

**Preparatory data Structure (CSCI 591)**



**Project - IV**

**Evaluating an Inventory using a linked list**

**February 20, 2020**

**St. Cloud state university**

**Department of Computer Science**

Submitted By: Taddese Erba

# **Design Document**

## **Introduction**

A linked list is a linear structure of ordered objects that are stored at random memory locations and linked together by pointers. A linked list can be a singly linked list or a doubly-linked list. A doubly linked list stores data in between two fixed containers usually called the header and the trailer. The header is the first Node and the trailer is the last node in the linked list.

This project will implement an inventory list using the MyInventory class. The MyInventory class consists of the inv.h header file where all the private and public variables and functions declared. The inv.cpp implementation file contains all the implementation for the classes. The main.cpp file is the testing file for all the implemented class functions.

## **Data Structure**

This program has three distinct files. The inv.h file contains all the declaration of the required functions and a few function decoration (implementation). It is the framework for MyInventory class implementation. It consists of two private objects, the struct Node object which is used to lold the two main components of a node and the Node \* getNode(long int num\_0, string name\_0, double p\_0, int q\_0, Node \* list) function used to declare and initialize the nodes. In addition to the two private objects, the header file contains twelve public functions which include two constructors, one destructor, eight operational functions, and one friend function.

## **Functions**

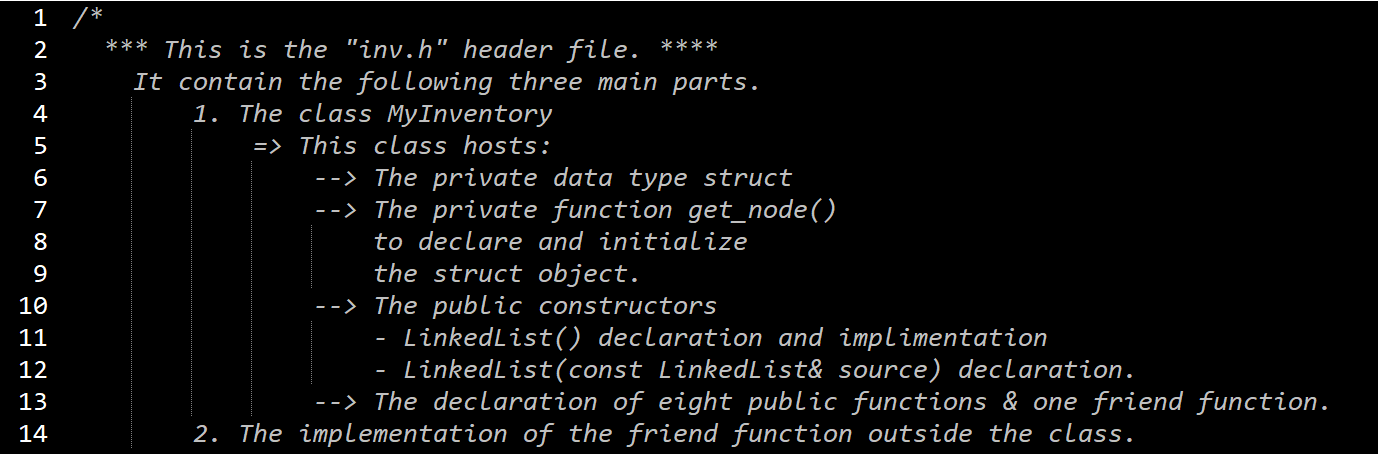
As mentioned in the Data Structure section, there are twelve functions in this project. The first two functions, the LinkedList() and LinkedList(const LinkedList& source) are constructors. The MyInventory ()constructor function is used to initialize the class. The constructor MyInventory(const MyInventory& source) is a copy constructor that is used to copy the elements of the list. The third function, ~ MyInventory (), is a destructor. It is used to delete all the nodes, deallocate the memory, and return it to the operating system. The void re\_Initialize() function is used to re-initialize the linked lists to empty. The void insert(long int new\_num, string new\_name, double new\_p, int new\_q) function is used to insert items into the list. The void remove(long int old\_num) function is used to remove a node from the list. This function uses the inventory number as a reference to search and remove the node. The void operator = (MyInventory source) is a function that overloads the assignment operator (=) to be used in the assignment operation involving copying the elements of the list. The bool isEmpty() function returns true if the list is empty or false otherwise. The int listLength() function will count each node in the list and return the size of the list (number of nodes). The bool isPresent(long int t\_num) will check if an item is in the list and returns true. This function also uses the inventory number to search for the item. If the item is not found it returns false to main(). The long int kthValue(int numval) function will return the kth node of the list. If the node is not found, or the list is empty, it returns nothing. The friend ostream& operator << (ostream& out\_s, const MyInventory& l) is a friend function that is used to overload the outstream operator (<<) for the purpose of printing all the elements of the list. For convenience reasons, the friend function is implemented in the header file where it was declared but outside of the class MyInventory.

