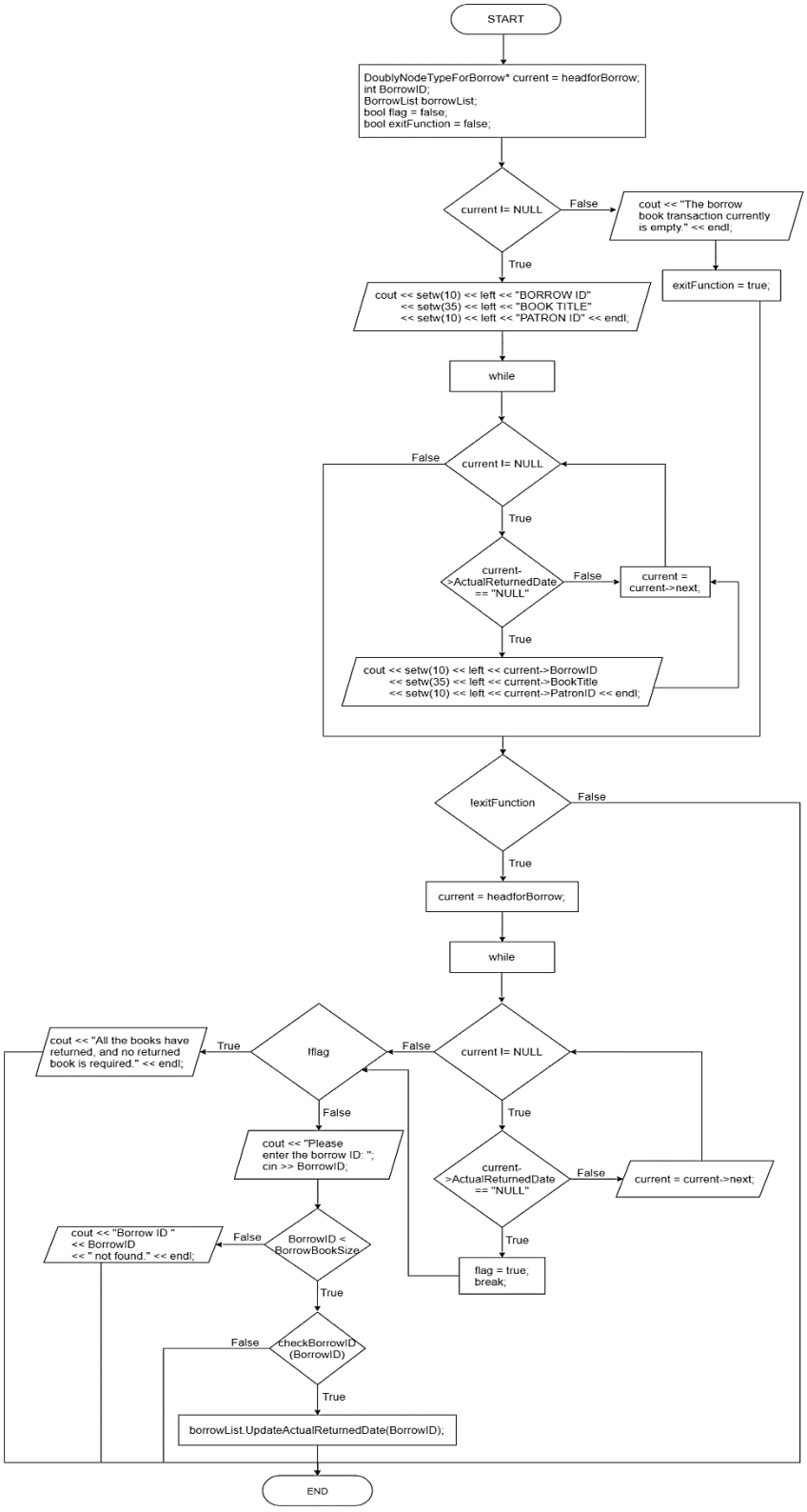


Figure 126: PrintBorrowList method flowchart

LINK:

https://drive.google.com/file/d/14BL6fNxOpc-QGb6orfBZAUzmewmmbOnn/view?usp=sharing

Code Snippet

```
943 void BorrowList::PrintBorrowList() {
944     DoublyNodeTypeForBorrow* current = headforBorrow; //Copy value of headforBorrow into current
945     int BorrowID;
946     BorrowList borrowList;
947 
948     bool flag = false;
949     bool exitFunction = false;
950 
951     cout << endl << endl;
952 
953     if (current != NULL) { //Check if the current is pointing to any
954         cout << setw(10) << left << "BORROW ID" << setw(35) << left << "BOOK TITLE" << setw(10) << left << "PATRON ID" << endl;
955         while (current != NULL) { //Iterate if the current is not NULL
956             if (current->ActualReturnedDate == "NULL") { //Check if the actual returned date is NULL
957                 //Print all records of borrow list
958                 cout << setw(10) << left << current->BorrowID << setw(35) << left << current->BookTitle << setw(10) << left <<
                        current->PatronID << endl;
959             }
960             current = current->next; //Move to the next borrow
961         }
962         cout << endl;
963     }
964     else {
965         cout << "The borrow book transaction currently is empty." << endl;
966         exitFunction = true; //If the current is not pointing to any, set exitFunction to be true
967     }
968 
969     if (!exitFunction) {
970         //Check if all the books have returned.
971         current = headforBorrow; //Copy value of headforBorrow into current
972         while (current != NULL) { //Iterate if the current is not NULL
973             if (current->ActualReturnedDate == "NULL") { //Check if the actual returned date is NULL
974                 flag = true; //Set true value to flag
976                   break; //Break the iteration
977              }
978              current = current->next; //Move to the next borrow
979          }
980 
981          if (!flag) { //Check if all the books were returned
982              cout << "All the books have returned, and no returned book is required." << endl;
983          }
984          else {
985              cout << "Please enter the borrow ID: ";
986              cin >> BorrowID;
987 
988              if(BorrowID < BorrowBookSize) {
989                  if (checkBorrowID(BorrowID)) //Check if the borrow ID is existed
990                          borrowList.UpdateActualReturnedDate(BorrowID); //Update the actual returned date of borrow ID
991              }
992              else {
993                  cout << "Borrow ID " << BorrowID << " not found." << endl;
994              }
995 
996          }
997      }
998      cout << endl;
999 }
```

Figure 127: PrintBorrowList method - borrowList

The Figure 127 shown above is the PrintBorrowList method. Line 944 to 949 is the variable declaration. Line 953 is to check if the head of borrow is not pointing to the NULL, if it does, then Line 954 will be executed. Then, Line 955 to 961 will be iterated the whole borrow list to check if the actual returned date is NULL. Line 956 is responsible to check the given actual returned date and display the borrow records is the actual returned date is NULL. Line 960 is to move to the next borrow. If the pointer in the borrow list is NULL, then it will execute Line 965 to 966 and set the exit function to true. Line 969 is to check is the exit function is false. Line 971 is a pointer assignment. Line 974 to 976 will be executed only if the head of borrow is not pointing to NULL. Line 973 is to check if the actual date is equal to NULL, if it does, it will set true value to flag and break the iteration or else it will move to next borrow. Line 981

is to check if all the books are returned. If it does, Line 982 will be executed, or else Line 985 to 994 will be executed. Line 988 is to check if the given borrow ID is lesser than the size of borrow records, Line 989 is to check if borrow ID is existed. If it does, Line 990 will be executed to update the actual returned date for the borrow ID.

getRandomISBN

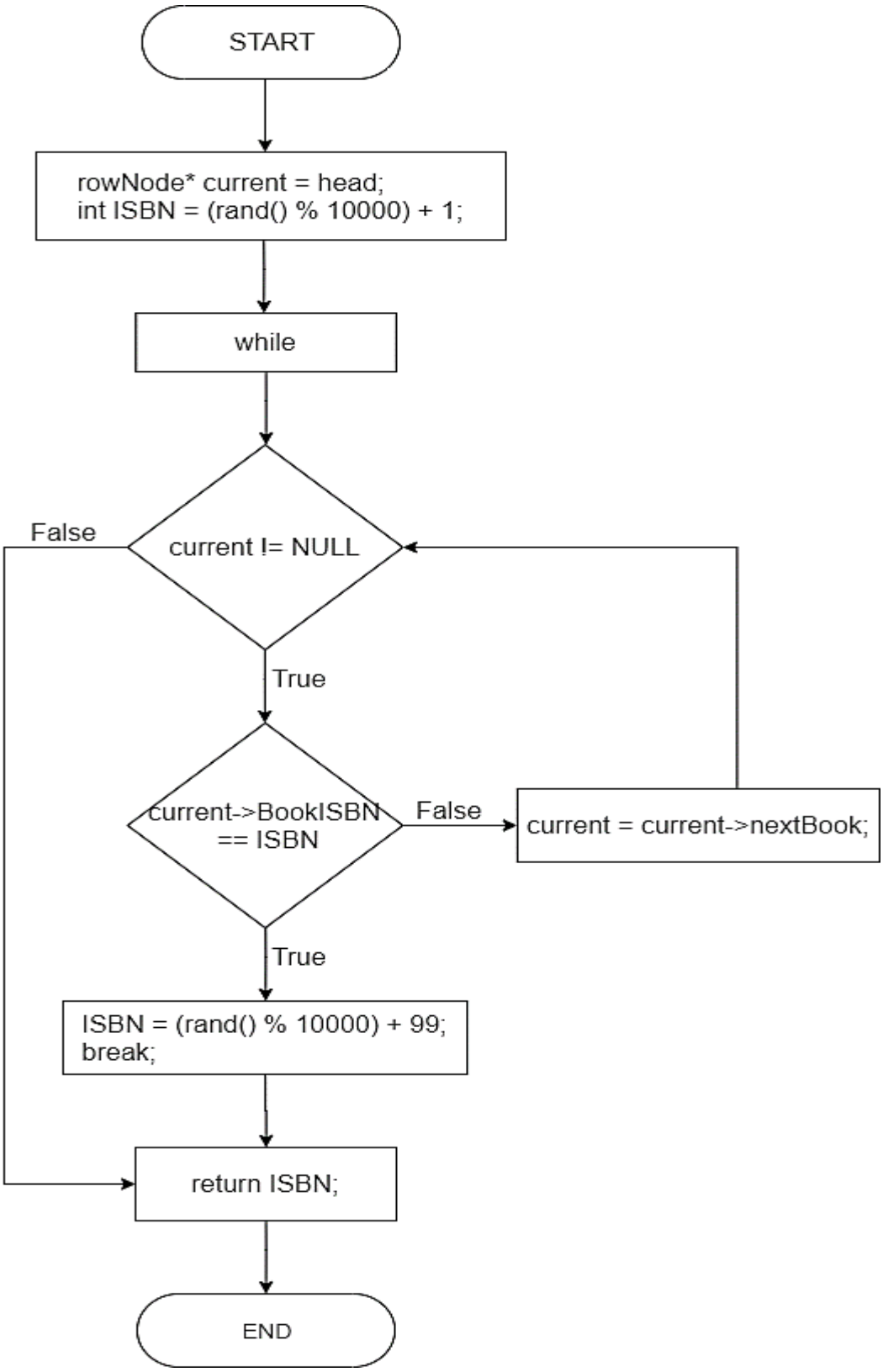


Figure 128: getRandomISBN method flowchart

Code Snippet

```
1256 int BookList::getRandomISBN() {
1257     //This method will get a random ISBN as integer
1258     rowNode* current = head;
1259     int ISBN = (rand()%10000)+1;
1260     while(current != NULL) {
1261         if(current -> BookISBN == ISBN) {
1262             ISBN = (rand()%10000)+99;
1263             break;
1264         }
1265         current = current -> nextBook;
1266     }
1267     return ISBN;
1268 };
```

Figure 129: getRandomISBN method - bookList

The Figure 129 shown above is the getRandomISBN method. Line 1258 to 1259 is the variable declaration. Line 1260 to 1266 is to check if the given book ISBN is matched. Line 1261 will be executed only if the book is not pointing to NULL. Line 1262 will randomly generate an ISBN number and break the iteration. Line 1265 is to move to the next book. Line 1267 will return an ISBN value.

PrintISBN

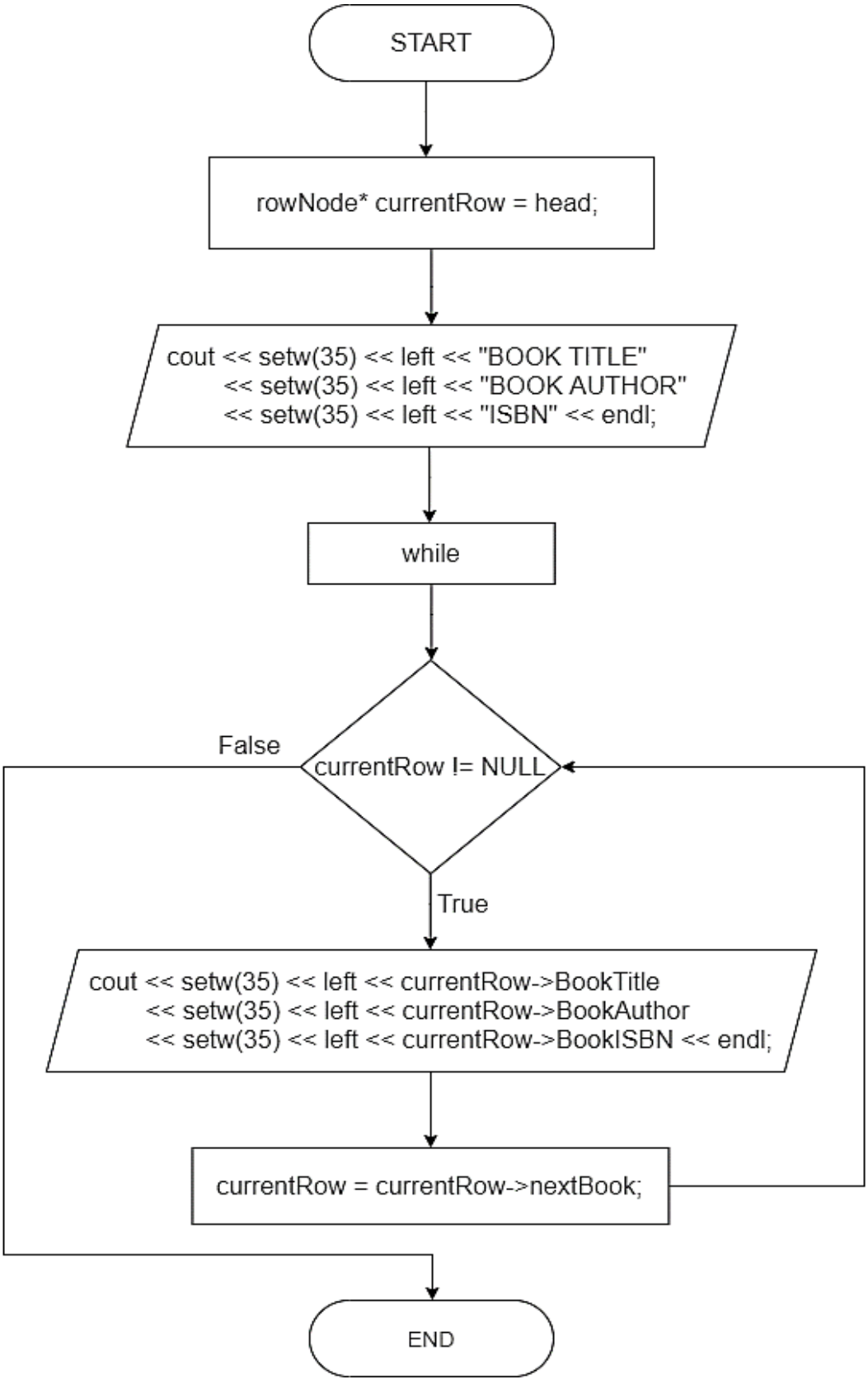


Figure 130: PrintISBN method flowchart

Code Snippet

```
1270 void BookList::PrintISBN() {
1271     rowNode* currentRow = head; //Copy value of head into currentRow
1272     cout << endl << endl;
1273     cout << setw(35) << left << "BOOK TITLE" << setw(35) << left << "BOOK AUTHOR" << setw(35) << left << "ISBN" <<endl;
1274     while (currentRow != NULL) { //Iterate if the currentRow is not NULL
1275         //Print all the records of book list
1276         cout << setw(35) << left << currentRow -> BookTitle << setw(35) << left << currentRow -> BookAuthor << setw(35) << left <<
             currentRow -> BookISBN << endl;
1277         currentRow = currentRow -> nextBook; //Move to the next book
1278     }
1279     cout << endl;
1280 }
```

Figure 131: PrintISBN method - bookList

The Figure 131 shown above is the PrintISBN method. Line 1271 is the variable declaration. Line 1273 is to print the headers, which is to show that are the columns to be displayed. Line 1274 is to check if the book list is not pointing to NULL, then it will print the record as shown in Line 1276 and Line 1277 is to move to the next book.

getFiction

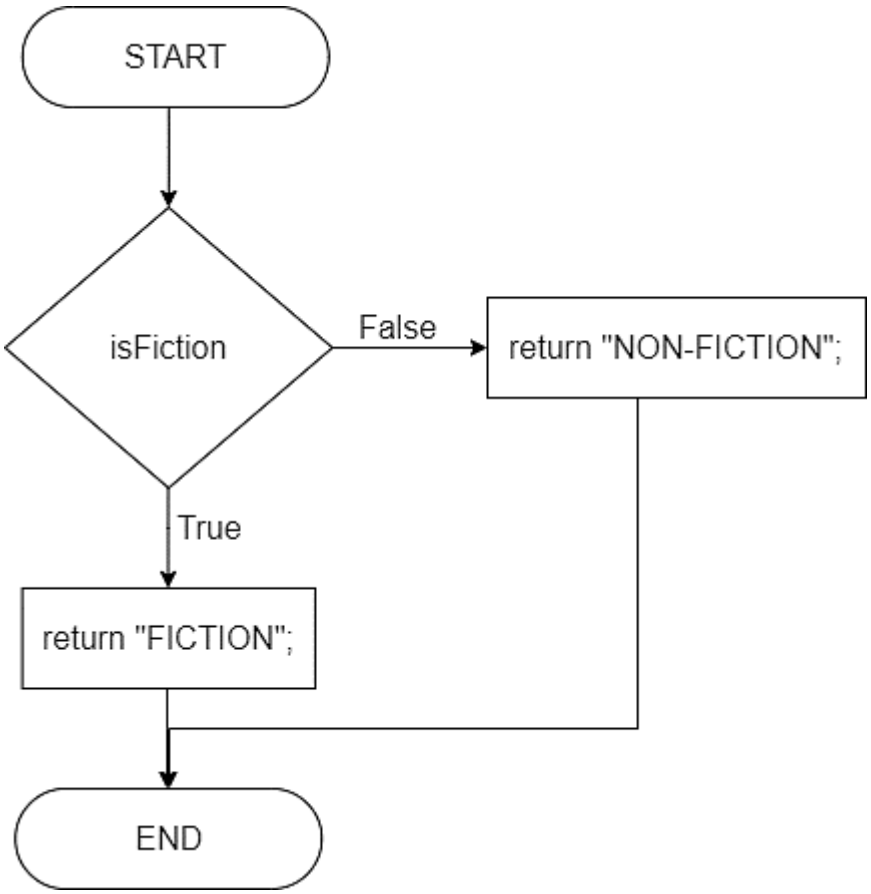


Figure 132: getFiction method flowchart

Code Snippet

```
1322 string getFiction(bool isFiction) {
1323     //This method is to return either FICTION or NON-FICTION based on the given boolean
1324     if(isFiction) {
1325         return "FICTION";
1326     } else {
1327         return "NON-FICTION";
1328     }
1329 };
```

