Introduction:

In this project, I am creating a Console Based C++ application that will allow the user to maintain the records of books stored in the Library. As I am creating this project to overcome the difficulties that are faced by users in manual record keeping, therefore, I am using special techniques to make user interaction better. Here are some of the key features of this application. I am using a dynamic data structure in this app so there is no fixed limit of records. I can load many books from file in our app and moreover, I can add new books that will be saved in the file along with the previously loaded data. Second, I can search the books using the Title as search key. I need to enter the book title and if it matches, then the result will be displayed. I have another functionality called "Partial Search" in which I do not have to write the whole title of book. If I forgot, or do not know whole title of book, I can just write a partial key word from title and it will display all records matching that search key. I can also delete a record by searching the book first by the title, then deleting it from the data-structure if found. This software is a menu based and user can select different operations from menu or select "Exit" if he/she wants to terminate the program. A user can load data from other files also and save new data into those files also but the data in those files should be according to the standard format in which the four attributes are separated by a "Tab" or "\t", otherwise it will give an error.

This was a brief introduction of our Project, now I will discuss the other aspects of this project. Here is the layout of the contents of this report:

1- Justification of Data Structure

2- Analysis of Algorithms that provided Key Functionality

3- Testing Statements

4- Table of test cases

5- Summary of Work Done

6- Limitations and critical reflection

7- How would change approach on similar task in future

8- Harvard References.

Justification Of Data Structure:

The Data Structure that I have used in this project is a "Singly Linked List". An array could also be selected here but linked list has a lot of advantages over array. So, I selected Linked List because it can fulfil all the major requirements such as addition, deletion and searching (CORMEN, T. H. (2009)). Following are the major reasons for selection of Linked List:

1) **Dynamic Data Structure:**
   Linked list is a dynamic data structure which means it can grow and shrink at runtime by allocation and deallocation of memory. Therefore, I do not need to specify a fixed initial size because it can grow to unknown size according to the scenario.
2) **Adding and deleting Data:**
   Addition and deletion of nodes are really easy. Unlike array, here I do not have to shift elements after insertion or deletion of an element. In linked list I just need to update the address present in next pointer of a node. (CORMEN, T. H. (2009))

3) **Memory Management:**
   As size of linked list can increase or decrease at run time so there is no memory wastage. In case of array there is lot of memory wastage. I need to declare a fixed size of array and it is not every time when I will occupy whole space. If I declare an array of size 10 and store only 6 elements in it then space of 4 elements is wasted. Similarly, if I have an array of 1000 elements but I read 2000 records from file then there will be an overflow error (CORMEN, T. H. (2009)). There is no such problem in linked list as memory is allocated only when required.
4) **Other Techniques:**
   Another great advantage of the Linked List is that they can be implemented as Queue or Stacks. So, if there is a scenario in which I have to distribute books in a queue then I can achieve this using Linked List.

Analysis of Algorithms:

Add Book Algorithm:

```
string addbook(string title,string author, string isbn, int quantity)
    if title != ""
        declare node*n
        set n = new node
        set n->title = title
        set n->author = author
        set n->isbn = isbn
        set n->quantity = quantity
        set n->next = head
        set head = n

        declare node * temphead = head
        set temphead = head
        declare string temptitle
        declare string tempauthor
        declare string tempisbn
        declare int tempquantity
        declare int counter
```

```
set counter = 0  //16
//16
set temphead = head
//17
        while  temphead->next

            if  temphead->Title > temphead->next->Title
                set temptitle = temphead->Title
                set temphead->Title = temphead->next->Title
                set temphead->next->Title = temptitle

                set tempauthor = temphead->Author
                set temphead->Author = temphead->next->Author
                set temphead->next->Author = tempauthor

                set tempisbn = temphead->ISBN
                set temphead->ISBN = temphead->next->ISBN
                set temphead->next->ISBN = tempisbn

                set tempquantity = temphead->Quantity
                set temphead->Quantity = temphead->next->Quantity
                set temphead->next->Quantity = tempquantity

            set temphead = temphead->next

        set temphead = head

    return message
    //17 + n +n^2
end addbook
```