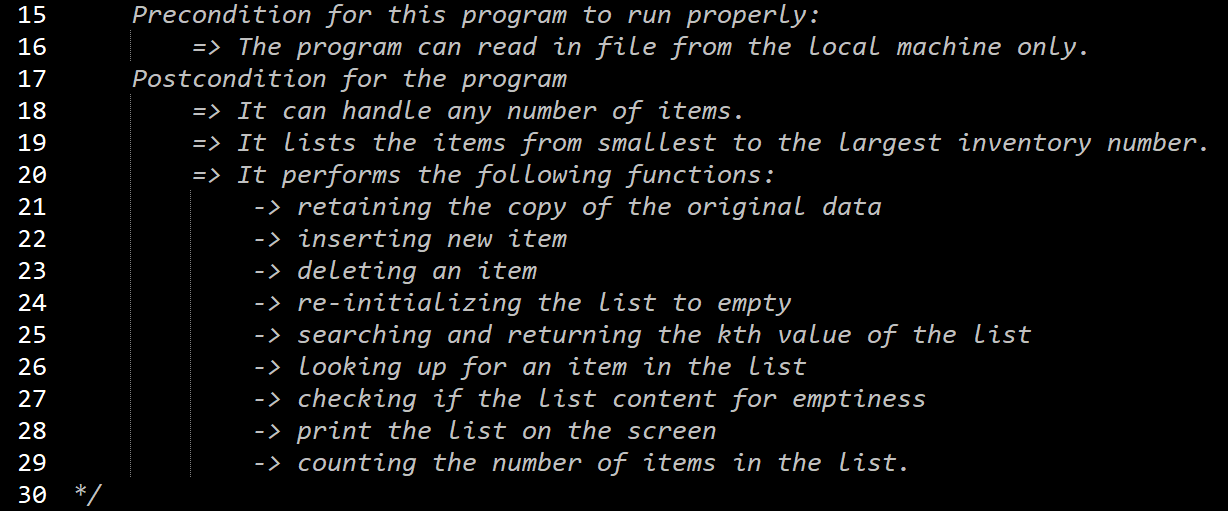
## **The Main Program**

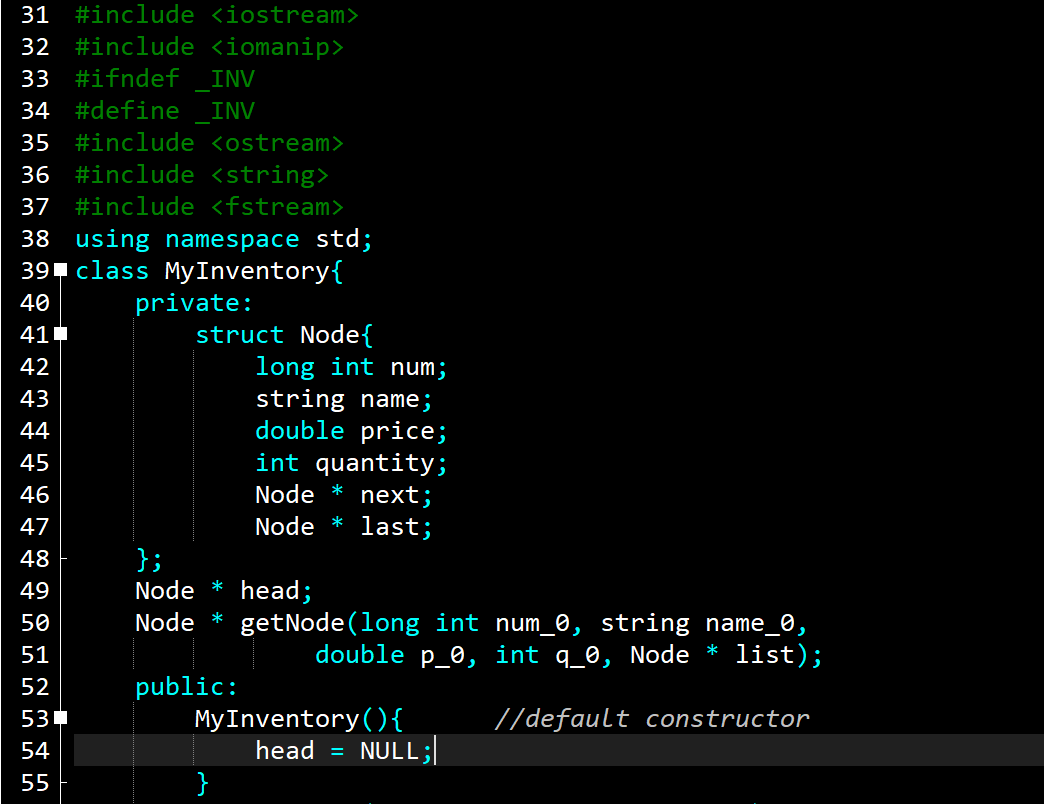
As a testing function, where the implementation is tested, there are many things going on in the main() function. To keep things simple, I will talk only about the main components of the main() function. The key frameworks in the main() function are the instantiation of the class MyInventory and representation of the key operations by a menu system. There are ten main menus from which the user can choose to perform an operation. The menus are represented by alphabets that are closely related to the operation followed by the name of the operation as in ***I -- Insert Item*** and ***R -- Remove Item***. The menus are continuously displayed after each operation until the user chooses to quit the program. A switch statement will track each choice of the user and perform the necessary operation accordingly. It may also worth mentioning the bool searchArray(const char [], int, char) function that is the part of the main() function that is used to search the array of constants that hold the alphabets designated to the menu. This enables that if the user enters a choice that is not available, the program can display the necessary message and exit the program.