Figure 133: getFiction method

The Figure 133 shown above is the getFiction method. This method it to return either Fiction or Non-Fiction as string type when passing the value from the object. The isFiction will only give based on boolean type and it will be controlled and called from other methods of linked list such as book.

getGenre

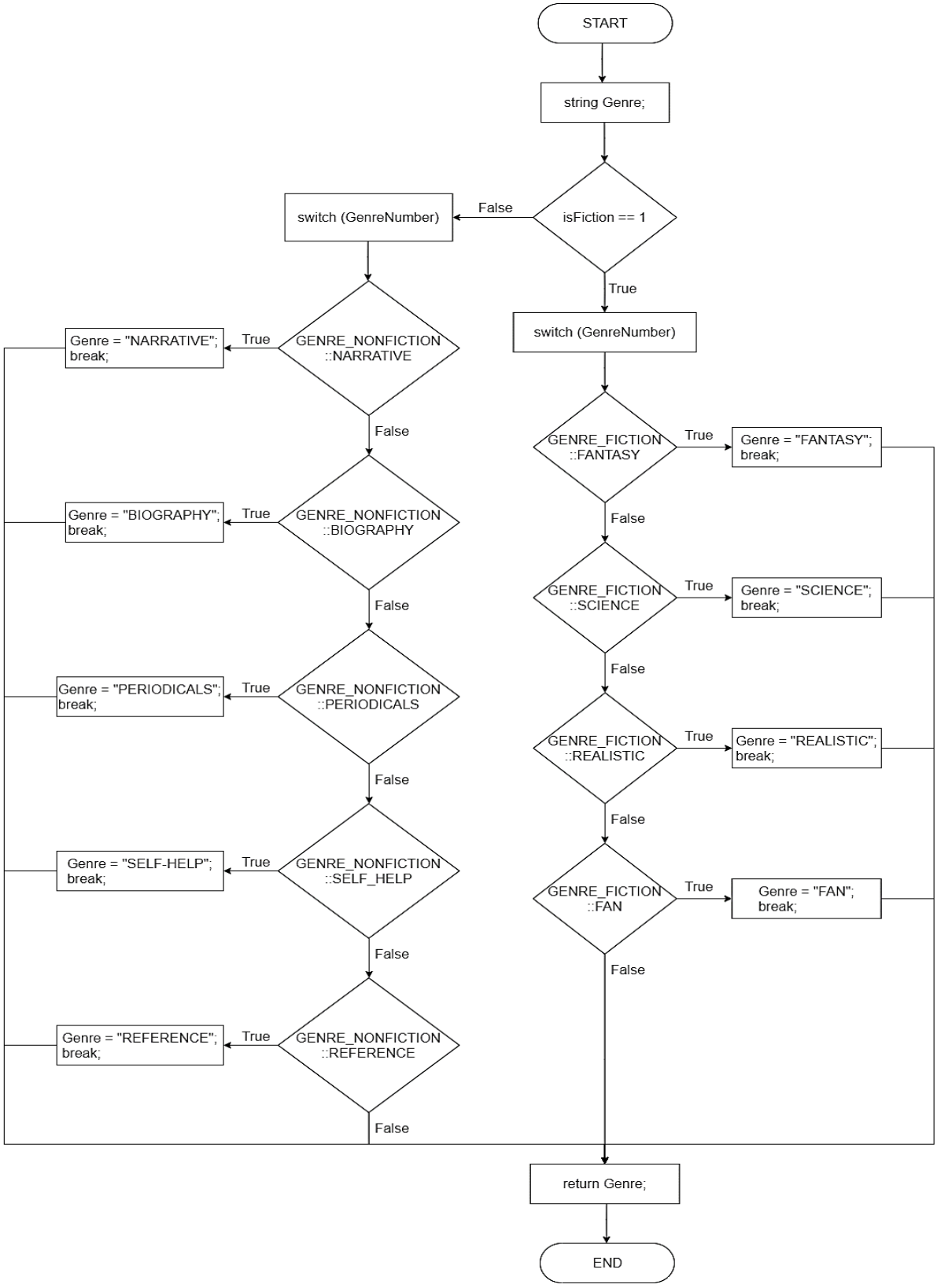


Figure 134: getGenre method flowchart

Code Snippet

```
1331 string getGenre(bool isFiction, int GenreNumber) {
1332     //This method is to return Book Genre based on the given integer
1333     string Genre;
1334     if(isFiction == 1) { //If it is fiction
1335         switch(GenreNumber) {
1336             case GENRE_FICTION::FANTASY:
1337                 Genre = "FANTASY";
1338                 break;
1339             case GENRE_FICTION::SCIENCE:
1340                 Genre = "SCIENCE";
1341                 break;
1342             case GENRE_FICTION::HISTORICAL:
1343                 Genre = "HISTORICAL";
1344                 break;
1345             case GENRE_FICTION::REALISTIC:
1346                 Genre = "REALISTIC";
1347                 break;
1348             case GENRE_FICTION::FAN:
1349                 Genre = "FAN";
1350                 break;
1351         }
1352     } else { //If it is non-fiction
1353         switch(GenreNumber) {
1354             case GENRE_NONFICTION::NARRATIVE:
1355                 Genre = "NARRATIVE";
1356                 break;
1357             case GENRE_NONFICTION::BIOGRAPHY:
1358                 Genre = "BIOGRAPHY";
1359                 break;
1360             case GENRE_NONFICTION::PERIODICALS:
1361                 Genre = "PERIODICALS";
1362                 break;
1363             case GENRE_NONFICTION::SELF_HELP:
1364                 Genre = "SELF-HELP";
1365                 break;
1366             case GENRE_NONFICTION::REFERENCE:
1367                 Genre = "REFERENCE";
1368                 break;
1369         }
1370     }
1371     return Genre;
1372 };
```

Figure 135: getGenre method

The Figure 135 shown above is the getGenre method. Line 1333 is the variable declaration. Line 1335 to 1351 will be executed if it is Fiction while it will also receive GenreNumber and display the Genre in string based on the selection and break the case. Line 1353 to 1369 will be executed if it is Non-Fiction while it will also receive GenreNumber and display the Genre in string based on the selection and break the case. Line 1371 will be returning the Genre in string type.

getNewBook

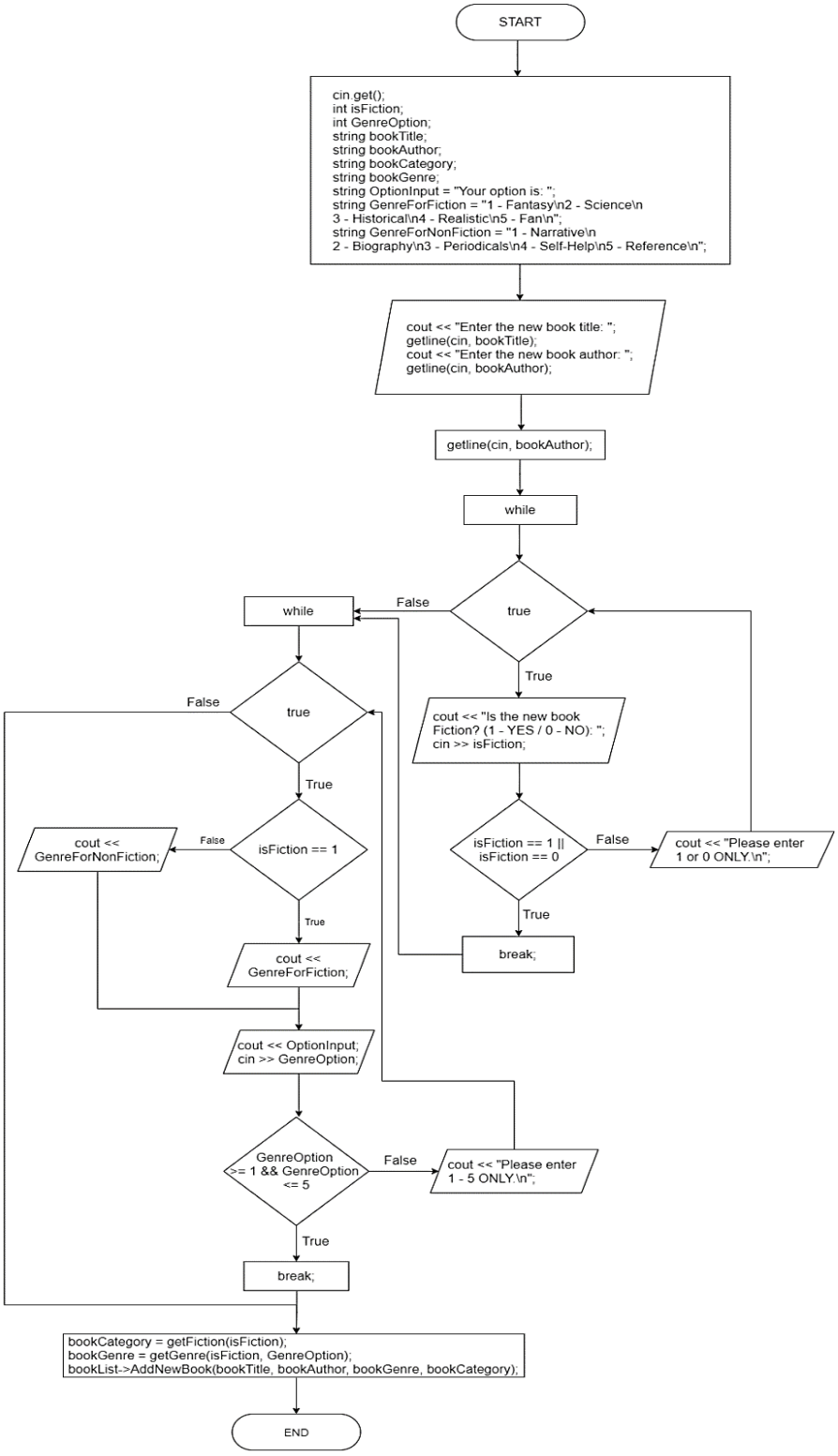


Figure 136: getNewBook method flowchart

LINK:

https://drive.google.com/file/d/14Zt1NwAwMif2Qwi7o8L6ujKDWJtUTPCZ/view?usp=sharing

Code Snippet

```
1374 void getNewbook(BookList* bookList) {
1375     cin.get();
1376     int isFiction;
1377     int GenreOption;
1378     string bookTitle;
1379     string bookAuthor;
1380     string bookCategory;
1381     string bookGenre;
1382     string OptionInput = "Your option is: ";
1383     string GenreForFiction = "1 - Fantasy\n2 - Science\n3 - Historical\n4 - Realistic\n5 - Fan\n";
1384     string GenreForNonFiction = "1 - Narrative\n2 - Biography\n3 - Periodicals\n4 - Self-Help\n5 - Reference\n";
1385 
1386     cout << "Enter the new book title: ";
1387     getline(cin, bookTitle);
1388     cout << "Enter the new book author: ";
1389     getline(cin, bookAuthor);
1390 
1391     while(true) { //Iterate if the isFiction is not either 1 or 0
1392         cout << "Is the new book Fiction? (1 - YES / 0 - NO): ";
1393         cin >> isFiction;
1394         if(isFiction == 1 || isFiction == 0) {
1395             break; //Break the iteration
1396         } else {
1397             cout << "Please enter 1 or 0 ONLY.\n";
1398         }
1399     }
1400 
1401     while(true) { //Iterate if the GenreOption is not between 1 and 5
1402         if(isFiction == 1) {
1403             cout << GenreForFiction;
1404         } else {
1405             cout << GenreForNonFiction;
1406         }
1407 
1408         cout << OptionInput;
1409         cin >> GenreOption;
1410 
1411         if(GenreOption >= 1 && GenreOption <= 5) {
1412             break; //Break the iteration
1413         } else {
1414             cout << "Please enter 1 - 5 ONLY.\n";
1415         }
1416     }
1417 
1418     bookCategory = getFiction(isFiction); //Get the book category by passing integer value
1419     bookGenre = getGenre(isFiction, GenreOption); //Get the book genre by passing integer value
1420 
1421     bookList -> AddNewBook(bookTitle, bookAuthor, bookGenre, bookCategory); //Call method to add new book
1422 
1423 }
```

Figure 137: getNewBook method

The Figure 137 shown above is the getNewBook method. Line 1375 to 1384 is the variable declaration. Line 1386 to 1389 is to allow the users to enter value and get the input value. Line 1391 to 1399 will be iterated until the user enters either 0 or 1 for the isFiction option, the iteration break happens in Line 1394 and 1395. Line 1401 to 1416 will be iterated if the Genre Option is not between 1 and 5, the iteration break happens in Line 1411 and 1412. Line 1418 and 1419 are to get the book category and book genre by passing the integer value entered. Line 1421 is to call the method to add new book.

getBook

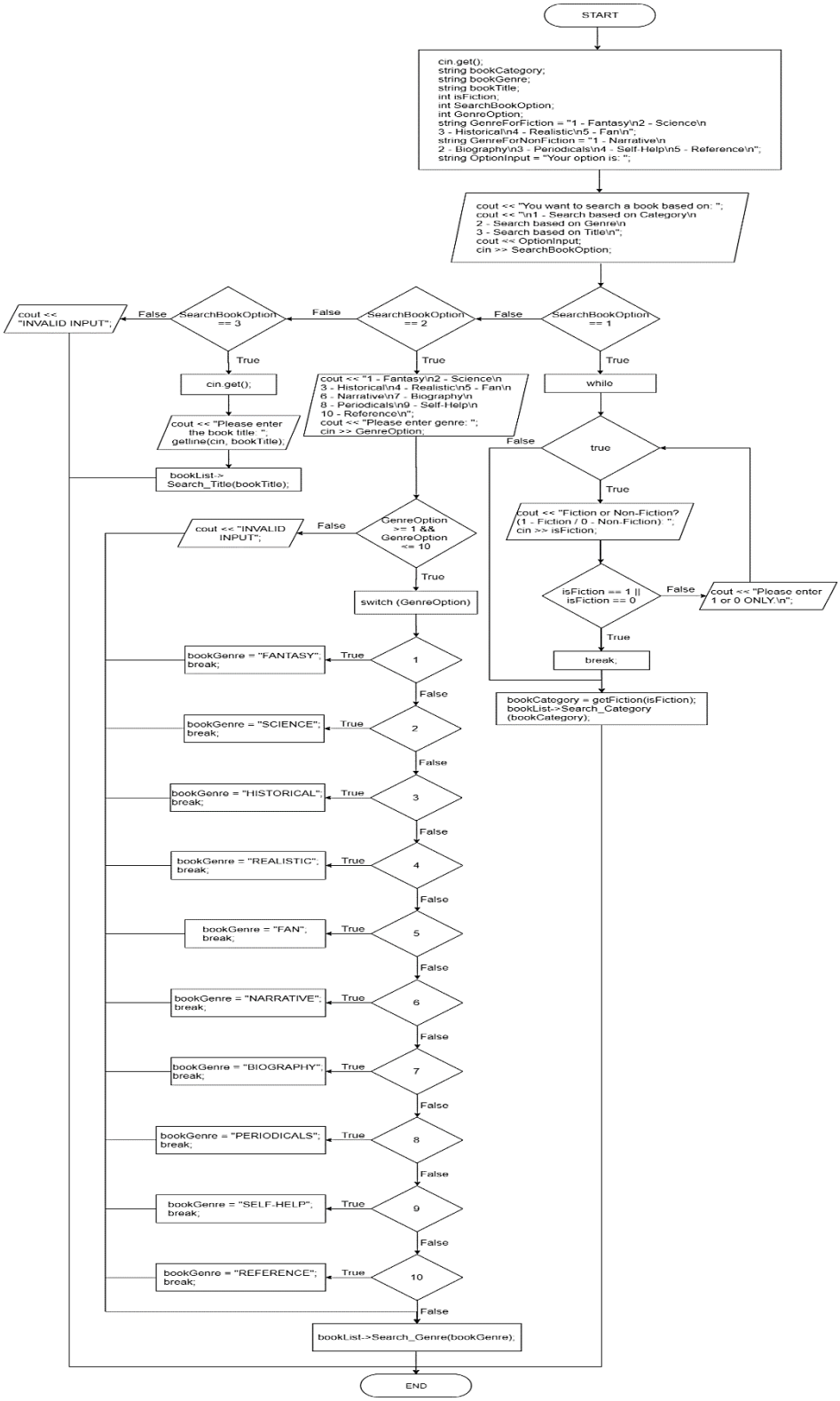


Figure 138: getBook method flowchart

LINK:

https://drive.google.com/file/d/1TETwIOnbKj__8BYNxVxtCSxoosPNXt33/view?usp=sharing

Code Snippet

```
1425 void getBook(BookList* bookList) {
1426     cin.get();
1427     string bookCategory;
1428     string bookGenre;
1429     string bookTitle;
1430     int isFiction;
1431     int SearchBookOption;
1432     int GenreOption;
1433 
1434     string GenreForFiction = "1 - Fantasy\n2 - Science\n3 - Historical\n4 - Realistic\n5 - Fan\n";
1435     string GenreForNonFiction = "1 - Narrative\n2 - Biography\n3 - Periodicals\n4 - Self-Help\n5 - Reference\n";
1436     string OptionInput = "Your option is: ";
1437 
1438     cout << "You want to search a book based on: ";
1439     cout << "\n1 - Search based on Category\n2 - Search based on Genre\n3 - Search based on Title\n";
1440     cout << OptionInput;
1441     cin >> SearchBookOption;
1442 
1443     if(SearchBookOption == 1) {
1444         while(true) { //Iterate if the isFiction is not either 1 or 0
1445             cout << "Fiction or Non-Fiction? (1 - Fiction / 0 - Non-Fiction): ";
1446             cin >> isFiction;
1447             if(isFiction == 1 || isFiction == 0) {
1448                 break; //Break the iteration
1449             } else {
1450                 cout << "Please enter 1 or 0 ONLY.\n";
1451             }
1452         }
1453         bookCategory = getFiction(isFiction); //Get the book category by passiing integer value
1454         bookList -> Search_Category(bookCategory);
1455     } else if (SearchBookOption == 2) {
1456         cout << "1 - Fantasy\n2 - Science\n3 - Historical\n4 - Realistic\n5 - Fan\n6 - Narrative\n7 - Biography\n8 - Periodicals\n9
            - Self-Help\n10 - Reference\n";
1457         cout << "Please enter genre: ";
1458         cin >> GenreOption;
1459 
1460         if(GenreOption >= 1 && GenreOption <= 10) {
1461             switch (GenreOption) {
1462                 case 1:
1463                     bookGenre = "FANTASY";
1464                     break;
1465                 case 2:
1466                     bookGenre = "SCIENCE";
1467                     break;
1468                 case 3:
1469                     bookGenre = "HISTORICAL";
1470                     break;
1471                 case 4:
1472                     bookGenre = "REALISTIC";
1473                     break;
1474                 case 5:
1475                     bookGenre = "FAN";
1476                     break;
1477                 case 6:
1478                     bookGenre = "NARRATIVE";
1479                     break;
1480                 case 7:
1481                     bookGenre = "BIOGRAPHY";
1482                     break;
1483                 case 8:
1484                     bookGenre = "PERIODICALS";
1485                     break;
1486                 case 9:
1487                     bookGenre = "SELF-HELP";
1488                     break;
1489                 case 10:
1490                     bookGenre = "REFERENCE";
1491                     break;
1492             }
1493         } else{
1494             cout << "INVALID INPUT";
1495         }
1496 
1497         bookList -> Search_Genre(bookGenre);
1498     } else if(SearchBookOption == 3) {
1499         cin.get();
1500         cout << "Please enter the book title: ";
1501         getline(cin, bookTitle);
1502         bookList -> Search_Title(bookTitle);
1503     } else {
1504         cout << "INVALID INPUT";
1505     }
1506 };
```

Figure 139: getBook method

The Figure 139 shown above is the getBook method. Line 1426 to 1436 is the variable declaration. Line 1438 to 1441 is to allow the users to enter values and get the input values.