As a whole , time Complexity is O(n^2) because of the time taken to add and the sort function

Search Book Algorithm:

```
string searchBook(string searchkey)
        if isempty() =="Empty"
                return "error"
        //2
        else
                declare node *ptr
                set ptr = head
                declare integer found
                set found = 0
                //6
                while ptr!=NULL   //N
                        if ptr->title == searchkey
                                print ptr->title,ptr->author,ptr->isbn,ptr->quantity;
                                set found++
                                return " Found message"


                        end if
                        set ptr=ptr->next
                end while
                //11+N
                if found == 0
                        print "not found
                end if
                //13+N
end searchBook
```

T(n) = 13 + n

So, over all complexity is O(n)

Delete Book Algorithm:

```
string deleteBook(string searchKey)
        declare node *ptr
        set ptr = head
        declare integer found
        set found = 0
        //4
        while ptr!=NULL   //N
                if ptr->title == searchkey
                        print ptr->title,ptr->author,ptr->isbn,ptr->quantity;
                        set found++
                        return " Found message"

                end if
                set ptr=ptr->next
        end while
        //7+N
        if found == 0
                print "not found
        end if
        //9+N
end deleteBook
```

T(n) = 9 + n

So, over all complexity is O(n)

Statement of Testing Approach:

I used Failure Based and Testing approach for this testing.

Table of Test Cases:

Objective of Test Case	Pre-Requisite	Steps	Input	Expected Output	Actual Output	Status
Check if Linked list is Empty or not	Data should not be Loaded from file into linked list.	Select Search, Delete or Display Function	Null	Empty	Empty	Pass
Add Book in Linked List	None	Select Add book function, then add the values.	Title, author, ISBN, Quantity	Book added successfully	Book added successfully	Pass
Delete Book	Data should be Loaded from file into linked list.	Select Delete book options, Enter search key	Search Key (123)	false	false	Pass
Load Data	File should be present with correct format of data.	Select load data option and enter file name	File name (books.txt)	Records Successfully loaded from File.	Records Successfully loaded from File.	Pass
Load Data Error	File should not be present.	Select load data option and enter file name	File name (bo00oks.txt)	Error! File not found.	Error! File not found.	Pass

```
Manu@LAPTOP-28K1L9BS /cygdrive/c/M00735429
$ ./testcases
All tests passed (5 assertions in 5 test cases)
```

Summary of work done:

Till now, I have written the code and fulfilled all major requirements including file handling, adding, deleting and searching of data. As all testing has been done and test cases have been passed so the software is complete. I can also run the software with makefile by using make command in the terminal.

Limitations of this program:

There are some Limitations of the data structure that I have used in this project. They are described as follow:

Memory Usage:
More memory is required to store elements in linked list as compared to array. Because in linked list each node contains a pointer and it requires extra memory for itself.

Traversal:
Elements or nodes traversal is difficult in linked list. I do not randomly access any element as I do in array by index. For example, if I want to access a node at position n then I need to traverse all the nodes before it. So, time required to access a node is large.

How would change approach on similar task in future:

Following improvements can be made in future in order to improve the interaction of user with program:

1) Console based program should be converted to GUI based program.

2) More efficient data structure like Hash table can be introduced.

3) The searching mechanism is Case sensitive, this should be improved so that user can search more efficiently.

4) Exception Handling should be introduced so that program does not crash on wrong inputs from user.

References:

Dr Barry D. Nichols (no date) Algorithms Introduction [PDF]. Available at: https://mdx.mrooms.net/pluginfile.php/2483026/mod_resource/content/0/algorithms_introduction.pdf (Accessed: 2 April 2021).

[14:40]

CORMEN, T. H., & CORMEN, T. H. (2009). Introduction to algorithms. Cambridge, Mass, MIT Press.

https://mdx.mrooms.net/pluginfile.php/2489637/mod_resource/content/0/sorting_handout.pdf

Algorithms Book c++.pdf