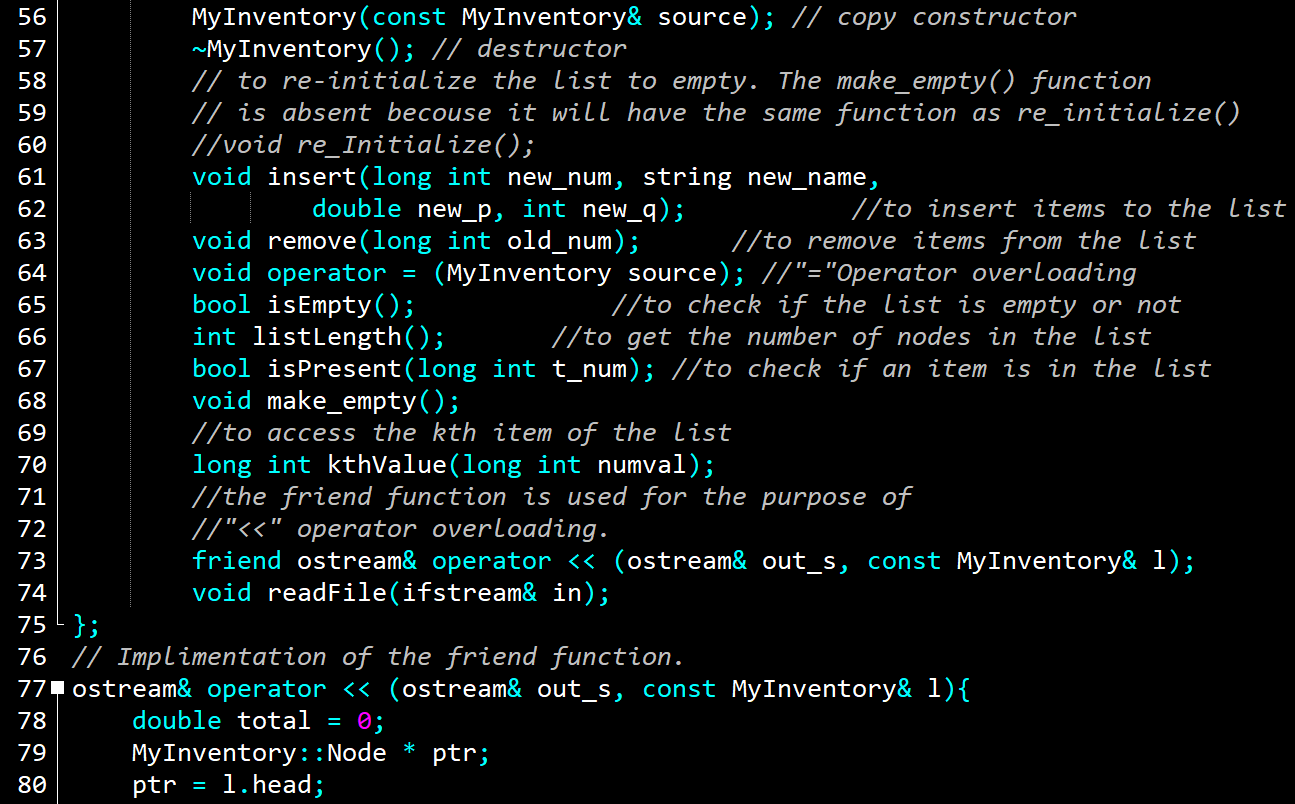
# **Code listing**

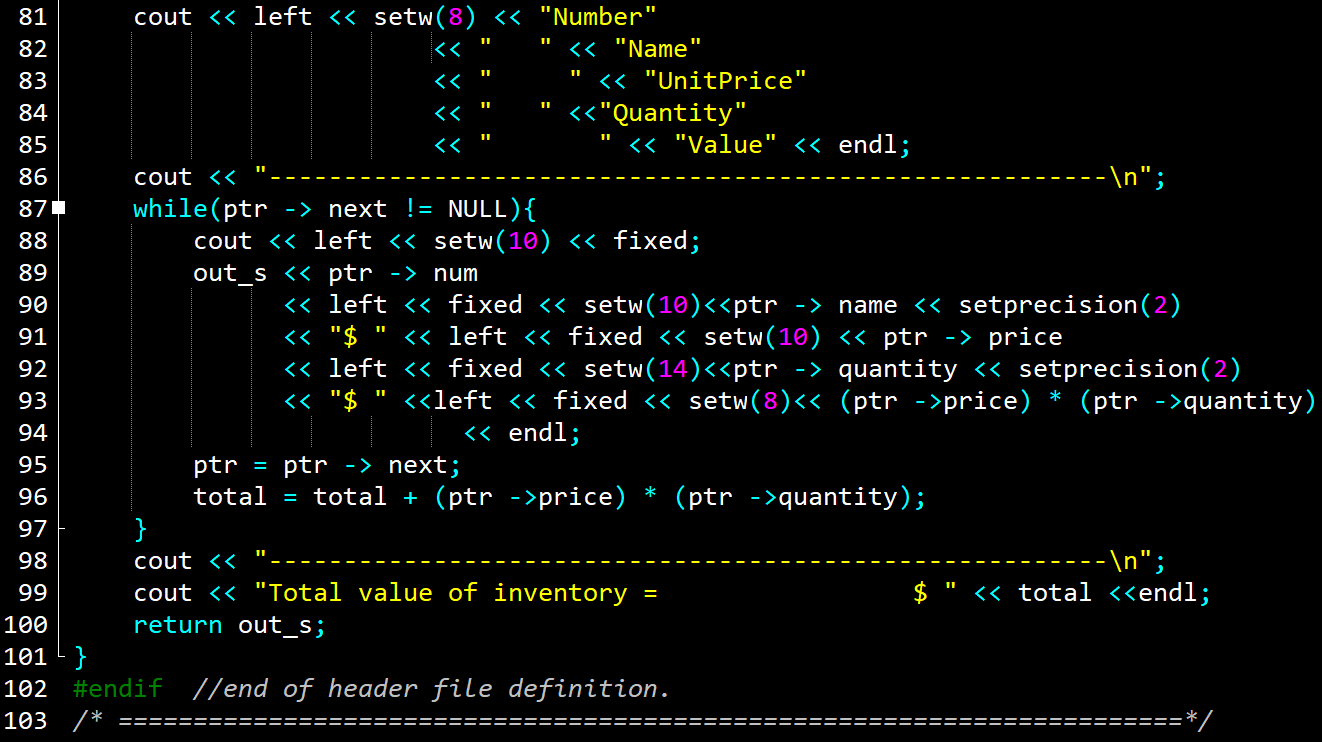