Line 1444 to 1454 will be executed if the search book option is “1” and Line 1444 to 1452 will start to execute the iteration if the isFiction value is either 1 or 0. It will break the iteration on Line 1447 and 1448 is the isFiction value is 1 or 0. Line 1453 will get the book category by passing the integer value enter and will call the method to search the book based on category on Line 1454. Line 1456 to 1497 will be executed if the search book option is “2”. Line 1456 to 1458 allows user to enter from value 1 to 10 for Genre Option and display the book genre in string based on the input selection and break the case. Line 1497 will call the method to search the book based on genre. Line 1499 to 1502 will be executed if the search book option is “3”. Line 1500 and 1501 allows users to enter the book title and Line 1502 will call the method to search the book based on book title.

getPrintBook

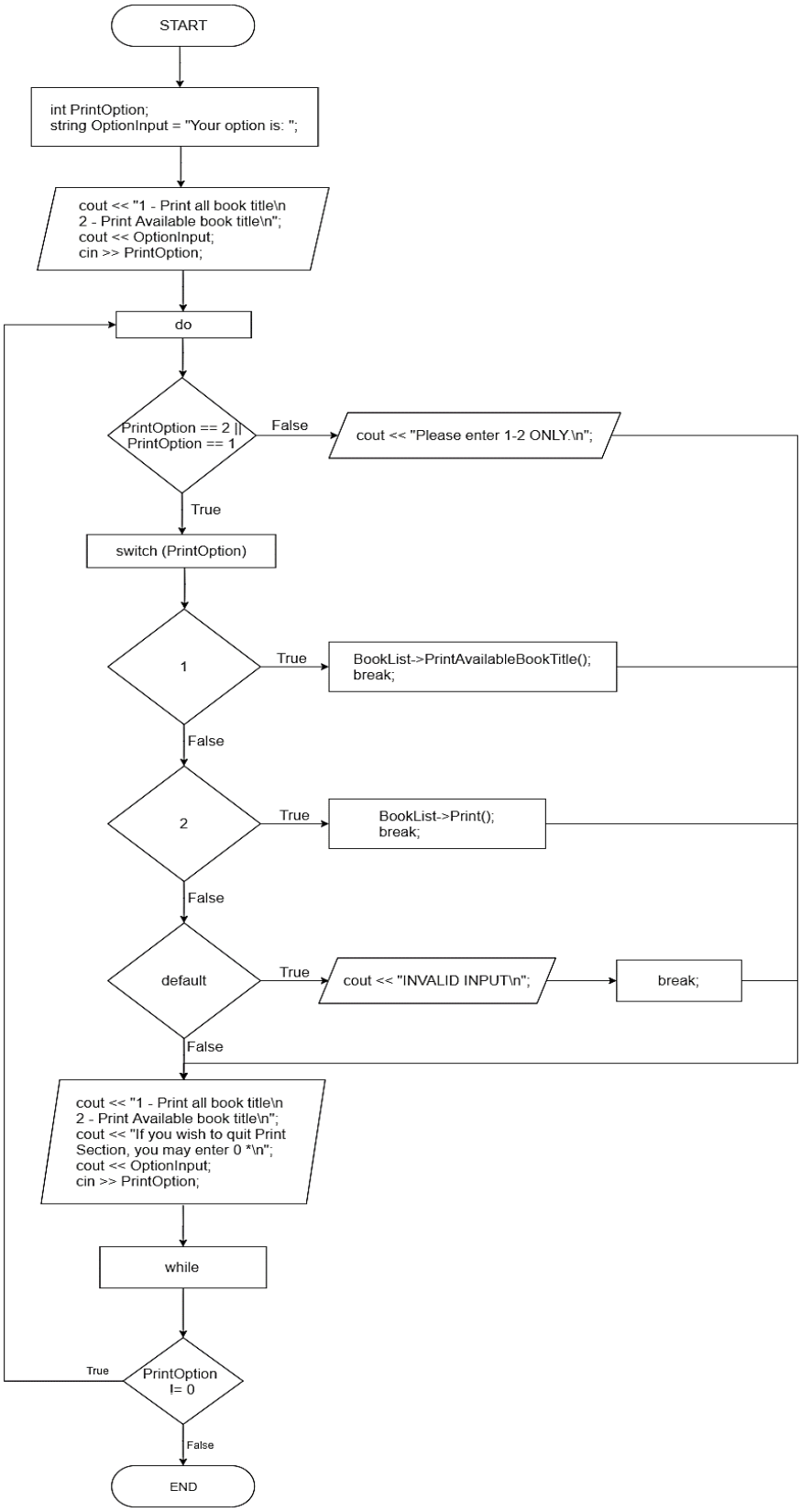


Figure 140: getPrintBook method flowchart

Code Snippet

```
1508 void getPrintBook(BookList* BookList) {
1509     int PrintOption;
1510     string OptionInput = "Your option is: ";
1511     cout << "1 - Print all book title\n2 - Print Available book title\n";
1512     cout << OptionInput;
1513     cin >> PrintOption;
1514     do { //Iterate if the PrintOption is not 0
1515         if (PrintOption == 2 || PrintOption == 1) {
1516             switch (PrintOption) {
1517                 case 1: {
1518                     BookList->PrintAvailableBookTitle();
1519                     break;
1520                 }
1521                 case 2: {
1522                     BookList->Print();
1523                     break;
1524                 }
1525                 default: {
1526                     cout << "INVALID INPUT\n";
1527                     break;
1528                 }
1529             }
1530         } else {
1531             cout << "Please enter 1-2 ONLY.\n";
1532         }
1533         cout << "1 - Print all book title\n2 - Print Available book title\n";
1534         cout << "If you wish to quit Print Section, you may enter 0 *\n";
1535         cout << OptionInput;
1536         cin >> PrintOption;
1537     } while (PrintOption != 0);
1538 };
```

Figure 141: getPrintBook method

The Figure 141 shown above is the getPrintBook method. Line 1509 to 1510 is the variable declaration. Line 1511 to 1513 is to allow the users to enter values and the get input values. Line 1516 to 1536 will be executed if the print option is either 1 or 0. Line 1517 to 1520 will be executed if the input value is 1 and the PrintAvailableBookTitle() method will be called and break the case. Line 1521 to 1524 will be executed if the input value is 2 and the Print() method will be called and break the case. Line 1537 will run the iteration if the print option is not equal to 0.

CheckIfISBNAvailable

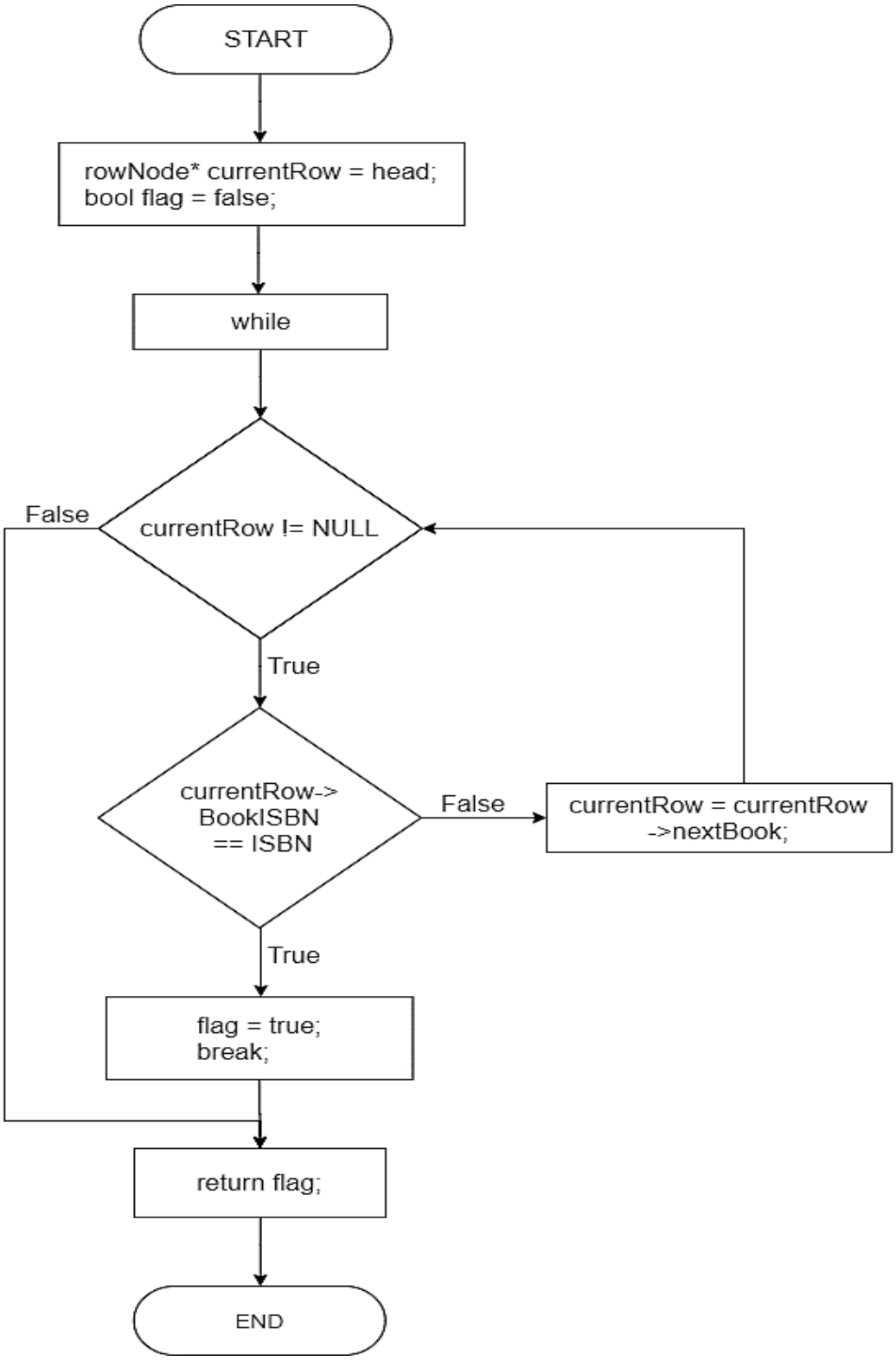


Figure 142: CheckIfISBNAvailable method flowchart

Code Snippet

```
1540 bool checkIfISBNAvailable(int ISBN) {
1541     rowNode* currentRow = head; //Copy value of head into currentRow
1542     bool flag = false;
1543     while(currentRow != NULL) { //Iterate if the currentRow is not NULL
1544         if(currentRow -> BookISBN == ISBN) { //Check if the given ISBN is matched with the book ISBN of book list
1545             flag = true; //Set flag to be true
1546             break; //Break the iteration
1547         }
1548         currentRow = currentRow -> nextBook; //Move to the next book
1549     }
1550     return flag; //Return true if the book ISBN is found, otherwise return false
1551 };
```

Figure 143: checkIfISBNAvailable

The Figure 143 shown above is the checkIfISBNAvailabe method. Line 1541 to 1542 is the variable declaration. Line 1544 to 1548 will check if the given ISBN is matched with the book ISBN of book list while iterating of the currentRow is not NULL. Line 1545 and 1546 will set the flag to true and break the iteration if the given ISBN is matched. Line 1548 will move to the next book. Line 1550 will return true if the book ISBN is found or else it will return false.

getUpdateBook

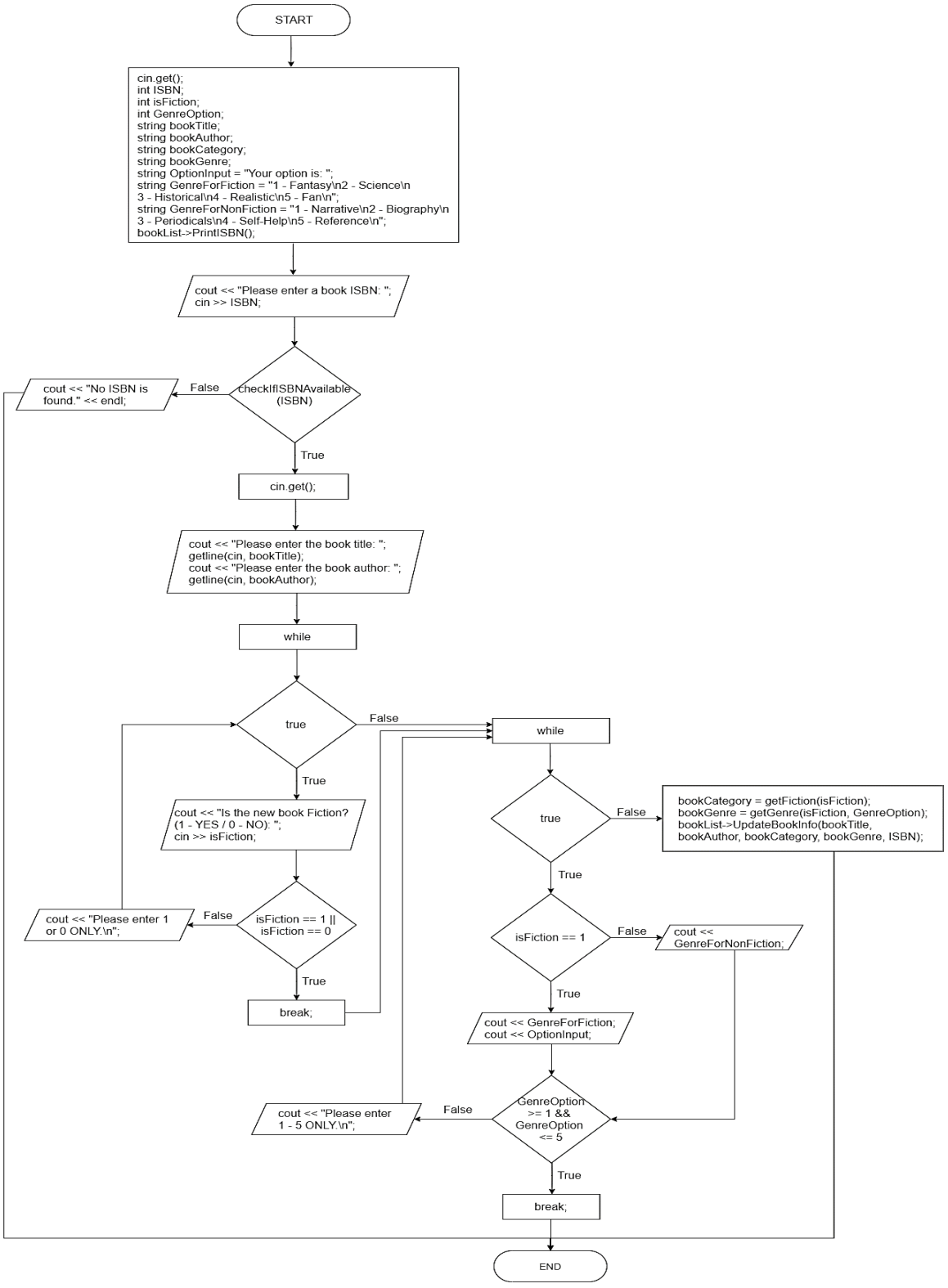


Figure 144: getUpdateBook method flowchart

LINK:

https://drive.google.com/file/d/1K6zOaJHuoEePg2ppiFPLeF8WnWJhCKuU/view?usp=sharing

Code Snippet

```
1553 void getUpdateABook(BookList* bookList) {
1554     cin.get();
1555     int ISBN;
1556     int isFiction;
1557     int GenreOption;
1558     string bookTitle;
1559     string bookAuthor;
1560     string bookCategory;
1561     string bookGenre;
1562     string OptionInput = "Your option is: ";
1563     string GenreForFiction = "1 - Fantasy\n2 - Science\n3 - Historical\n4 - Realistic\n5 - Fan\n";
1564     string GenreForNonFiction = "1 - Narrative\n2 - Biography\n3 - Periodicals\n4 - Self-Help\n5 - Reference\n";
1565 
1566     bookList -> PrintISBN();
1567     cout << "Please enter a book ISBN: ";
1568     cin >> ISBN;
1569 
1570     if(checkIfISBNAvailable(ISBN)) { //Check the method to check if the book ISBN is existed
1571         cin.get();
1572         cout << "Please enter the book title: ";
1573         getline(cin, bookTitle);
1574         cout << "Please enter the book author: ";
1575         getline(cin, bookAuthor);
1576 
1577         while(true) { //Iterate if the isFiction is not either 1 or 0
1578             cout << "Is the new book Fiction? (1 - YES / 0 - NO): ";
1579             cin >> isFiction;
1580             if(isFiction == 1 || isFiction == 0) {
1581                 break; //Break the iteratioin
1582             } else {
1583                 cout << "Please enter 1 or 0 ONLY.\n";
1584             }
1585         }
1586 
1587         while(true) { //Iterate if the genre option is not between 1 and 5
1588             if(isFiction == 1) {
1589                 cout << GenreForFiction;
1590             } else {
1591                 cout << GenreForNonFiction;
1592             }
1593 
1594             cout << OptionInput;
1595             cin >> GenreOption;
1596 
1597             if(GenreOption >= 1 && GenreOption <= 5) {
1598                 break; //Break the iterationi
1599             } else {
1600                 cout << "Please enter 1 - 5 ONLY.\n";
1601             }
1602         }
1603 
1604         bookCategory = getFiction(isFiction); //Get the book category by passing integer value
1605         bookGenre = getGenre(isFiction, GenreOption); //Get the book genre by passing integer value
1606 
1607         bookList -> UpdateBookInfo(bookTitle, bookAuthor, bookCategory, bookGenre, ISBN); //Call method to update the book detals
1608     } else {
1609         cout << "No ISBN is found." << endl;
1610     }
1611 };
```

Figure 145: getUpdateABook method

The Figure 145 shown above is the getUpdateBook method. Line 1554 to 1566 is the variable declaration. Line 1567 to 1568 is to allow the users to enter value and get the input value. Line 1571 to 1585 will be executed to check if the book ISBN is existed. Line 1571 to 1575 is to allow the users to enter value and get the input value. Line 1577 to 1584 will iterate if the isFiction is not either 1 or 0. Line 1580 will check if the isFiction is 1 or 0, if it does, it will break the iteration. Line 1587 to 1602 will iterate if the genre option is not between 1 and 5. Line 597 will check if the GenreOption is between 1 and 5, if it does, it will break the iteration. Line 1604 and 1605 are to get the book category and book genre by passing the integer value entered. Line 1607 is to call the method to update book details.

