CST 2550 coursework 2

Aditaya Santokee

2021

CST 2550 coursework 2

M00738443 Final report



Table of Contents

[Introduction 2](#_Toc70700212)

[Design 2](#_Toc70700213)

[Search. (Anon., n.d.) 2](#_Toc70700214)

[Insert (Friesen, n.d.) 3](#_Toc70700215)

[3](#_Toc70700216)

[Delete (team, n.d.) 3](#_Toc70700217)

[Time complexity analysis 3](#_Toc70700218)

[Adding 3](#_Toc70700219)

[Search 3](#_Toc70700220)

[Delete 4](#_Toc70700221)

[Testing 4](#_Toc70700222)

[Conclusion 4](#_Toc70700223)

[Summary 4](#_Toc70700224)

[Reflection on implementation 4](#_Toc70700225)

[Alternate approaches 5](#_Toc70700226)

[Bibliography 6](#_Toc70700227)

# Introduction

A library has found their system needs an upgrade as their search system is slower now that their collection of books has grown. They have approached us with the task to design and implement a system for them which will allow them to search for a book by title, add new books and deleting books

This report will cover the design process of the search system program, this will include the overall design, the test cases we have implemented and time-complexity analysis we have prepared while designing the program to ensure the program works at its optimal best.

* The report layout will be as follows:  
  The design – This section will cover information regarding the algorithms and data structures used, it will also justify my choices and validate it by means of pseudo code, time complexity analysis.
* Testing- This section will contain information regarding the testing approach used as well as the test cases which will be displayed in table format.
* The conclusion- This will include a summary of the report, a critical reflection of the implementation, design process and the limitations faced during the process, pointers on how I would approach a similar task in the future will also be listed here.
* References- The references to validate my facts and in-text references

# Design

There are many algorithms that could have been a possible option for this coursework. However, after much research I believe the optimal Algorithm is linked list. Compared to an array, a linked list is more dynamic in size and it’s easier to insert and delete data.

### Search. (Anon., n.d.)

The recursive solution:

**bool search (head, x)**

1) If head is NULL, return false.

2) If head's key is same as x, return true;

2) Else return search(head->next, x)

Pseudo code:

bool search (struct Node\* head, int x)

{

    // Base case

    if (head == NULL)

        return false;

    // If key is present in current node, return true

    if (head->key == x)

        return true;

    // Recur for remaining list

    return search(head->next, x);

}

### Insert (Friesen, n.d.)

Pseudo code:

Void insert(){

temp = NEW **Node**

temp.name = "C"

temp.**next** = NULL

DECLARE **Node** temp2

temp2 = top

WHILE temp2.**next** NE NULL

temp2 = temp2.**next**

**END** WHILE

temp2.**next** = temp

}

### jw 3527188 pt4fig4 1200x186

Figure 1:Overview of insertion (Friesen, n.d.)

### Delete (team, n.d.)

Pseudo code:

**void** **delete** (**int** pos)

{

node \*current=**new** node;

node \*previous=**new** node;

current=head;

**for** (**int** i=1; i<pos; i++)

{

previous=current;

current=current->next;

}

previous->next=current->next;

}

## Time complexity analysis

### Adding

Adding a book to the library database will be O(n)--linear. Its time complexity could be improved to O (1) by maintaining a reference to the last nod

### Search

The time complexity of searching will be 0(1)

### Delete

To delete a book from the database, the time complexity will be 0(n).

# Testing

I will be using the Unit testing approach to ensure the programs functionalities are working.

The following are the functionalities that will need to work:

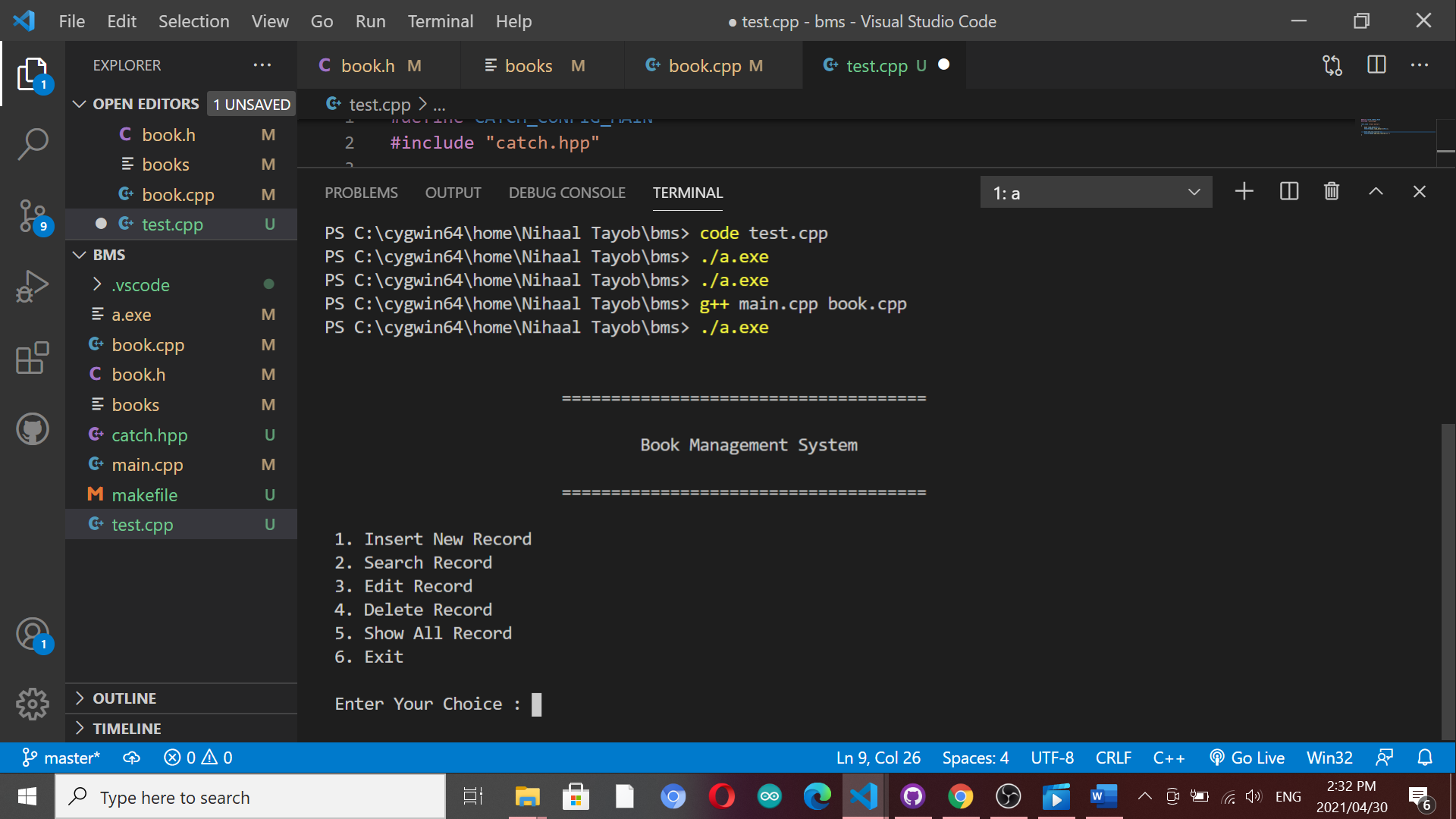
* Adding a new book
* Searching for a book
* Delete a book

# Conclusion

## Summary

The application designed consists of 1 class, book.

When the application is run, the data is loaded using the book.loadData() function. The data is loaded from a file. The application then displays the menu where the user will choose what they would like to do.



The insert new record will ask the user to the required information and store it in a file. Each record will be stored on a new line, however information about a single record will be stored in a single line. The various data inserted about a record will be separated using tabs.

The search and edit function will ask the user to input the name of the book and the function will then perform the necessary tasks.