## **The header file (List.h)**



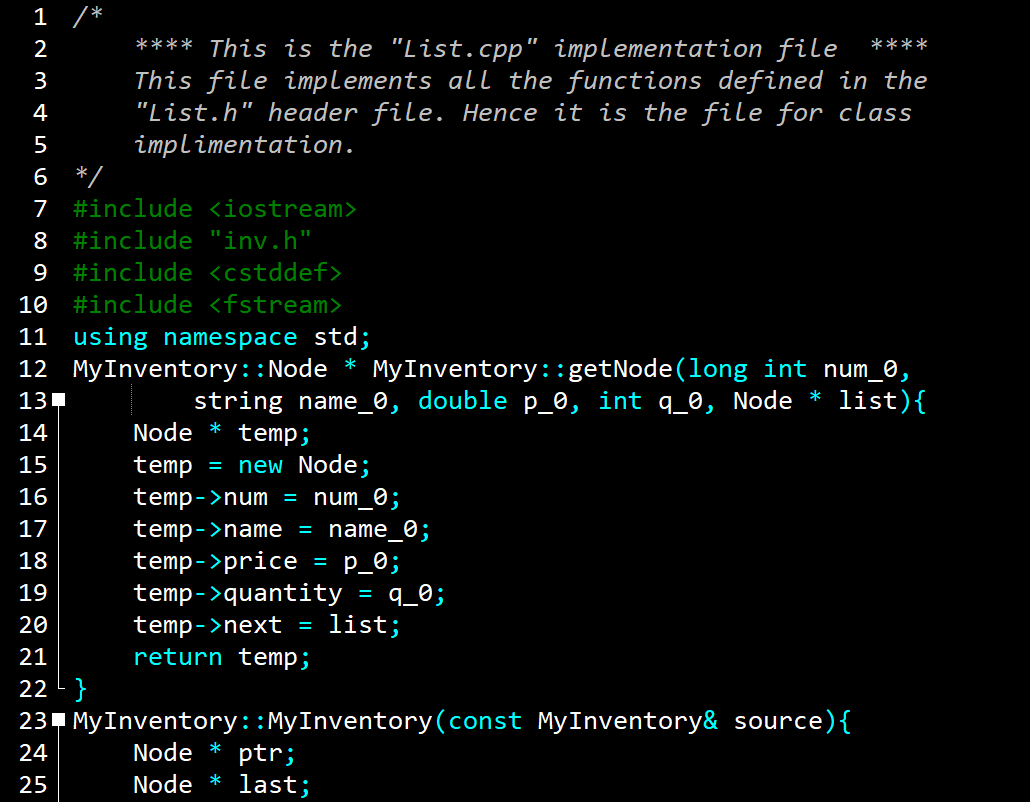