DeleteABook

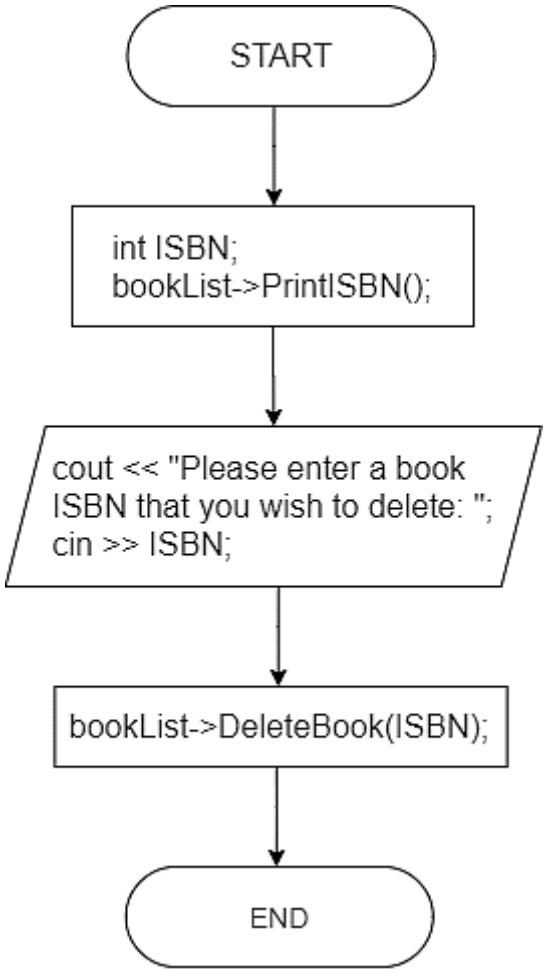


Figure 146: DeleteABook method flowchart

Code Snippet

```
1613 void DeleteABook(BookList* bookList) {
1614     int ISBN;
1615     bookList -> PrintISBN();
1616     cout << "Please enter a book ISBN that you wish to delete: ";
1617     cin >> ISBN;
1618 
1619     bookList -> DeleteBook(ISBN); //Call deletebook method to delete book by passing ISBN
1620 }
```

Figure 147: DeleleABook method

The Figure 147 shown above is the DeleteABook method. Line 1614 is the variable declaration. Line 1615 is to call the PrintISBN() method. Line 1616 to 1617 is to allow the users to enter the value and get the input value. Line 1619 is to call the method to delete book.

getNewPatron

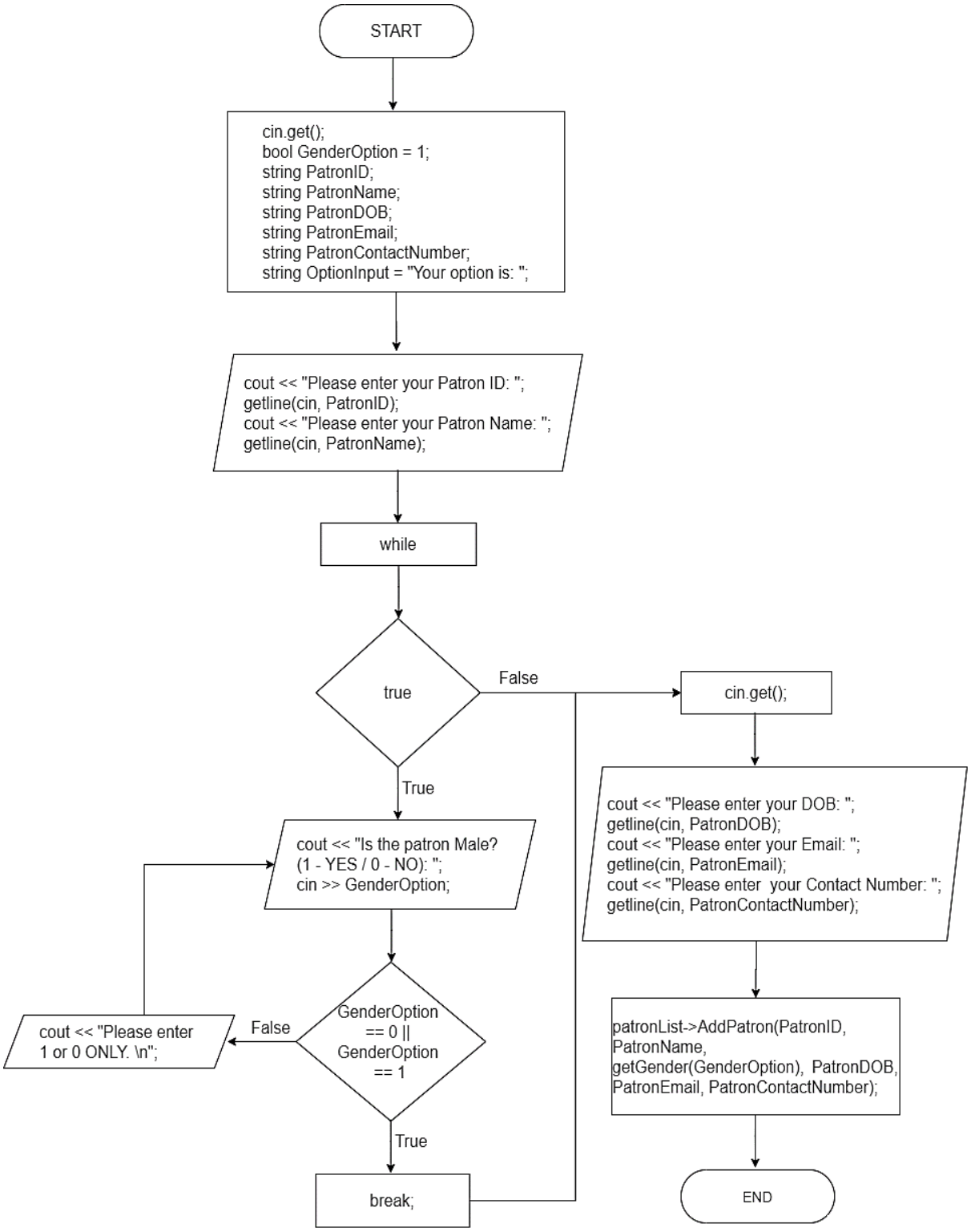


Figure 148: getNewPatron method flowchart

Code Snippet

```
1629 void getNewPatron(PatronList* patronList) {
1630     cin.get();
1631     bool GenderOption = 1;
1632     string PatronID;
1633     string PatronName;
1634     string PatronDOB;
1635     string PatronEmail;
1636     string PatronContactNumber;
1637     string OptionInput = "Your option is: ";
1638 
1639     cout << "Please enter your Patron ID: ";
1640     getline(cin, PatronID);
1641     cout << "Please enter your Patron Name: ";
1642     getline(cin, PatronName);
1643 
1644     while(true) { //Iterate if the gender option is not ether 1 or 0
1645         cout << "Is the patron Male? (1 - YES / 0 - NO): ";
1646         cin >> GenderOption;
1647         if(GenderOption == 0 || GenderOption == 1) {
1648             break; //Break the iteration
1649         } else {
1650             cout << "Please enter 1 or 0 ONLY. \n";
1651         }
1652     }
1653     cin.get();
1654     cout << "Please enter your DOB: ";
1655     getline(cin, PatronDOB);
1656     cout << "Please enter your Email: ";
1657     getline(cin, PatronEmail);
1658     cout << "Please enter your Contact Number: ";
1659     getline(cin, PatronContactNumber);
1660 
1661     //Call method to add new patron by passing arguments
1662     patronList -> AddPatron(PatronID, PatronName, getGender(GenderOption), PatronDOB, PatronEmail, PatronContactNumber);
1663 
1664 };
```

Figure 149: getNewPatron method

The Figure 149 shown above is the getNewPatron method. Line 1630 to 1637 is the variable declaration. Line 1639 to 1642 is to allow the users to enter the value and get the input value. Line 1644 to 1652 will iterate if the gender option is not either 1 or 0. Line 1647 will check is the GenreOption is 1 or 0, if it does, it will break the iteration. Line 1653 to 1659 will allow the users to enter the value and get the input value. Line 1662 will call the method to add new patron by passing the arguments.

getPatron

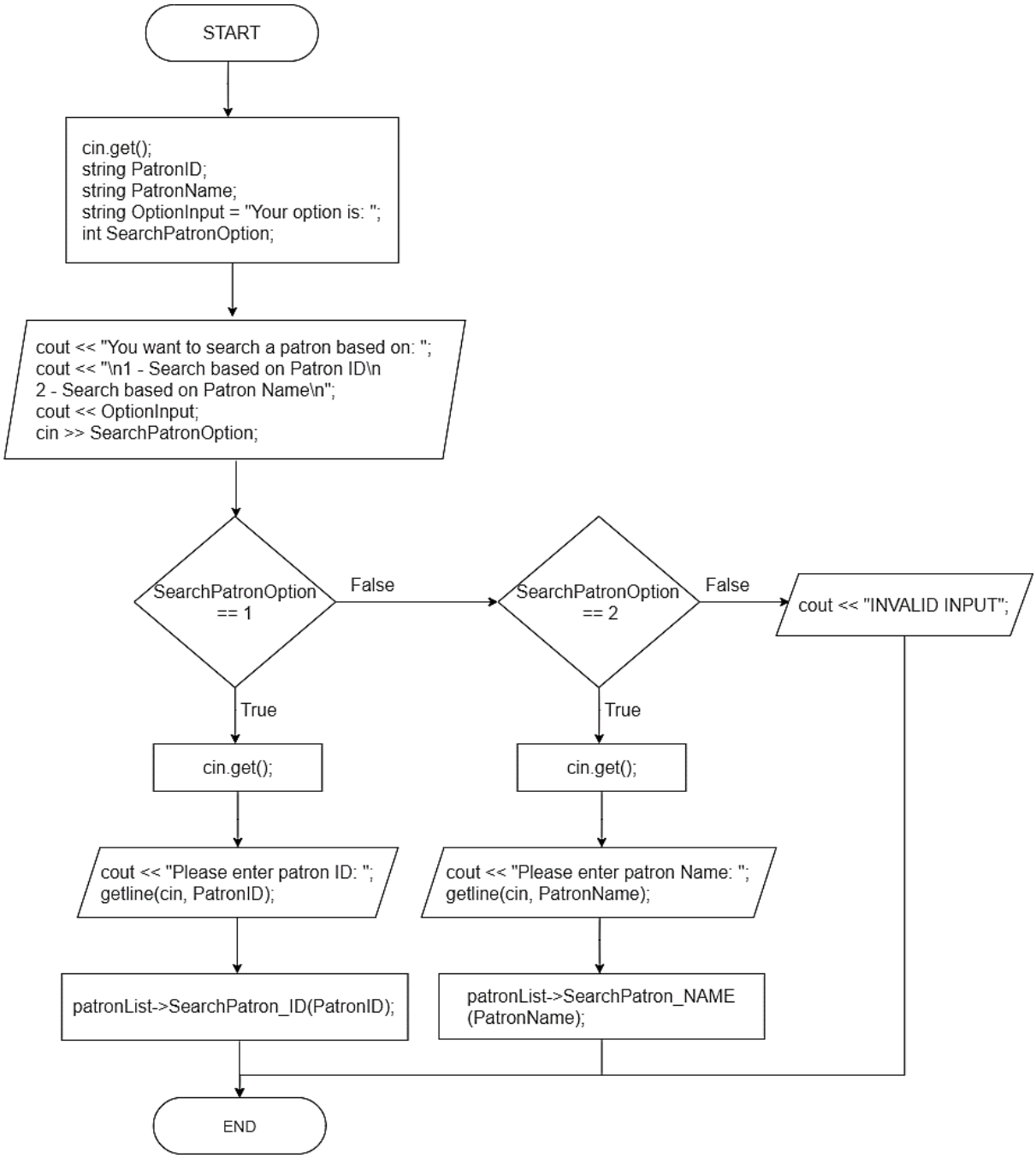


Figure 150: getPatron method flowchart

Code Snippet

```
1660 void getPatron(PatronList* patronList) {
1661     cin.get();
1662     string PatronID;
1663     string PatronName;
1664     string OptionInput = "Your option is: ";
1665     int SearchPatronOption;
1666 
1667     cout << "You want to search a patron based on: ";
1668     cout << "\n1 - Search based on Patron ID\n2 - Search based on Patron Name\n";
1669     cout << OptionInput;
1670     cin >> SearchPatronOption;
1671 
1672     if(SearchPatronOption == 1) {
1673         cin.get();
1674         cout << "Please enter patron ID: ";
1675         getline(cin, PatronID);
1676         patronList -> SearchPatron_ID(PatronID);
1677     } else if (SearchPatronOption == 2) {
1678         cin.get();
1679         cout << "Please enter patron Name: ";
1680         getline(cin, PatronName);
1681         patronList -> SearchPatron_NAME(PatronName);
1682     } else {
1683         cout << "INVALID INPUT";
1684     }
1685 };
```

Figure 151: getPatron method

The Figure 151 shown above is the getPatron method. Line 1661 to 1665 is the variable declaration. Line 1667 to 1670 is allow the users to enter value and get the input value. Line 1672 is to check if the SearchPatronOption is equal to “1”, if it does, Line 1673 to 1675 will allow users to enter value and get the input value. Line 1676 is to call the method to search the patron based on patron ID. Line 1677 is to check if the SearchPatronOption is equal to “2”, if it does, Line 1678 to 1680 will allow users to enter value and get the input value. Line 1681 is to call the method to search the patron based on patron Name.

getPrintPatron

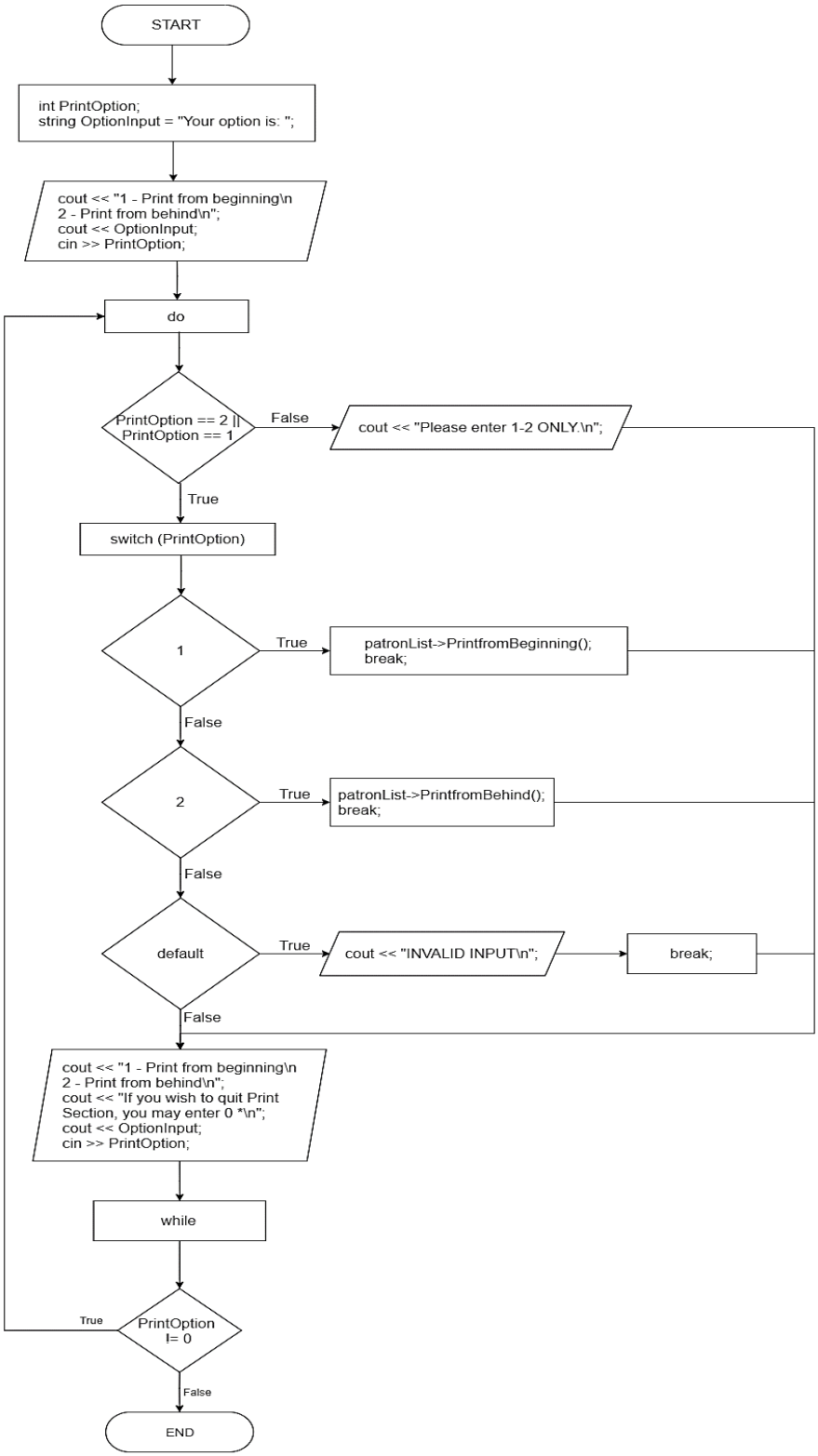


Figure 152: getPrintPatron method flowchart

Code Snippet

```
1687 void getPrintPatron(PatronList* patronList) {
1688     int PrintOption;
1689     string OptionInput = "Your option is: ";
1690     cout << "1 - Print from beginning\n2 - Print from behind\n";
1691     cout << OptionInput;
1692     cin >> PrintOption;
1693 
1694     do { //Iterate if the print option is not 0
1695         if(PrintOption == 2 || PrintOption == 1) {
1696             switch(PrintOption) {
1697                 case 1: {
1698                     patronList -> PrintfromBeginning();
1699                     break;
1700                 }
1701                 case 2: {
1702                     patronList -> PrintfromBehind();
1703                     break;
1704                 }
1705                 default: {
1706                     cout << "INVALID INPUT\n";
1707                     break;
1708                 }
1709             }
1710         } else {
1711             cout << "Please enter 1-2 ONLY.\n";
1712         }
1713 
1714         cout << "1 - Print from beginning\n2 - Print from behind\n";
1715         cout << "If you wish to quit Print Section, you may enter 0 *\n";
1716         cout << OptionInput;
1717         cin >> PrintOption;
1718     } while (PrintOption != 0);
1719 };
```