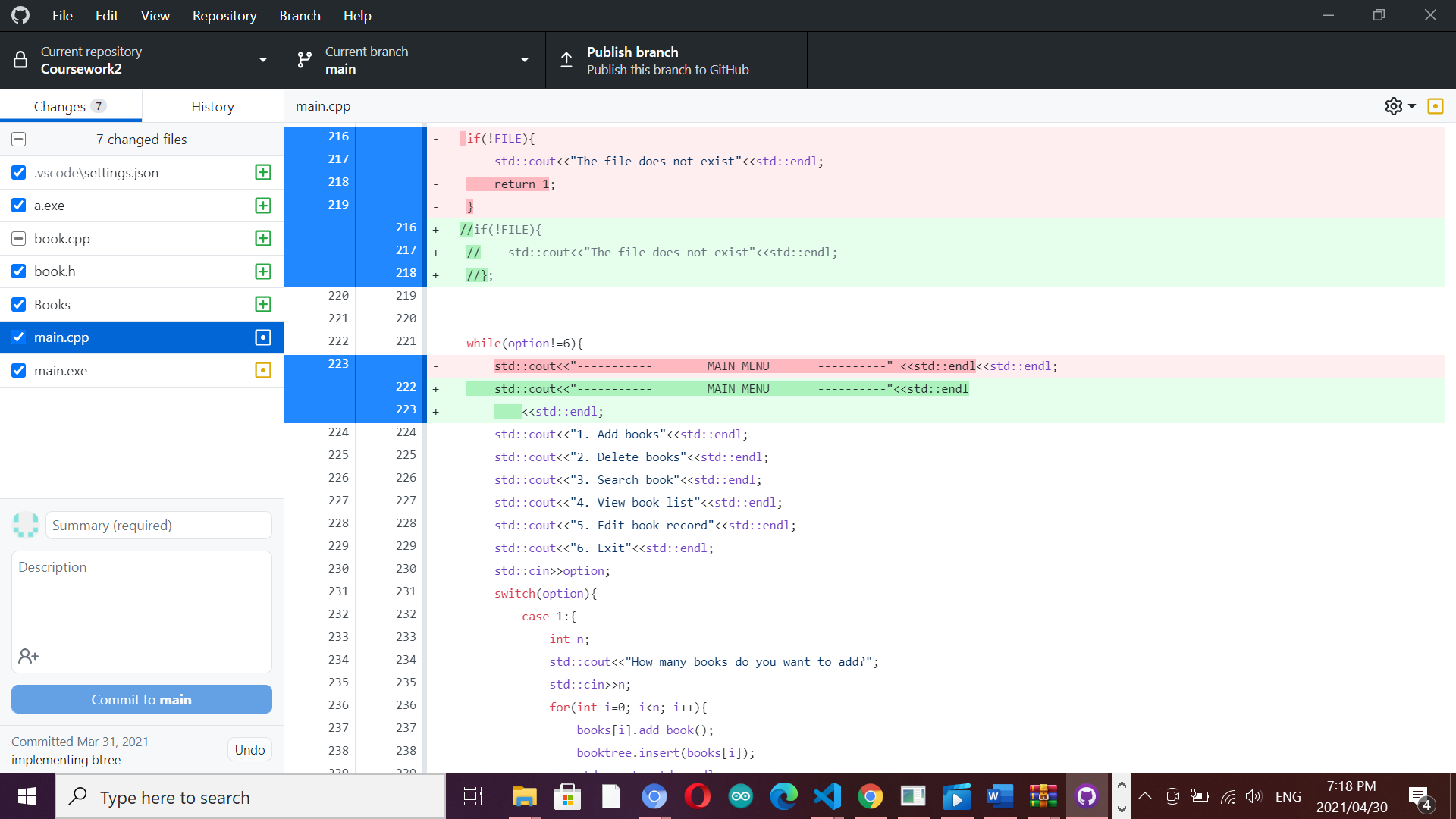
The edit and delete record functions are more complex as they required for a new file to be created, the data to be deleted/changed, the new data as well as all the existing data will then be transferred to a new file. The old file will be deleted and the new file will be renamed to match the old files name.

The show all function will read the data from the file and display it using a table format.