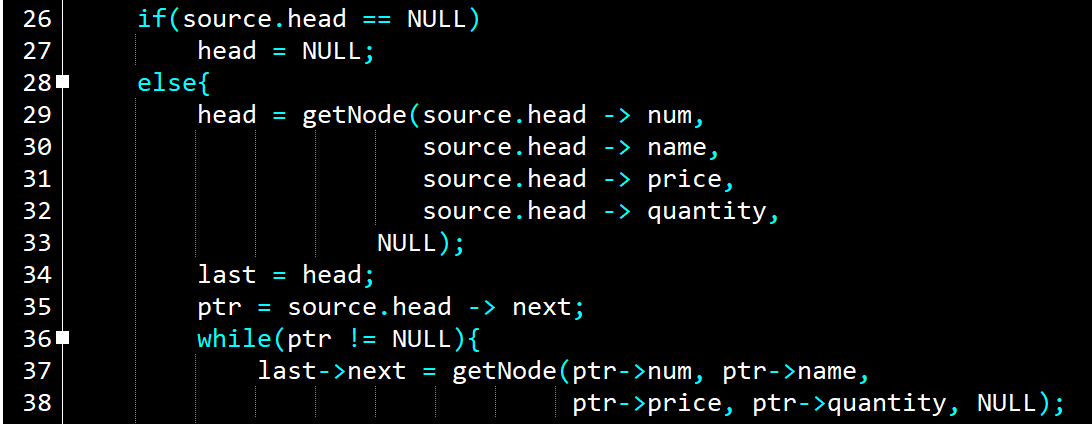


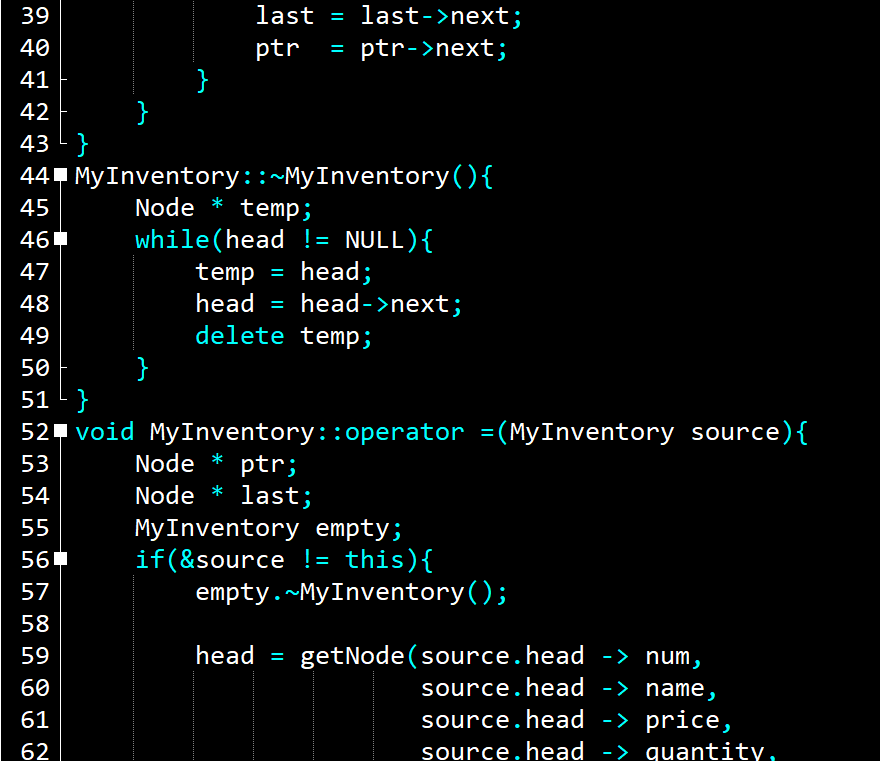


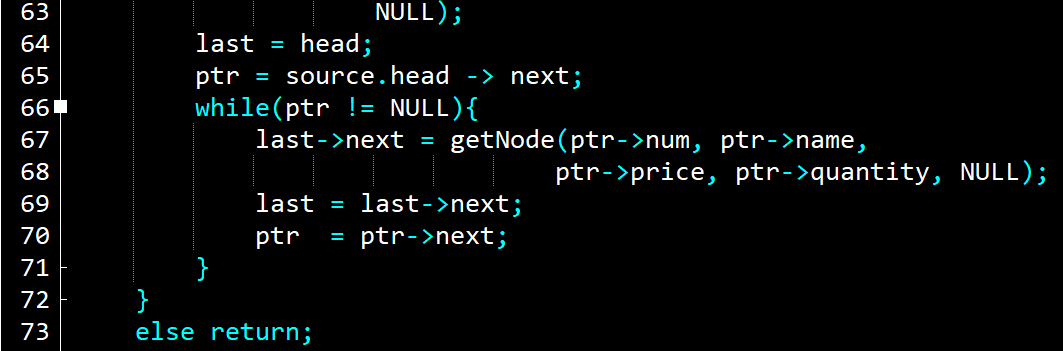