Figure 153: getPrintPatron method

The Figure 153 shown above is the getPrintPatron method. Line 1688 to 1689 is the variable declaration. Line 1690 to 1692 is to allow the users to enter values and get the input values. Line 1694 to 1718 will iterate if the print option is not 0 and Line 1695 to 1712 will be executed if the print option is either 1 or 2. Line 1697 to 1700 will be executed if the input value I 1 and the PrintFromBeginning() method will be called and break the case. Line 1701 to 1704 will be executed if the input value is 2 and the PrintFromBehind() method will be called and break the case. Line 1718 will run the iteration if the print option is not equal to 0.

makeReturnBook

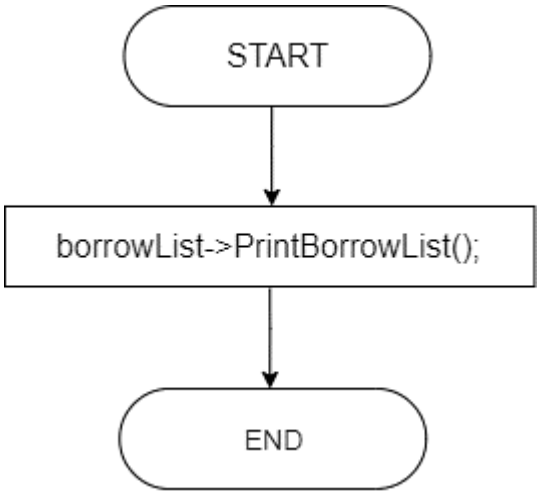


Figure 154: makeReturnBook method flowchart

Code Snippet

```
1721 void makeReturnBook(BorrowList* borrowList) {
1722     borrowList->PrintBorrowList(); //Call method to print borrow list
1723 }
```

Figure 155: makeReturnBook method

The Figure 155 shown above is the makeReturnBook method. Line 1722 is to call the method to print borrow list.

checkPatronID

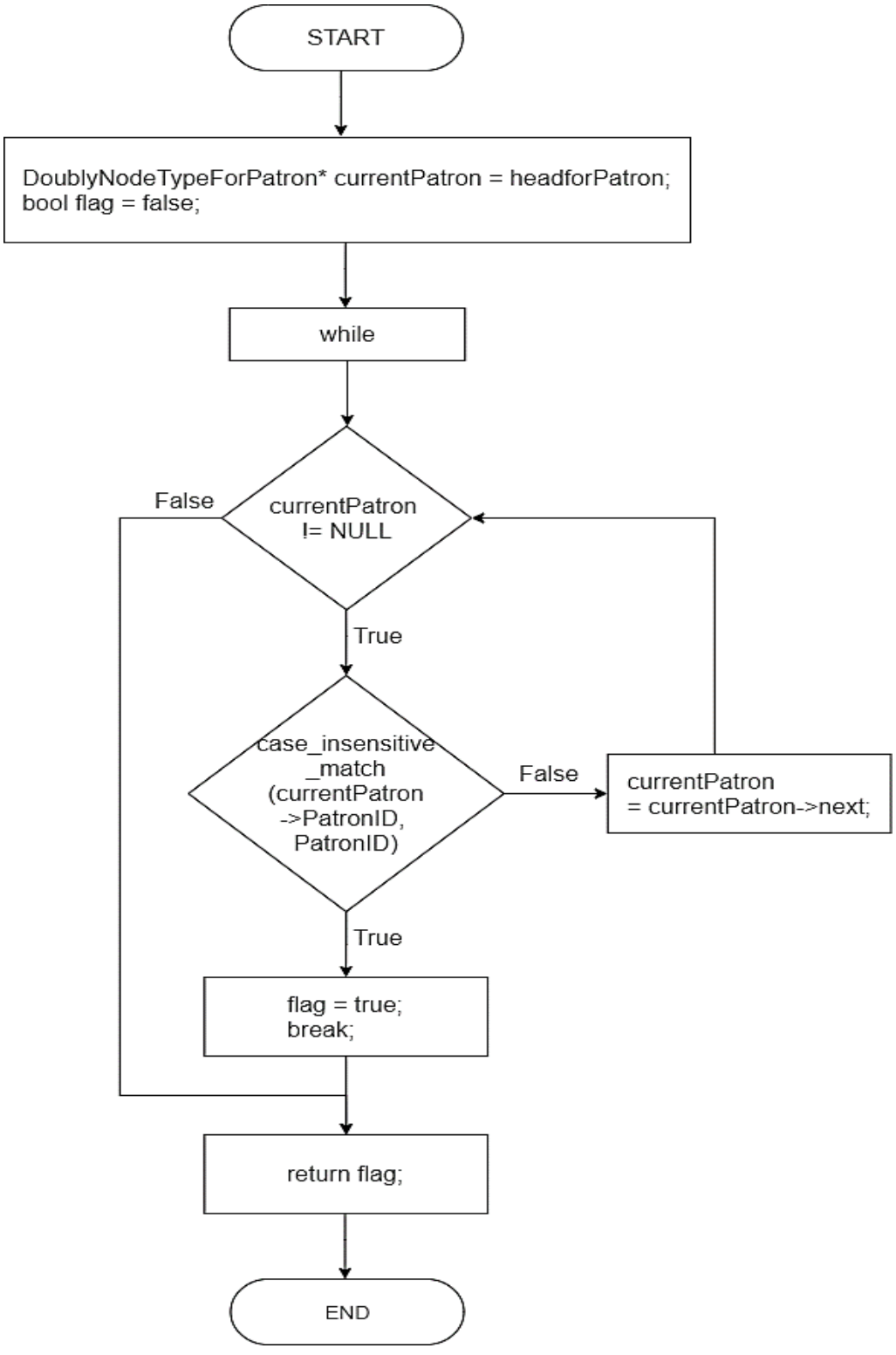


Figure 156: checkPatronID method flowchart

Code Snippet

```
1725 bool CheckPatronID(string PatronID) {
1726     //Copy value of headforPatron into currentPatron
1727     DoublyNodeTypeForPatron* currentPatron = headforPatron;
1728     bool flag = false;
1729     while(currentPatron != NULL) { //Iterate if the currentPatron is not NULL
1730         if(case_insensitive_match(currentPatron -> PatronID, PatronID)) {
1731             flag = true; //Set the flag to be true
1732             break; //Break the iteration
1733         }
1734         currentPatron = currentPatron -> next; //Move to the next patron
1735     }
1736     return flag;
1737 }
```

Figure 157: CheckPatronID method

The Figure 157 shown above is the CheckPatronID method. Line 1727 to 1728 is the variable declaration. Line 1729 to 1735 is to check if the given patron ID is exactly the same as the data stored in the patron linked list by converting into lower cases. Set the flag to be true to indicate the same patron ID is detected and then break the iteration. If the Line 1730 does not meet, then the flag is false as default. Line 1734 is to move to the next patron. Line 1736 is to return the flag value as boolean.

getUpdatePatron

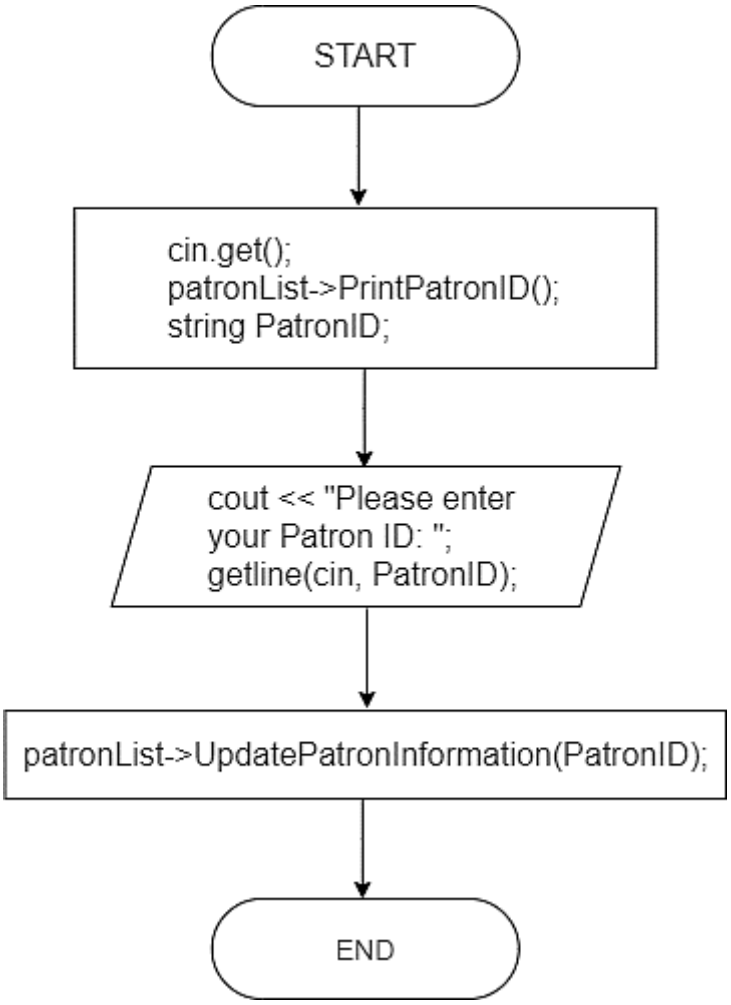


Figure 158: getUpdatePatron method flowchart

Code Snippet

```
1739 void getUpdatePatron(PatronList* patronList) {
1740     cin.get();
1741     patronList -> PrintPatronID();
1742     string PatronID;
1743     cout << "Please enter your Patron ID: ";
1744     getline(cin, PatronID);
1745     patronList -> UpdatePatronInformation(PatronID);
1746 };
```

Figure 159: getUpdatePatron method

The Figure 159 shown above is the getUpatePatron method. Line 1740 and 1742 is the variable declaration. Line 1741 is to call the method to print the patron ID. Line 1743 and 1744 is to allow the users to enter the value and get the input value. Line 1745 is to call the method to update patron information by passing argument.

getISBNtoBookTitle

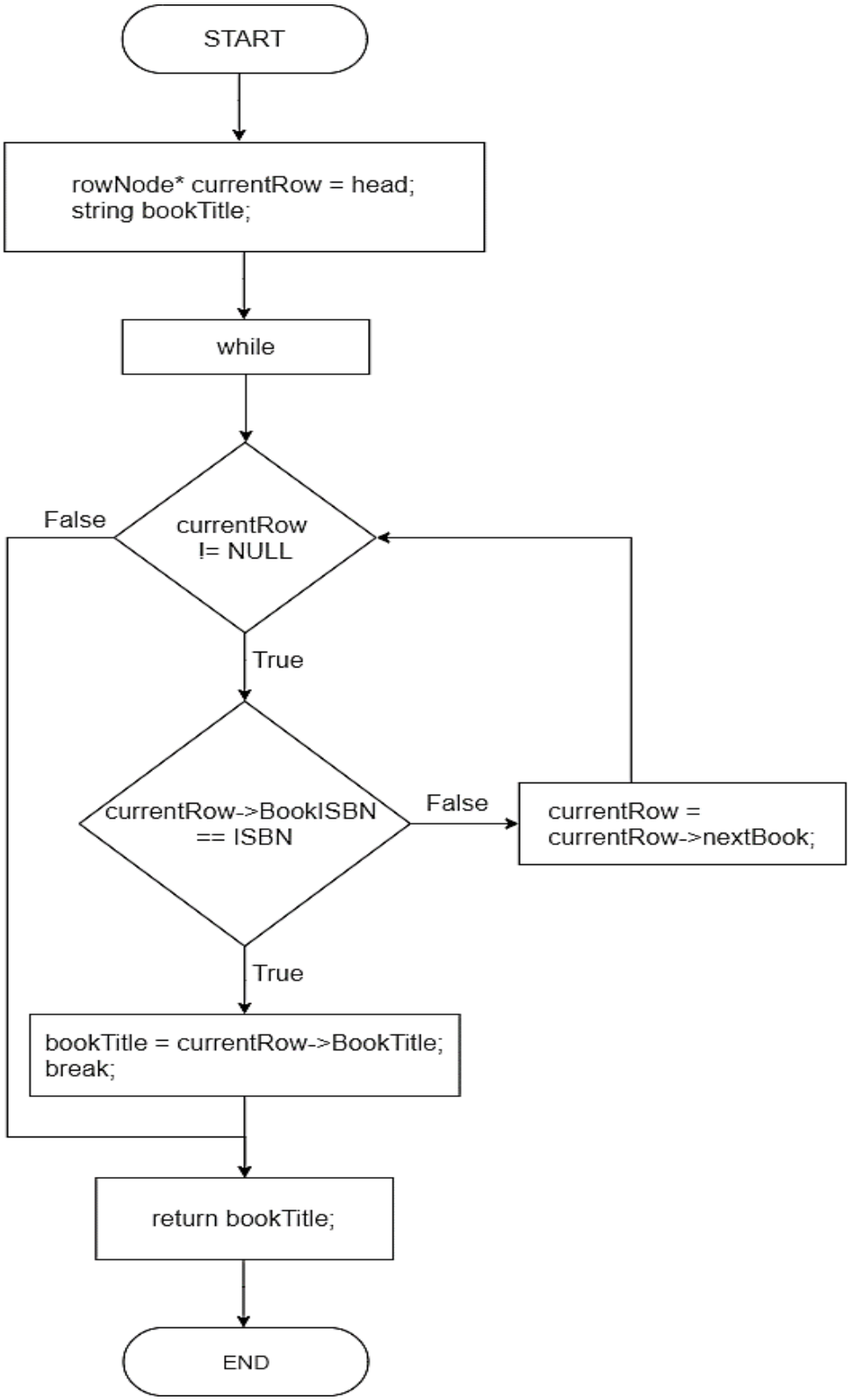


Figure 160: getISBNtoBookTitle method flowchart

Code Snippet

```
1749 string getISBNtoBookTitle(int ISBN) {
1750     rowNode* currentRow = head; //Copy value of head into currentRow
1751     string bookTitle;
1752     while (currentRow != NULL) { //Iterate if the currentRow is not NULL
1753         if (currentRow->BookISBN == ISBN) { //Check if the given Book ISBN is matched with the book ISBN of book list
1754             bookTitle = currentRow->BookTitle; //Assign book title to the BookTitle
1755             break; //Break the iteration
1756         }
1757         currentRow = currentRow->nextBook; //Move to the next book
1758     }
1759     return bookTitle; //Return the book title string
1760 };
```

Figure 161: getISBNtoBookTitle method

The Figure 161 shown above is the getISBNtoBookTitle method. Line 1750 to 1751 is the variable declaration. Line 1752 to 1758 is to check if the given book ISBN is matched with the book ISBN of the book list. Assign the book title to BookTitle if the given ISBN is matched with the book ISBN of the book list and break the iteration. Line 1757 is to move to next book. Line 1759 is to return the book title in string.