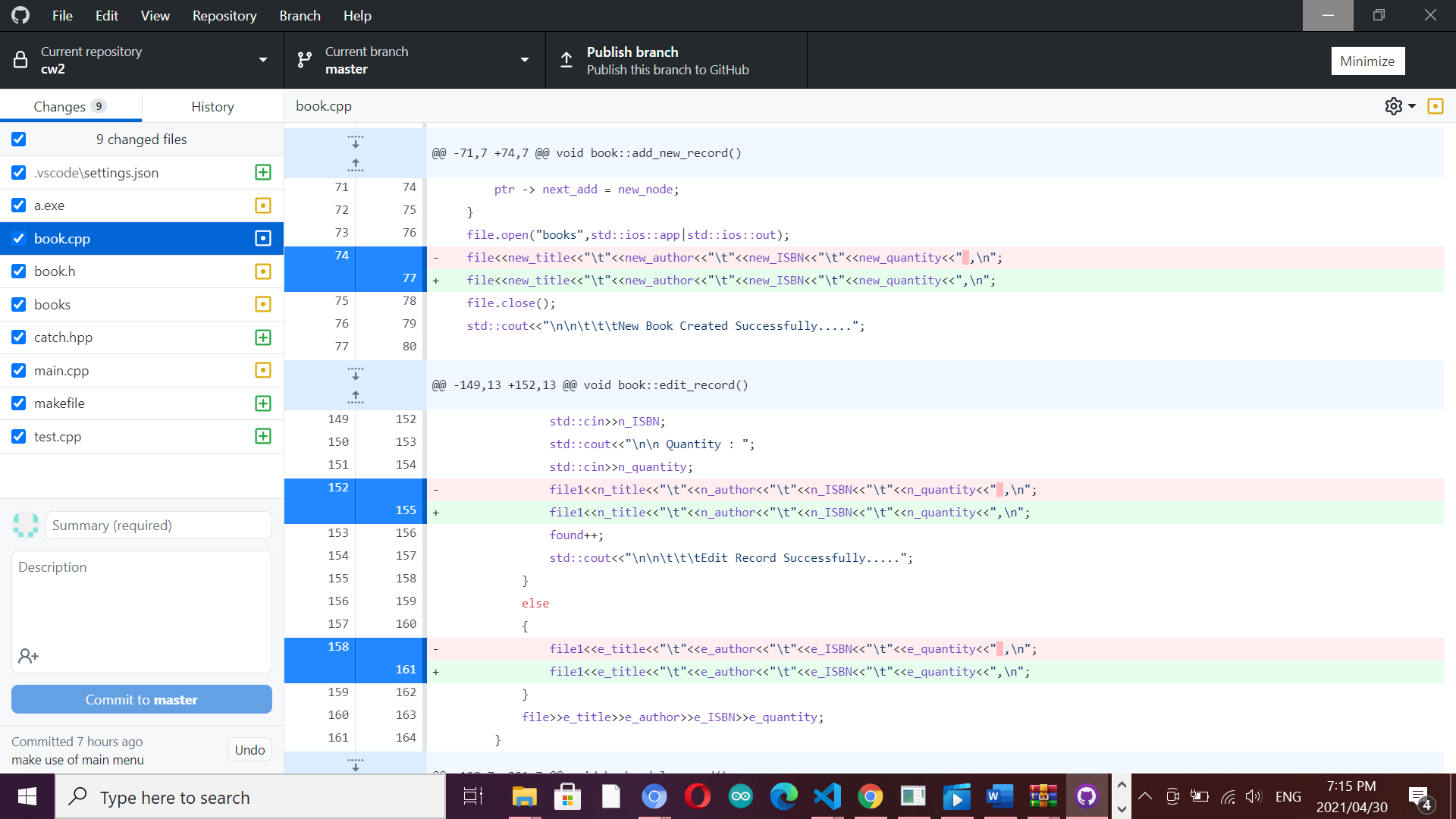
## Reflection on implementation

The application I have created is simple. The design is simple. However, in order to implement more functionalities a more complex application is required. I have implemented more functionalities than required, as stated before, the insertion of data is relatively easy. Making use of more than one data structure would be more complex but I feel it would be best in terms of functionality for future projects.

Initial github repository



Recent repository



## Alternate approaches

I am quite happy with the outcome considering the faults I have received along the way. One of the problems I have faced was that my git hub would, after some time display a message saying “file not found” when I ran the .exe file. I was required to re-create my git hub repository numerous times. An alternate approach I would have looked into, if I had not chosen linked list due to its functionalities is a binary tree as that would have been, in my opinion, the second-best option.

# Bibliography

Anon., n.d. *BlogsDope.* [Online]   
Available at: https://www.codesdope.com/blog/article/c-linked-lists-in-c-singly-linked-list/

Dr, n.d. [Online].

Friesen, J., n.d. *InfoWorld.* [Online]   
Available at: https://www.infoworld.com/article/3266674/data

Available at: https://mdx.mrooms.net/pluginfile.php/2528433/mod\_resource/content

team, c., n.d. *codementor.* [Online]   
Available at: https://www.codementor.io/@codementorteam/a-comprehensive-guide-to-implementation-of-singly-linked-list-using-c\_plus\_plus-ondlm5azr