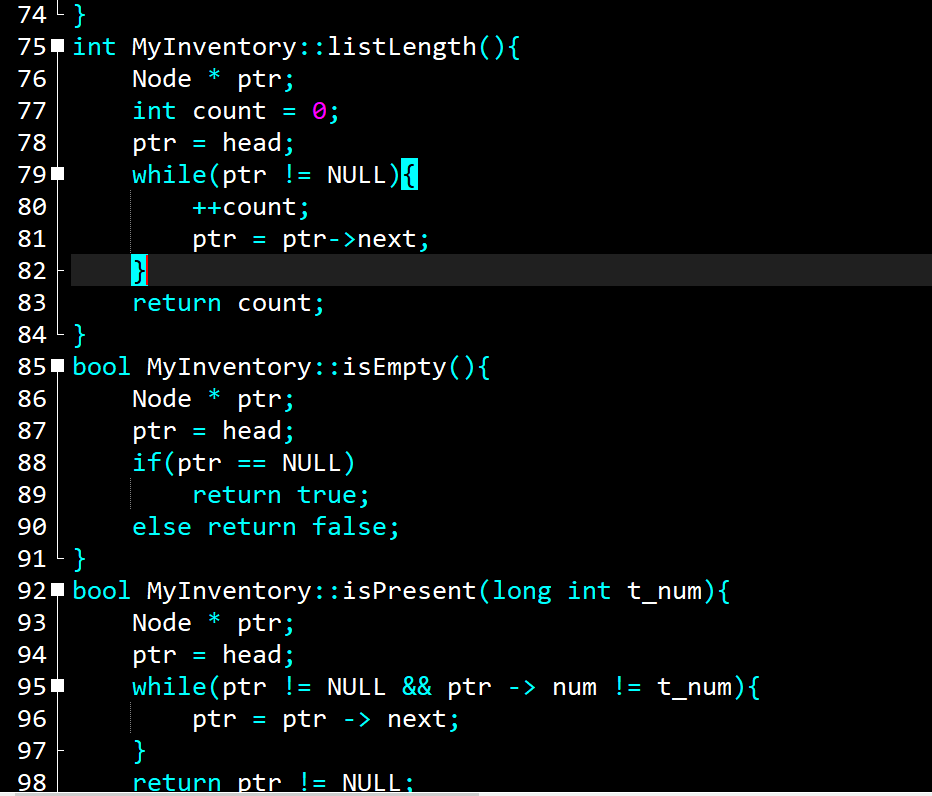
## **The implementation file (List.cpp)**