getNewBorrow

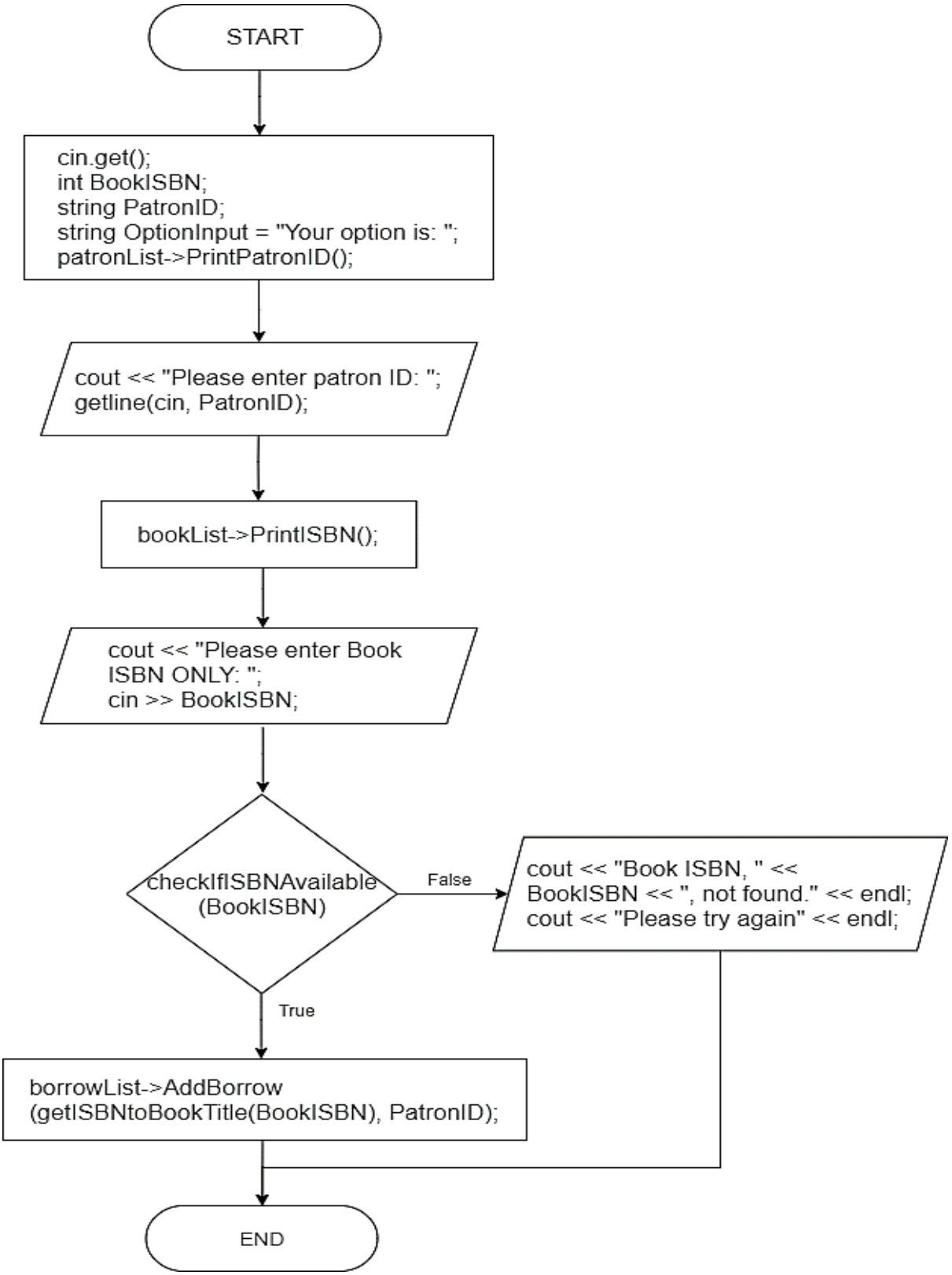


Figure 162: getNewBorrow method flowchart

Code Snippet

```
1762 void getNewBorrow(BorrowList* borrowList, PatronList* patronList, BookList* bookList) {
1763     cin.get();
1764     int BookISBN;
1765     string PatronID;
1766     string OptionInput = "Your option is: ";
1767 
1768     patronList->PrintPatronID();
1769     cout << "Please enter patron ID: ";
1770     getline(cin, PatronID);
1771 
1772     bookList->PrintISBN();
1773     cout << "Please enter Book ISBN ONLY: ";
1774     cin >> BookISBN;
1775 
1776     if (checkIfISBNAvailable(BookISBN)) { //Check if the ISBN is existed
1777         borrowList->AddBorrow(getISBNtoBookTitle(BookISBN), PatronID); //Add a new book borrow transaction
1778     }
1779     else {
1780         cout << "Book ISBN, " << BookISBN << ", not found." << endl;
1781         cout << "Please try again" << endl;
1782     }
1783 
1784 };
```

Figure 163: getNewBorrow method

The Figure 163 shown above is the getNewBorrow method. Line 1763 to 1766 is the variable declaration. Line 1769 to 1770 and 1773 to 1774 is allow the users to enter values and get the input values. Line 1768 is to call the method to print the patron ID. Line 1772 is to call the method to print the book ISBN. Line 1776 to 1782 is to check if the ISBN is existed, if it does, Line 1777 will call the method to add a new book borrow transaction by passing the arguments. Otherwise, it will notify the user that the book ISBN is not found.

Result

```
BORROW MENU:
1 - Add a new book borrow transaction
2 - Print
3 - Make Extension
4 - Return a book
If you wish to quit Borrow Section, you may enter 0 *
Your option is: 1


Patron Name                  Patron ID
HEN KIAN JUN                 HKJ001
LEE WAI YEN                  LWY002
LOW BOON KIAT                LBK003
NG ZHENG JUE                 NZJ004
LIM YI KANG                  LYK005
YONG MUN WEI                 YMW006
TEOH KHEN MENG               TKM007
LIM AH BENG                  TYJ008

Please enter patron ID: HKJ001


BOOK TITLE                        BOOK AUTHOR                     ISBN
HARRY POTTER                      J. K. ROWLING                   6808
MEIN KAMPF                        ADOLF HITLER                    5250
13 REASONS WHY                    JAY ASHER                       74
THE INVISIBLE MAN                 H. G. WELLS                     3659
THE HUNGER GAMES                  SUZANNE COLLINS                 8931
I HAVE A DREAM                    MARTIN LUTHER KING JR.          1273
CRIME AND PERIODICALS             NOVA EVERLY                     7545
A NEW EARTH                       ECKHART TOLLE                   879
THE HERO WITH A THOUSAND FACES    JOSEPH CAMPBELL                 7924

Please enter Book ISBN ONLY: 6888
Book ISBN, 6888, not found.
Please try again
```

Figure 164: Sample output - getNewBorrow method

The Figure 164 shown above is the sample output for AddBorrow method. The user will need to enter 1 to add a new borrow transaction. Then, user is required to base on the patron list shown in the picture to type the patron ID and based on the book list to type the book ISBN. Even though, if the patron ID typed is within the patron list however, if the book ISBN typed is not within the book list, the error message will be prompted to the user indicating that the book ISBN is not found within the book list.

getPrintBorrow

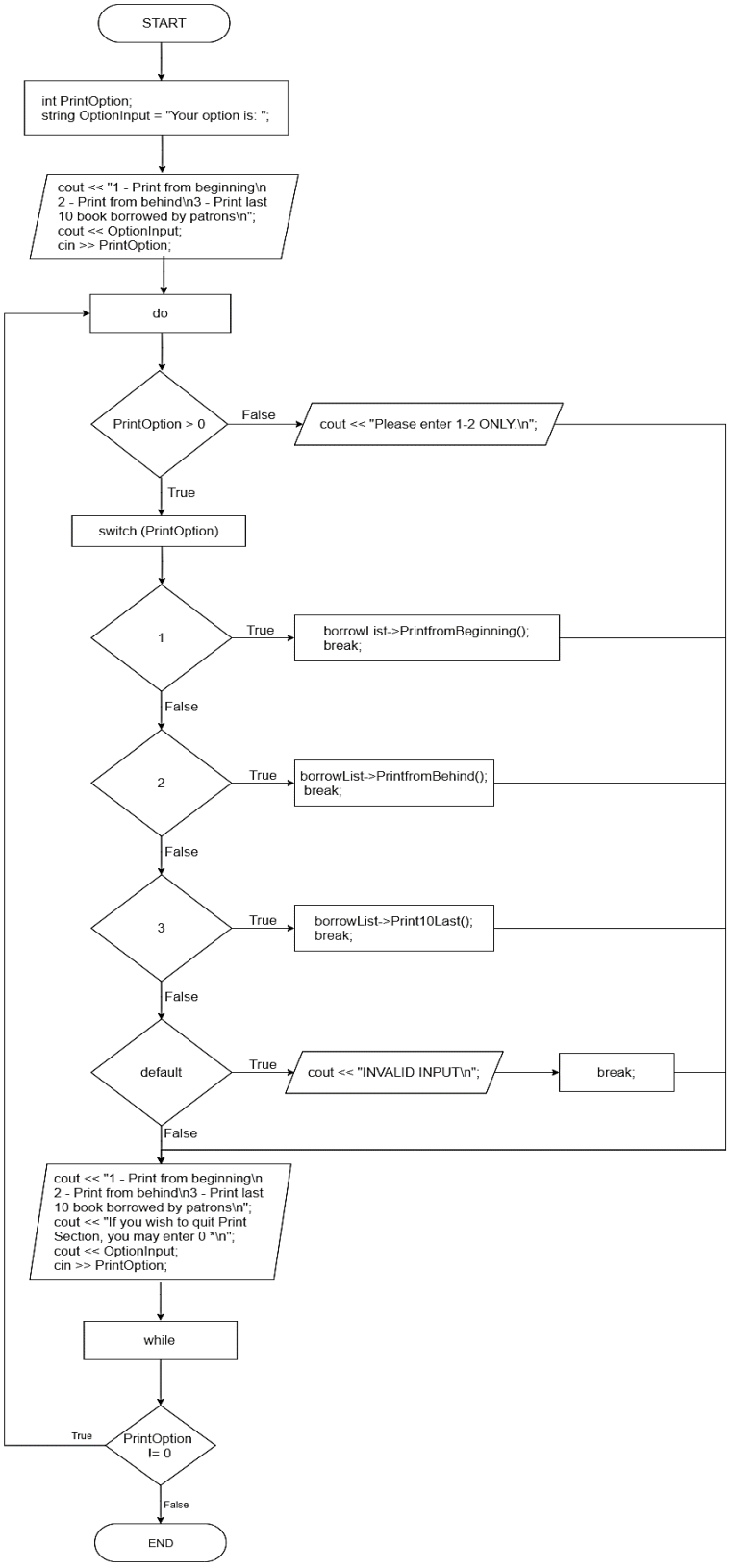


Figure 165: getPrintBorrow method flowchart

Code Snippet

```
1786 void getPrintBorrow(BorrowList* borrowList) {
1787     int PrintOption;
1788     string OptionInput = "Your option is: ";
1789     cout << "1 - Print from beginning\n2 - Print from behind\n3 - Print last 10 book borrowed by patrons\n";
1790     cout << OptionInput;
1791     cin >> PrintOption;
1792 
1793     do { //Iterate if the print option is not 0
1794         if(PrintOption > 0) {
1795             switch (PrintOption) {
1796                 case 1:
1797                     borrowList -> PrintfromBeginning();
1798                     break;
1799                 case 2:
1800                     borrowList -> PrintfromBehind();
1801                     break;
1802                 case 3:
1803                     borrowList -> Print10Last();
1804                     break;
1805                 default:
1806                     cout << "INVALID INPUT\n";
1807                     break;
1808             }
1809         } else {
1810             cout << "Please enter 1, 2, or 3 ONLY.\n";
1811         }
1812 
1813         cout << "1 - Print from beginning\n2 - Print from behind\n3 - Print last 10 book borrowed by patrons\n";
1814         cout << "If you wish to quit Print Section, you may enter 0 *\n";
1815         cout << OptionInput;
1816         cin >> PrintOption;
1817     } while (PrintOption != 0);
1818 };
```

Figure 166: getPrintBorrow method

The Figure 166 shown above is the getPrintBorrow method. Line 1787 to 1788 is the variable declaration. Line 1789 to 1791 is to allow the users to enter value and get the input values. Line 1793 to 1817 will iterate if the print option is not 0 however, within Line 1794 to 1811 will be executed if the print option is greater than 0. Line 1796 to 1798 will be executed if the input value is 1 and the PrintfromBeginning() method will be called and break the case. Line 1799 to 1801 will be executed if the input value is 2 and the PrintfromBehind() method will be called and break the case. Line 1802 to 1804 will be executed if the input value is 3 and the Print10last() method will be called and break the case. Line 1817 will run the iteration if the print option is not equal to 0.

getMakeUpdateExtension

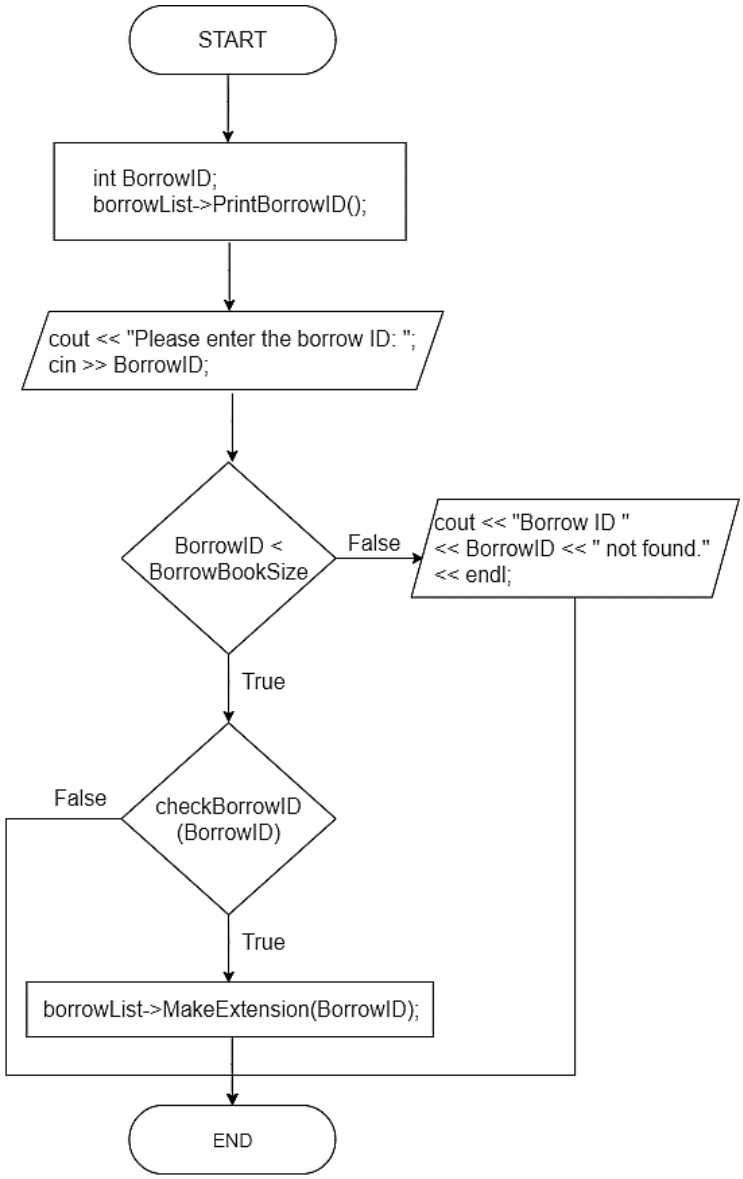


Figure 167: getMakeUpdateExtension method flowchart

Code Snippet

```
1827 void getMakeUpdateExtension(BorrowList* borrowList) {
1828     int BorrowID;
1829 
1830     borrowList -> PrintBorrowID();
1831     cout << "Please enter the borrow ID: ";
1832     cin >> BorrowID;
1833 
1834     if(BorrowID < BorrowBookSize) {
1835         if(checkBorrowID(BorrowID)) //Check if the given borrow ID is existed
1836             borrowList -> MakeExtension(BorrowID); //Call make extension method by passing borrow ID
1837     } else {
1838         cout << "Borrow ID " << BorrowID << " not found." << endl;
1839     }
1840 
1841 };
```

Figure 168: getMakeUpdateExtension method

The Figure 168 shown above is the getMakeUpdateExtension method. Line 1828 is the variable declaration. Line 1830 is to call the method to print the borrow ID. Line 1831 to 1832 is to allow the user to enter value and get the input value. Line 1834 is to check if the given borrow ID is lesser than the size for borrow record, the Line 1835 is to check if the given borrow ID is existed, if it does, then Line 1836 will be executed to call MakeExtension function by passing borrow ID. Otherwise, Line 1838 will be executed to print the given borrow ID not found.

Login

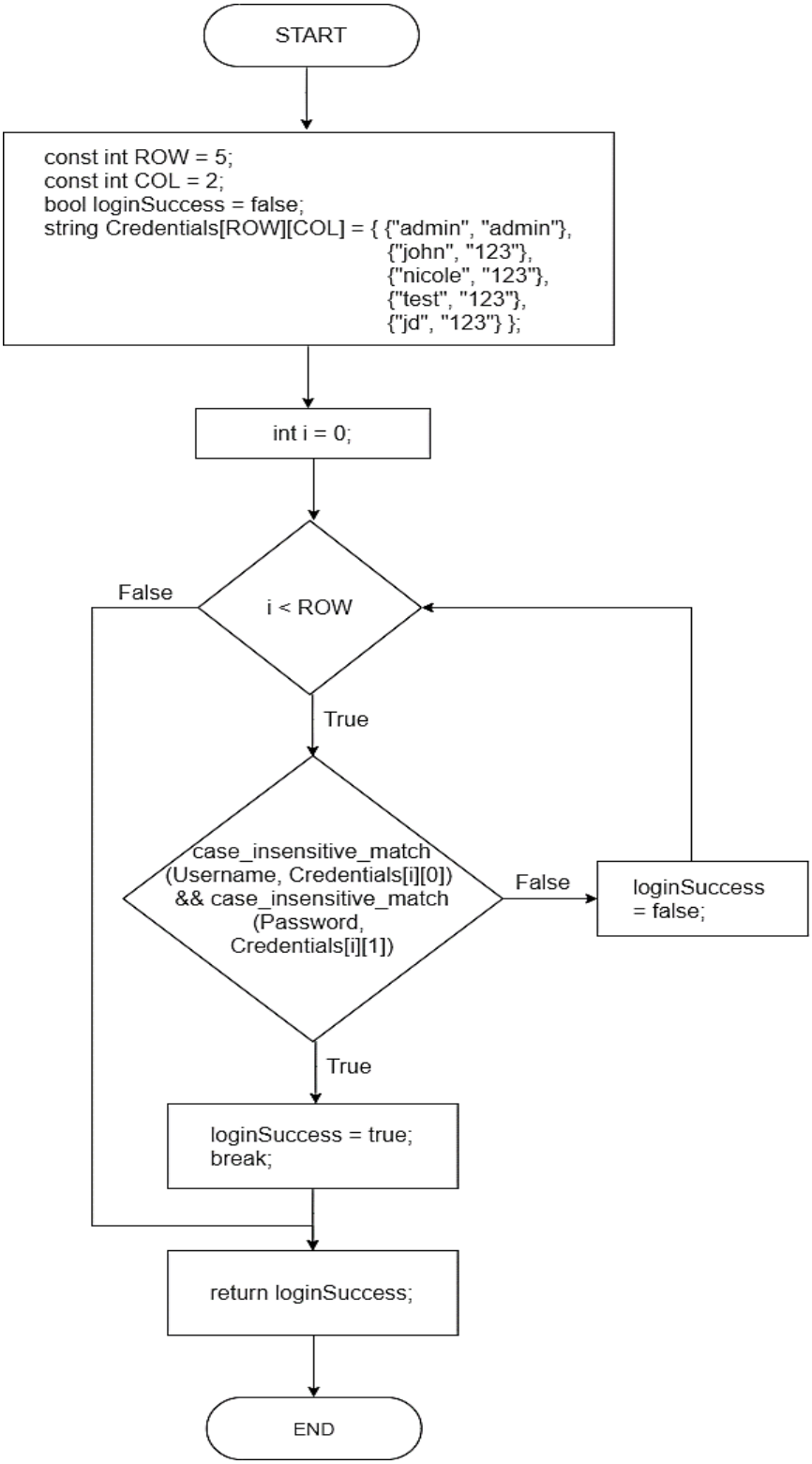


Figure 169: Login method flowchart

Code Snippet

```
1832 bool Login(string Username, string Password) {
1833     const int ROW = 5;
1834     const int COL = 2;
1835     bool loginSuccess = false;
1836                                     //username  //password
1837     //Create credentials database
1838     string Credentials[ROW][COL] = {{"admin", "admin"},
1839                                     {"john", "123"},
1840                                     {"nicole", "123"},
1841                                     {"test", "123"},
1842                                     {"jd", "123"}};
1843 
1844     for(int i = 0; i < ROW; i++){ //Iterate all the credentiials
1845         //Check if the given username and password are matched with the credentials database
1846         if(case_insensitive_match(Username, Credentials[i][0]) && case_insensitive_match(Password, Credentials[i][1])) {
1847             loginSuccess = true; //Set loginSuccess to be true
1848             break; //Break the iteration
1849         }
1850         else
1851             loginSuccess = false;
1852 
1853     }
1854 
1855     return loginSuccess;
1856 
1857 };
```

Figure 170: Login method

The Figure 170 shown above is the Login method. Line 1833 to 1842 is the variable declaration. Line 1844 to 1853 is using the for loop to check the credentials by each row. Line 1846 is to check if the given username and password are matched with the credentials database, if it does, set the loginSuccess to be true and break the iteration. Line 1855 is to return the loginSuccess value in boolean.