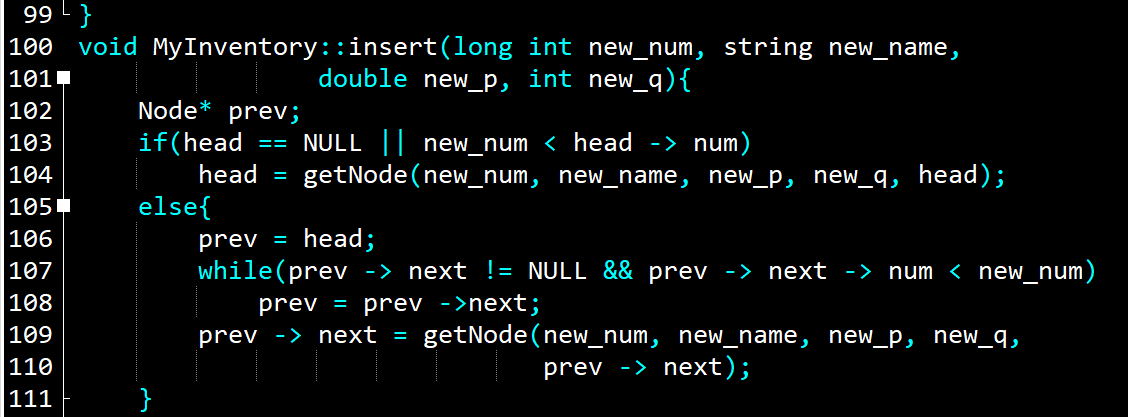


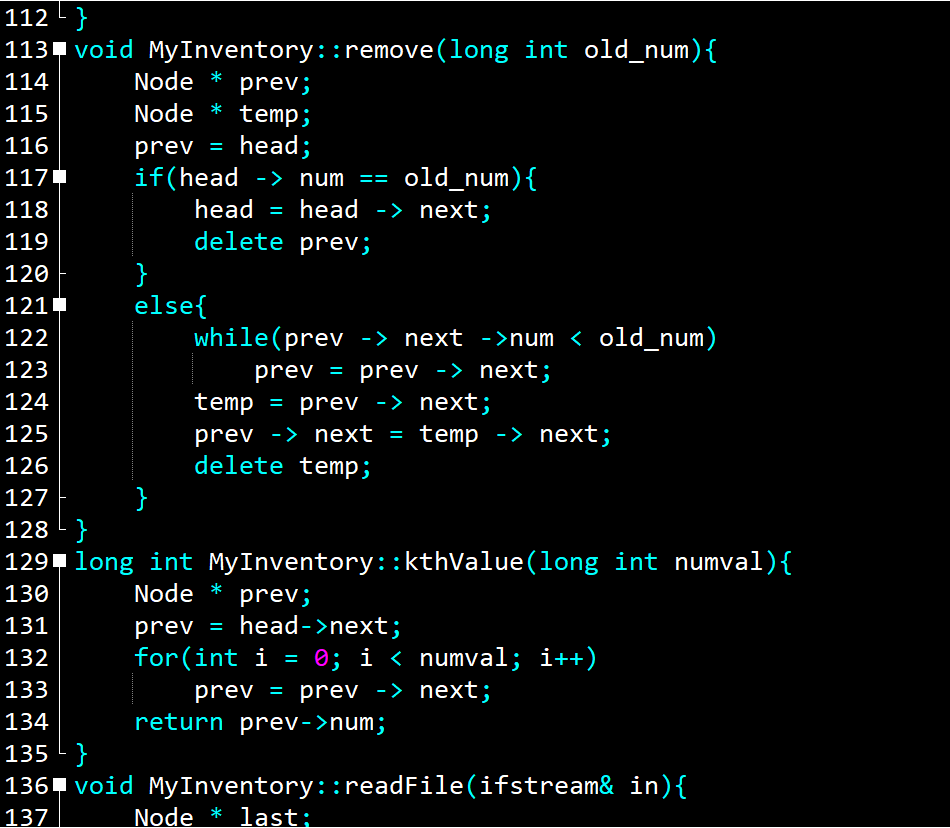


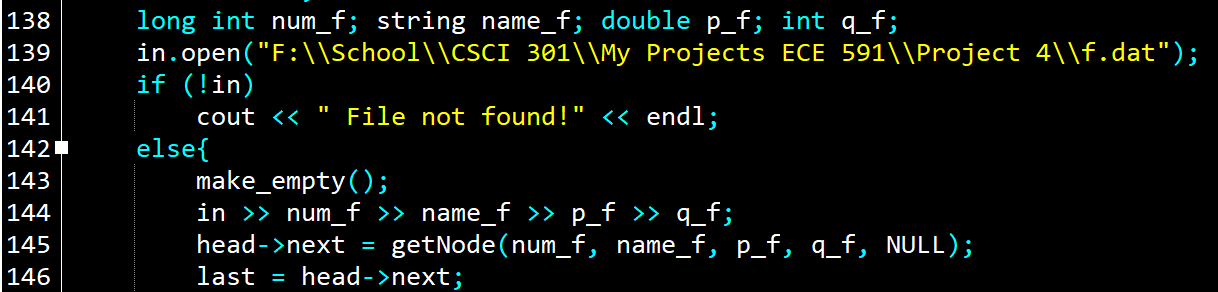