Main

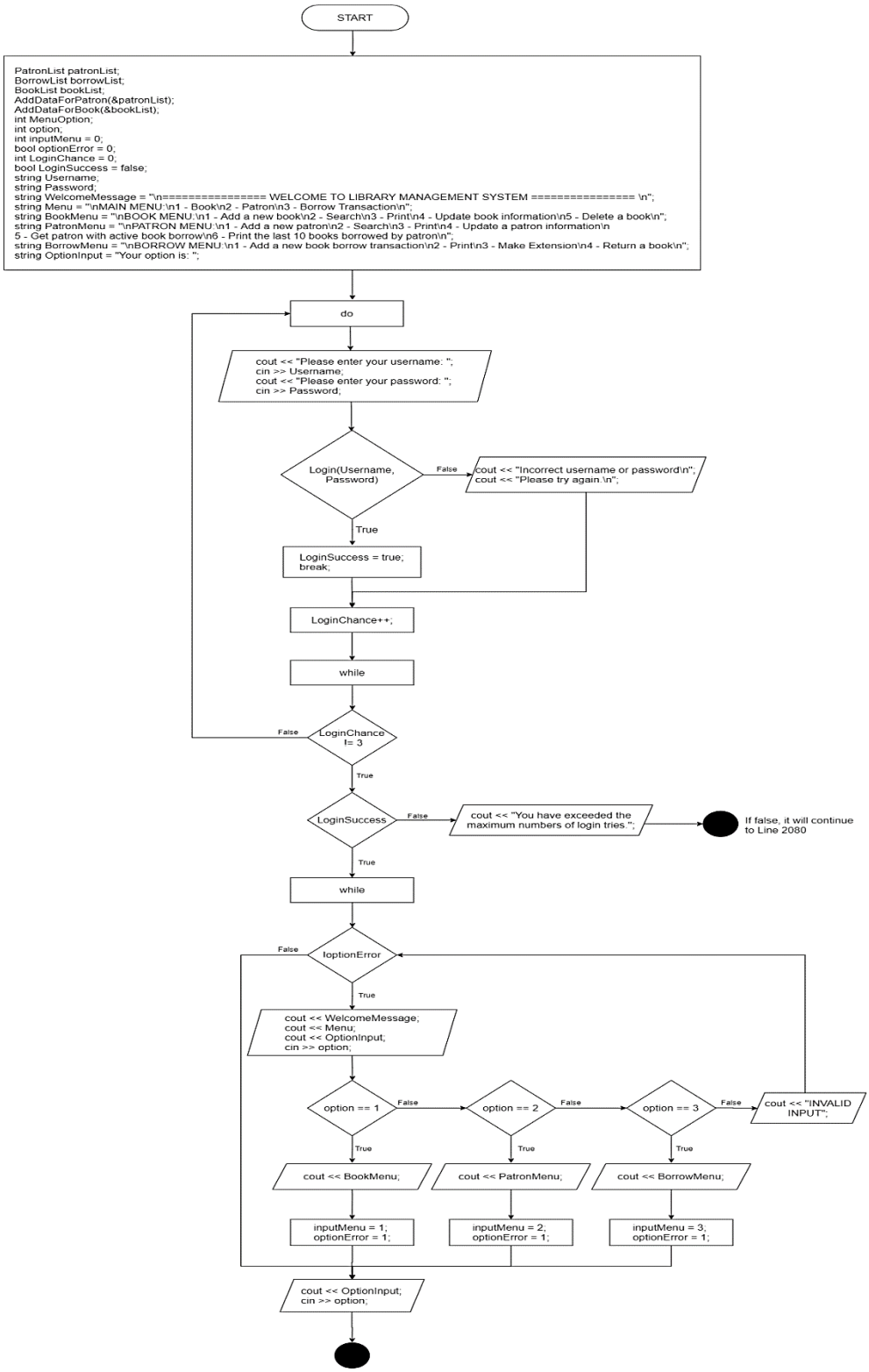


Figure 171: Main method - 1 flowchart

LINK:

https://drive.google.com/file/d/1ICIMfbxinn-Wy2RqVerkhPN79JwmoFza/view?usp=sharing

```
1858 int main() {
1859
1860     // Object instatiation
1861     PatronList patronList;
1862     BorrowList borrowList;
1863     BookList bookList;
1864
1865     //Add dummy data
1866     AddDataForPatron(&patronList);
1867     AddDataForBook(&bookList);
1868
1869     int MenuOption;
1870     int option;
1871     int inputMenu = 0;
1872     bool optionError = 0;
1873     int LoginChance = 0;
1874     bool LoginSuccess = false;
1875     string Username;
1876     string Password;
1877
1878     string WelcomeMessage = "\n=============== WELCOME TO LIBRARY MANAGEMENT SYSTEM =============== \n";
1879     string Menu = "\nMAIN MENU:\n1 - Book\n2 - Patron\n3 - Borrow Transaction\n";
1880     string BookMenu = "\nBOOK MENU:\n1 - Add a new book\n2 - Search\n3 - Print\n4 - Update book information\n5 - Delete a book\n";
1881     string PatronMenu = "\nPATRON MENU:\n1 - Add a new patron\n2 - Search\n3 - Print\n4 - Update a patron information\n5 - Get
         patron with active book borrow\n6 - Print the last 10 books borrowed by patron\n";
1882     string BorrowMenu = "\nBORROW MENU:\n1 - Add a new book borrow transaction\n2 - Print\n3 - Make Extension\n4 - Return a
         book\n";
1883     string OptionInput = "Your option is: ";
1884
1885
1886     do { //Iterate if the LoginChance is not equal to 3
1887         cout << "Please enter your username: ";
1888         cin >> Username;
1889         cout << "Please enter your password: ";
1890         cin >> Password;
1891
1892         if(Login(Username, Password)) {
1893             LoginSuccess = true;
1894             break;
1895         } else {
1896             cout << "Incorrect username or password\n";
1897             cout << "Please try again.\n";
1898         }
1899         LoginChance++;
1900     } while(LoginChance != 3);
1901
1902     if(LoginSuccess) { //If LoginSuccess is true
1903         while(!optionError) {
1904             cout << WelcomeMessage;
1905             cout << Menu;
1906             cout << OptionInput;
1907             cin >> option;
1908
1909             if(option == 1) {
1910                 cout << BookMenu;
1911                 inputMenu = 1;
1912                 optionError = 1;
1913             } else if(option == 2) {
1914                 cout << PatronMenu;
1915                 inputMenu = 2;
1916                 optionError = 1;
1917             } else if(option == 3) {
1918                 cout << BorrowMenu;
1919                 inputMenu = 3;
1920                 optionError = 1;
1921             } else {
1922                 cout << "INVALID INPUT";
1923             }
1924         }
1925
1926         cout << OptionInput;
1927         cin >> option;
1928
```

Figure 172: Main method - 1

The Figure 172 shown above is the Main method -1. Line 1861 to 1883 is the variable declaration. Line 1886 to 1900 will iterate if the LoginChance is not equal to 3. Line 1887 to 1890 allow users to enter the values and get the input values. Line 1892 to 1898 will check if the login credentials are valid, if it does, it will set the LoginSuccess to true and break the iteration. Line 1900 will iterate if the LoginChance is not equal to 3. Line 1902 to 1924 will check if the LoginSuccess is true. Line 1909 to 1912 is to check if the option is equal to 1, if it does, it will display the book menu. Line 1913 to 1916 is to check if the option is equal to 2, if

it does, it will display the patron menu. Line 1917 to 1920 is to check if the option is equal to 3, if it does, it will display the borrow menu.

Result

```
Please enter your username: test
Please enter your password: test
Incorrect username or password
Please try again.
Please enter your username: test1
Please enter your password: test1
Incorrect username or password
Please try again.
Please enter your username: admin
Please enter your password: admin

================ WELCOME TO LIBRARY MANAGEMENT SYSTEM ================

MAIN MENU:
1 - Book
2 - Patron
3 - Borrow Transaction
Your option is:
```

Figure 173: Sample Output 1

```
Please enter your username: test
Please enter your password: test
Incorrect username or password
Please try again.
Please enter your username: test1
Please enter your password: test1
Incorrect username or password
Please try again.
Please enter your username: test2
Please enter your password: test2
Incorrect username or password
Please try again.
You have exceeded the maximum numbers of login tries.Program ended with exit code: 0
```

Figure 174: Sample Output 2

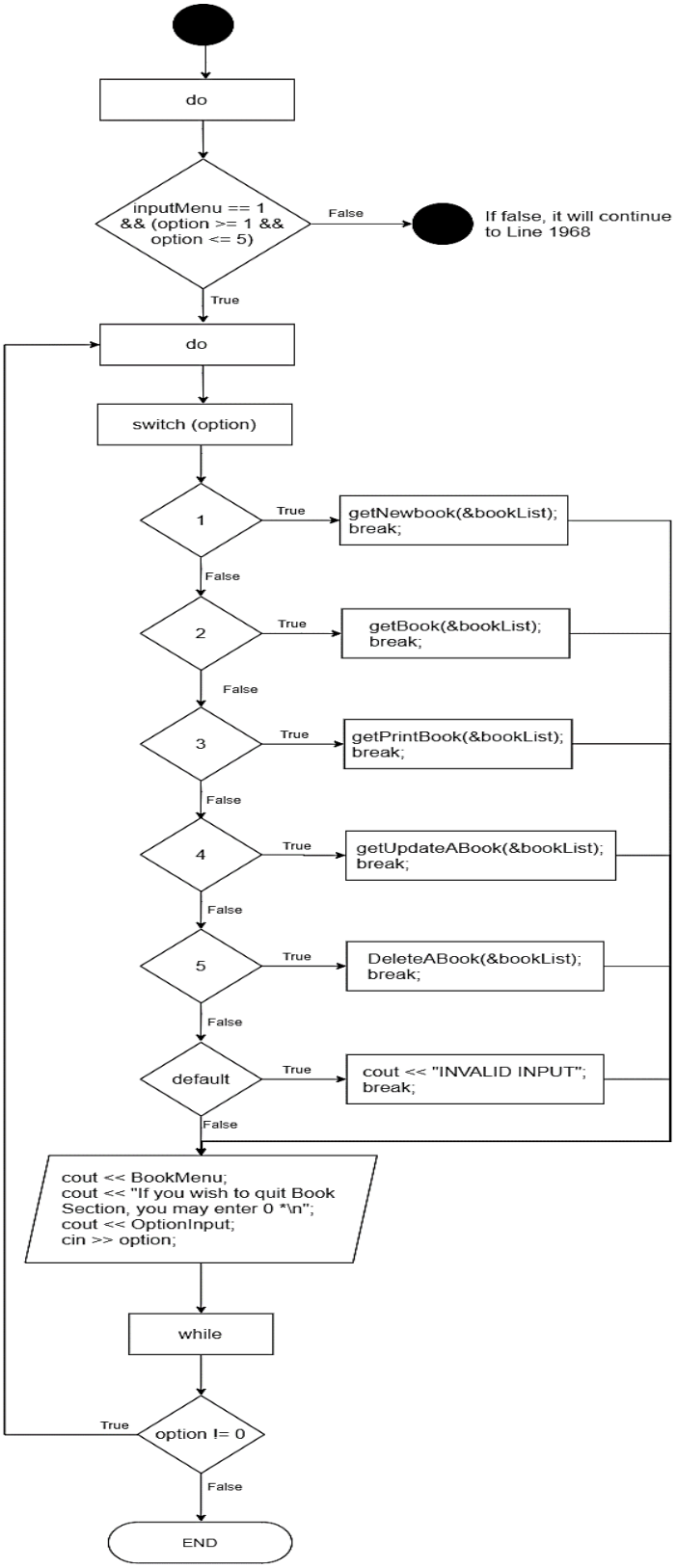


Figure 175: Main method - 2 flowchart

```
1929        do { //Iterate if the MenuOption is not 0
1930
1931            if(inputMenu == 1 && (option >= 1 && option <= 5)) {
1932                do { //Iterate if the option is not 0
1933                    switch(option) {
1934                        case 1: {
1935                            getNewbook(&bookList);
1936                            break;
1937                        }
1938                        case 2: {
1939                            getBook(&bookList);
1940                            break;
1941                        }
1942                        case 3: {
1943                            getPrintBook(&bookList);
1944                            break;
1945                        }
1946                        case 4: {
1947                            getUpdateABook(&bookList);
1948                            break;
1949                        }
1950                        case 5: {
1951                            DeleteABook(&bookList);
1952                            break;
1953                        }
1954                        default: {
1955                            cout << "INVALID INPUT";
1956                            break;
1957                        }
1958                    }
1959
1960                    cout << BookMenu;
1961                    cout << "If you wish to quit Book Section, you may enter 0 *\n";
1962                    cout << OptionInput;
1963                    cin >> option;
1964                } while (option != 0);
1965
1966            }
1967
```

Figure 176: Main method - 2

The Figure 176 shown above is the Main method – 2. Line 1929 to 1966 will iterate if the MenuOption is not 0. Line 1931 is to check is the inputMenu equal to 1 and the option is between 1 and 5. Line 1932 will start to iterate if the option is not 0. Line 1934 to 1937 will be executed if the option value is 1 and the getNewBook(&bookList) method will be called and the break the case. Line 1938 to 1941 will be executed if the option value is 2 and the getBook(&bookList) method will be called and the break the case. Line 1942 to 1945 will be executed if the option value is 3 and the getPrintBook(&bookList) method will be called and the break the case. Line 1946 to 1949 will be executed if the option value is 4 and the getUpdateABook(&bookList) method will be called and the break the case. Line 1950 to 1953 will be executed if the option value is 5 and the DeleteABook(&bookList) method will be called and the break the case. Line 1964 will run the iteration if the option is not equal to 0.

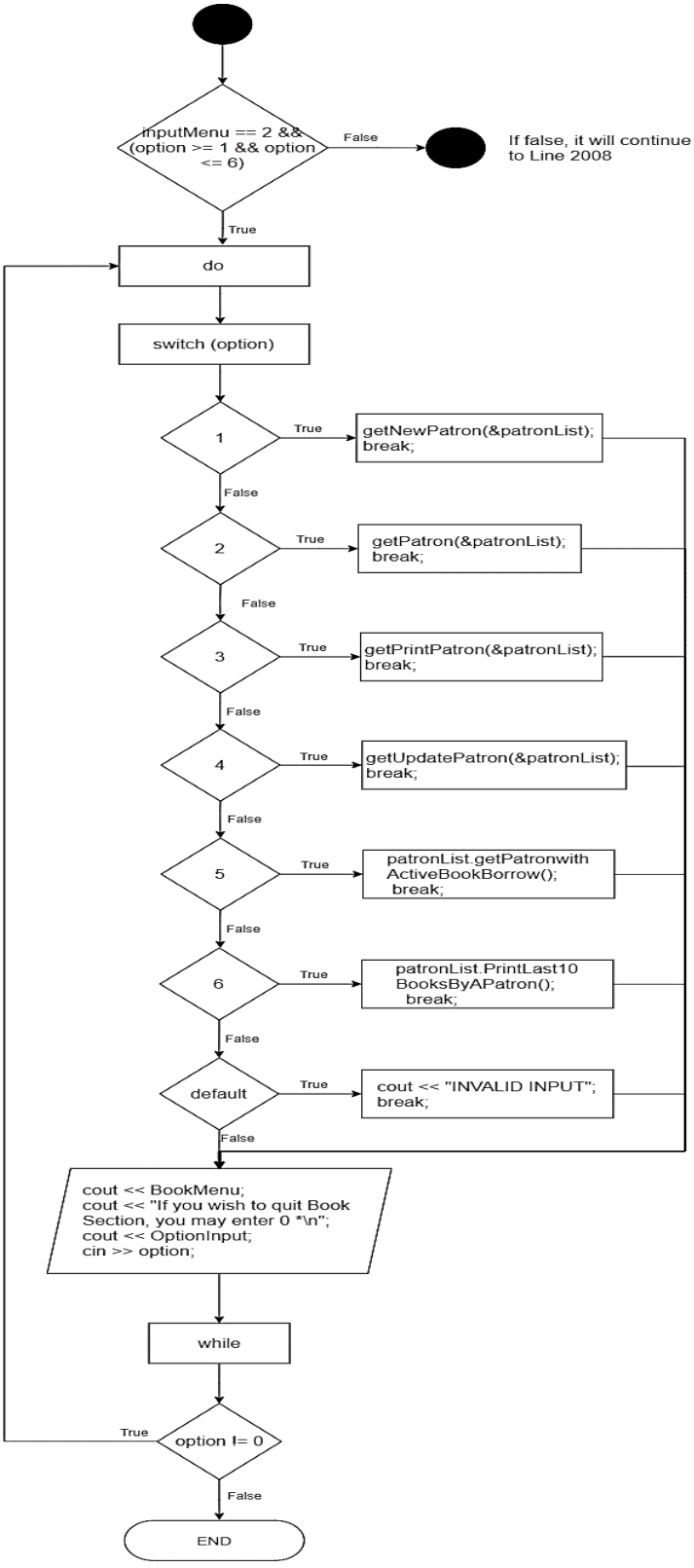


Figure 177: Main method - 3 flowchart

```
1968            if (inputMenu == 2 && (option >= 1 && option <= 6)) {
1969                do { //Iterate if the option is not 0
1970                    switch(option) {
1971                        case 1: {
1972                            getNewPatron(&patronList);
1973                            break;
1974                        }
1975                        case 2: {
1976                            getPatron(&patronList);
1977                            break;
1978                        }
1979                        case 3: {
1980                            getPrintPatron(&patronList);
1981                            break;
1982                        }
1983                        case 4: {
1984                            getUpdatePatron(&patronList);
1985                            break;
1986                        }
1987                        case 5: {
1988                            patronList.getPatronwithActiveBookBorrow();
1989                            break;
1990                        }
1991                        case 6: {
1992                            patronList.PrintLast10BooksByAPatron();
1993                            break;
1994                        }
1995                        default: {
1996                            cout << "INVALID INPUT";
1997                            break;
1998                        }
1999                    }
2000
2001                    cout << PatronMenu;
2002                    cout << "If you wish to quit Patron Section, you may enter 0 *\n";
2003                    cout << OptionInput;
2004                    cin >> option;
2005                } while (option != 0);
2006            }
```

Figure 178: Main method - 3

The Figure 178 shown above is the Main method – 3. Line 1968 is to check is the inputMenu equal to 2 and the option is between 1 and 6. Line 1969 will start to iterate if the option is not 0. Line 1971 to 1974 will be executed if the option value is 1 and the getNewPatron(&patronList) method will be called and the break the case. Line 1975 to 1978 will be executed if the option value is 2 and the getPatron(&patronList) method will be called and the break the case. Line 1979 to 1982 will be executed if the option value is 3 and the getPrintPatron(&patronList) method will be called and the break the case. Line 1983 to 1986 will be executed if the option value is 4 and the getUpdatePatron(&patronList) method will be called and the break the case. Line 1987 to 1990 will be executed if the option value is 5 and the getPatronwithActiveBookBorrow(&patronList) method will be called and the break the case. Line 1991 to 1994 will be executed if the option value is 6 and the PrintLast10BooksByAPatron(&patronList) method will be called and the break the case. Line 2005 will run the iteration if the option is not equal to 0.

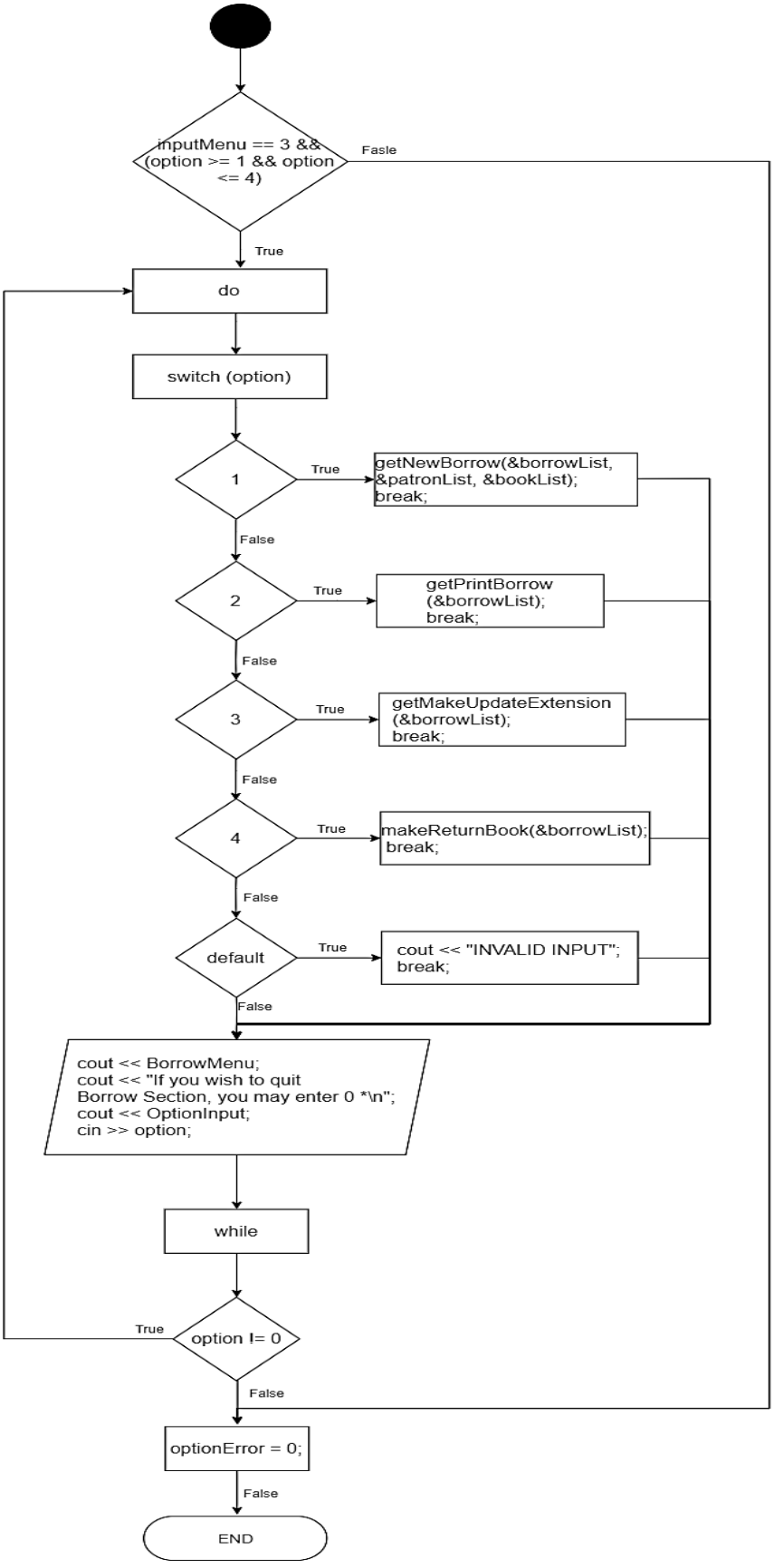


Figure 179: Main method - 4 flowchart

```
2007
2008            if (inputMenu == 3 && (option >= 1 && option <= 4)) {
2009                do { //Iterate if the option is not 0
2010                    switch(option) {
2011                        case 1: {
2012                            getNewBorrow(&borrowList, &patronList, &bookList);
2013                            break;
2014                        }
2015                        case 2: {
2016                            getPrintBorrow(&borrowList);
2017                            break;
2018                        }
2019                        case 3: {
2020                            getMakeUpdateExtension(&borrowList);
2021                            break;
2022                        }
2023                        case 4: {
2024                            makeReturnBook(&borrowList);
2025                            break;
2026                        }
2027                        default: {
2028                            cout << "INVALID INPUT";
2029                            break;
2030                        }
2031                    }
2032
2033                    cout << BorrowMenu;
2034                    cout << "If you wish to quit Borrow Section, you may enter 0 *\n";
2035                    cout << OptionInput;
2036                    cin >> option;
2037
2038                } while (option != 0);
2039
2040            }
2041
2042            //Enter back to the Main Menu
2043            optionError = 0;
2044
```

Figure 180: Main method - 4

The Figure 180 shown above is the Main method – 4. Line 2008 is to check is the inputMenu equal to 3 and the option is between 1 and 4. Line 2009 will start to iterate if the option is not 0. Line 2011 to 2014 will be executed if the option value is 1 and the getNewBorrow(&borrowList, &patronList, &bookList) method will be called and the break the case. Line 2015 to 2018 will be executed if the option value is 2 and the getPrintBorrow(&borrowList) method will be called and the break the case. Line 2019 to 2022 will be executed if the option value is 3 and the getMakeUpdateExtension(&borrowList) method will be called and the break the case. Line 2023 to 2026 will be executed if the option value is 4 and the makeReturnBook(&borrowList) method will be called and the break the case. Line 2038 will run the iteration if the option is not equal to 0.

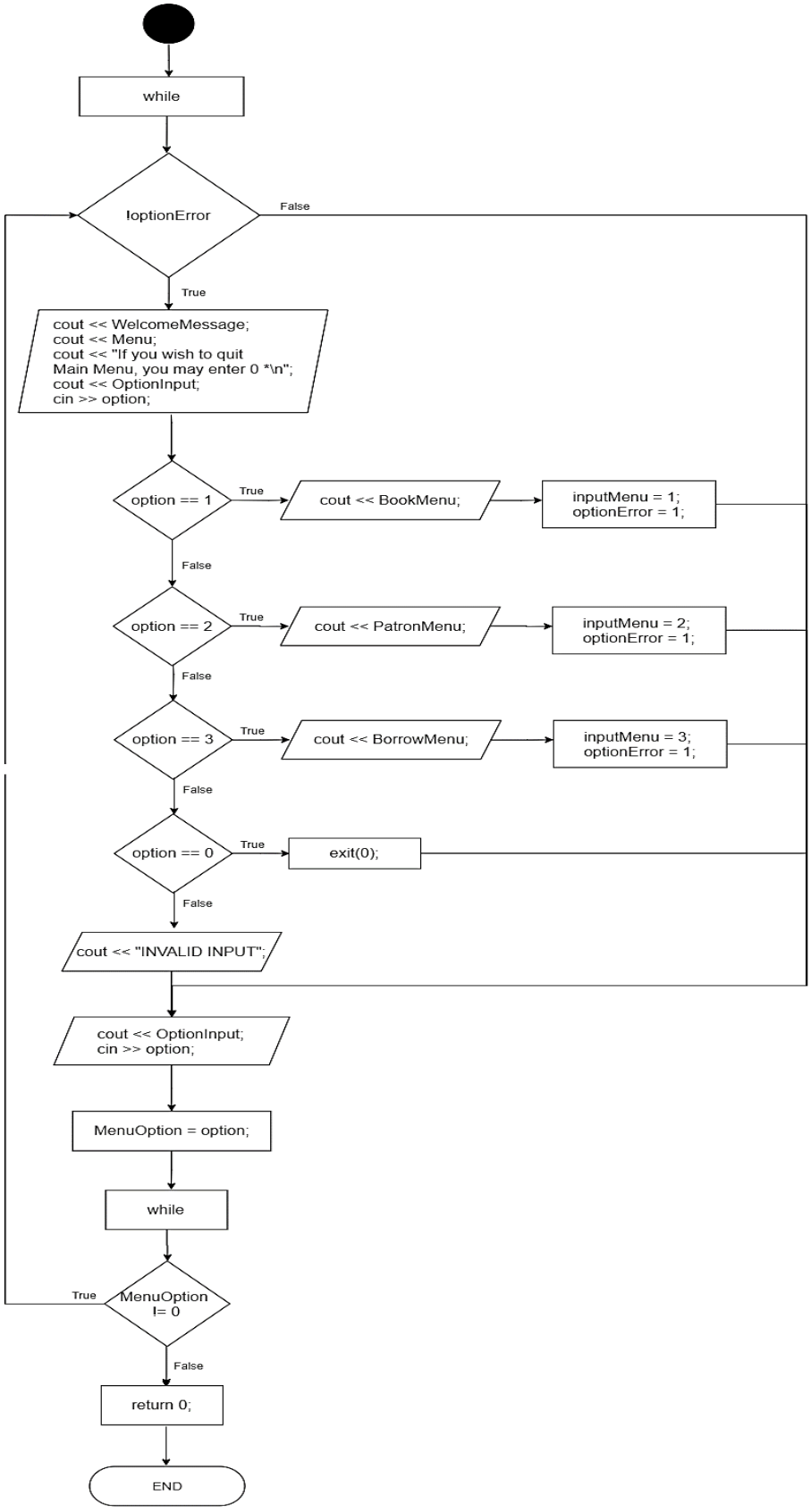


Figure 181: Main method - 5 flowchart

```
2045            while(!optionError) {
2046                cout << WelcomeMessage;
2047                cout << Menu;
2048                cout << "If you wish to quit Main Menu, you may enter 0 *\n";
2049                cout << OptionInput;
2050                cin >> option;
2051
2052                if(option == 1) {
2053                    cout << BookMenu;
2054                    inputMenu = 1;
2055                    optionError = 1;
2056                } else if(option == 2) {
2057                    cout << PatronMenu;
2058                    inputMenu = 2;
2059                    optionError = 1;
2060                } else if(option == 3) {
2061                    cout << BorrowMenu;
2062                    inputMenu = 3;
2063                    optionError = 1;
2064                } else if(option == 0) {
2065                    exit(0);
2066                } else {
2067                    cout << "INVALID INPUT";
2068                }
2069            }
2070
2071            cout << OptionInput;
2072            cin >> option;
2073            MenuOption = option;
2074
2075        } while (MenuOption != 0);
2076    } else {
2077        //If the loginChance is more than 3 times
2078        cout << "You have exceeded the maximum numbers of login tries.";
2079    }
2080    return 0;
2081 }
```

Figure 182: Main method - 5

The Figure 182 shown above is the Main method – 5. Line 2046 to 2050 is to allow the users to enter the values and to get the input values. Line 2052 is to check is the option equal to 1, if it does, it will display the book menu. Line 2056 is to check is the option equal to 2, if it does, it will display the patron menu. Line 2060 is to check is the option equal to 3, if it does, it will display the borrow menu. Line 2064 is to check is the option equal to 0, if it does, it will exit the system. Line 2075 will run the iteration if the menu option is not equal to 0. Line 2080 will return value 0.

4.0 Assumption

In this project, an LMS is proposed to AACL to manage the library books and library patrons. The proposed LMS will be validating the Patron information before adding a new patron into the database (linked list), for example, patron ID will be used to verify if the patron ID is existing in the database (linked list). If it does, the proposed LMS then will be notifying the user, saying that the detection of duplication of patron ID. Otherwise, a new patron information is successfully added.

Book ID will be auto generated and assigned to the new book when adding a new book. The proposed LMS will be validating if the book title and author is existing in the proposed system, because the same book name will normally not be published by the same author, therefore, the proposed LMS will validate the book title and author before adding to the linked list. This is to prevent the same book title from the same author happened in the proposed system. However, one author with multiple books is allowed or the same book titles with multiple authors are allowed. The system will auto add two copies whenever a new book added with status of available by default. The two copies will have similar information such as book title, book author, book genre, book category, and even book ISBN. The difference between each copy is the book ID.

The borrow ID will be auto generated and assigned to the new borrow list as well when a new borrow transaction is made. The proposed system will be automatically assigning today's date as the borrow date, and return date is within 15 days starting from the borrow date. Book borrowed extension is possible to the proposed LMS also, but only each patron can make only one extension for each borrowed book because, by taking into the consideration, other patrons may want to borrow the books. This is to prevent that the book is borrowed by one patron exceed more than 30 days. To overcome this issue, book borrow extension will be only available once for each borrow transaction. Therefore, other patrons can have opportunities to borrow the books. Each patron will have a maximum number of three books borrowed at any given time, the proposed LMS will be checking if the patron has returned the book. If a patron borrows three books at any given time and not returned the book yet and trying to borrow another book, the proposed LMS will be showing an error message, saying that the patron has reached its maximum number of books borrowed. Moreover, the library book in this proposed LMS can be deleted, but the condition is the copies of the book must be available.

5.0 Limitation and Future Enhancement

One of the limitations of this project is the patron cannot use the Library Manage System since all the books information and library patron will be managed only by the admin. The next limitation is that the book will only have two copies when a new book is added into the system, which the book copies are fixed to be added by its system. The last limitation is that each book borrowed by the patron can only be extended once. For future enhancement, the library patron can use the system to log into the system by using the login account credentials, and to borrow and return the books. Moreover, the system should be developed in such a way where the admin can enter amount of book copies to be added for that particular book. The Library Management System should allow patron to extend the return date more than one time.

References

Arun, P., 2020. *Doubly Linked List (Data Structure) In C++.* [Online]
Available at: https://www.hackerearth.com/practice/notes/doubly-linked-list-data-structure-in-c/#:~:text=A%20doubly%2Dlinked%20list%20is,in%20the%20sequence%20of%20nodes.
[Accessed 26 July 2020].

Bhatt, T., 2020. *Doubly Linked List: An introduction.* [Online]
Available at: https://medium.com/@tarunbhatt9784/doubly-linked-list-an-introduction-a81aa8c4016e
[Accessed 01 August 2020].

c4learn.com, 2015. *Linked list advantages.* [Online]
Available at: http://www.c4learn.com/data-structure/linked-list-advantages/
[Accessed 31 July 2020].

Frai, M., 2009. *How to get current time and date in C++?.* [Online]
Available at: https://stackoverflow.com/questions/997946/how-to-get-current-time-and-date-in-c
[Accessed 1 September 2020].

Kakria, R., 2020. *What are Advantages and Disadvantages of Doubly Linked List.* [Online]
Available at: http://www.xpode.com/ShowArticle.aspx?Articleid=295
[Accessed 21 July 2020].

Mishra, N., 2016. *Doubly Linked List in C and C++.* [Online]
Available at: https://www.thecrazyprogrammer.com/2015/09/doubly-linked-list-in-c-and-cpp.html
[Accessed 25 July 2020].

Pankaj, 2015. *Data Structure : Doubly Linked List.* [Online]
Available at: https://codeforwin.org/2015/10/doubly-linked-list-data-structure-in-c.html
[Accessed 21 July 2020].

Rajput, D., 2018. *Find the nth node from the end of a singly linked list ?.* [Online]
Available at: https://www.dineshonjava.com/nth-node-from-the-end-of-a-singly-linked-list/
[Accessed 31 July 2020].

Rudolph, K., 2009. *How do I tokenize a string in C++?.* [Online]
Available at: https://stackoverflow.com/questions/53849/how-do-i-tokenize-a-string-in-c
[Accessed 01 September 2020].

SoftwareTestingHelp, 2020. *Doubly Linked List Data Structure In C++ With Illustration.* [Online]
Available at: https://www.softwaretestinghelp.com/doubly-linked-list-2/#:~:text=A%20doubly%20linked%20list%20is%20a%20variation%20of%20the%20singly%20linked%20list.&text=A%20doubly%20linked%20list%20is%20also%20a%20collection%20of%20nodes,Depth%20C%2B%2B%20Training%20T
[Accessed 25 July 2020].

Studytonight.com, 2020. *Doubly Linked List.* [Online]
Available at: https://www.studytonight.com/data-structures/doubly-linked-list
[Accessed 26 July 2020].

Thakur, A. S., 2017. *Doubly Linked List | C++ Implementation.* [Online]
Available at: https://medium.com/programmercave/doubly-linked-list-c-implementation-e24686aa3d6d
[Accessed 26 July 2020].

yangerleo, 2019. *Advantage and disadvantages of singly linked list.* [Online]
Available at: https://brainly.in/question/10363542
[Accessed 31 July 2020].

Yashas, 2020. *Singly Linked List C++.* [Online]
Available at: https://www.codespeedy.com/singly-linked-lists-in-cpp/
[Accessed 31 July 2020].