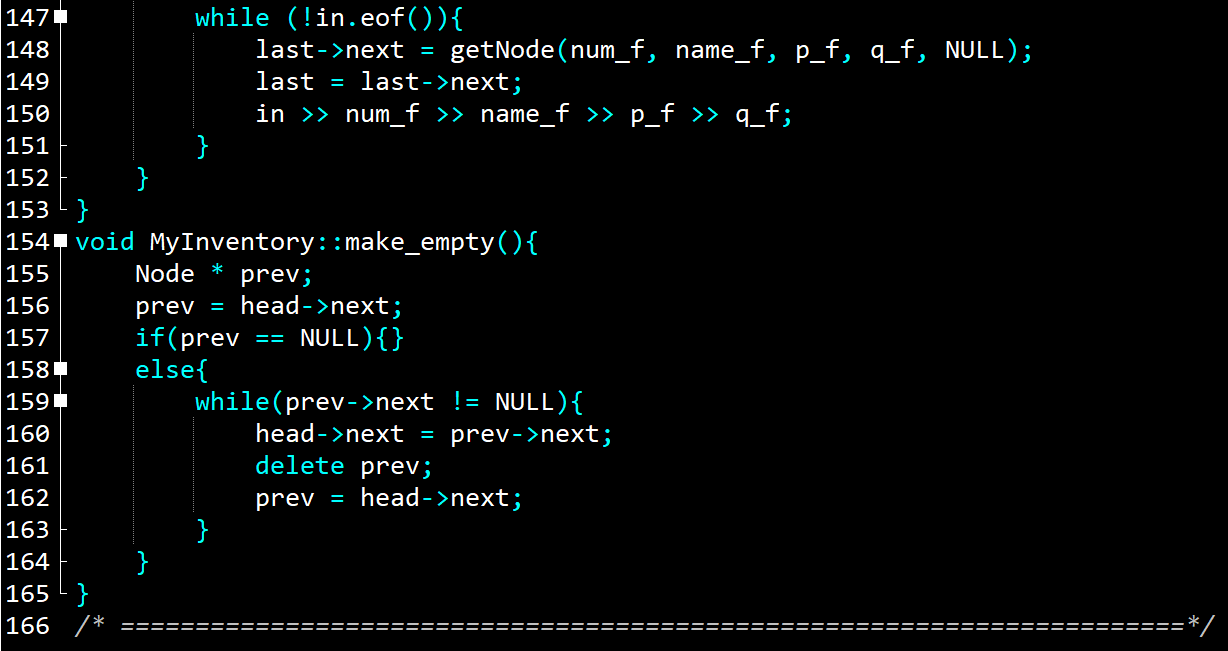




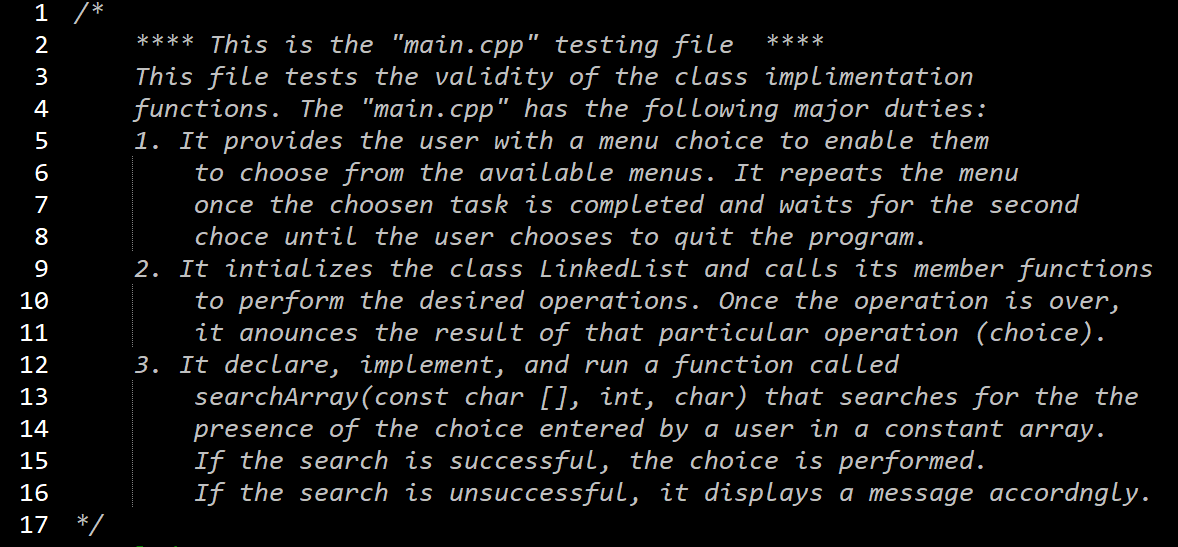


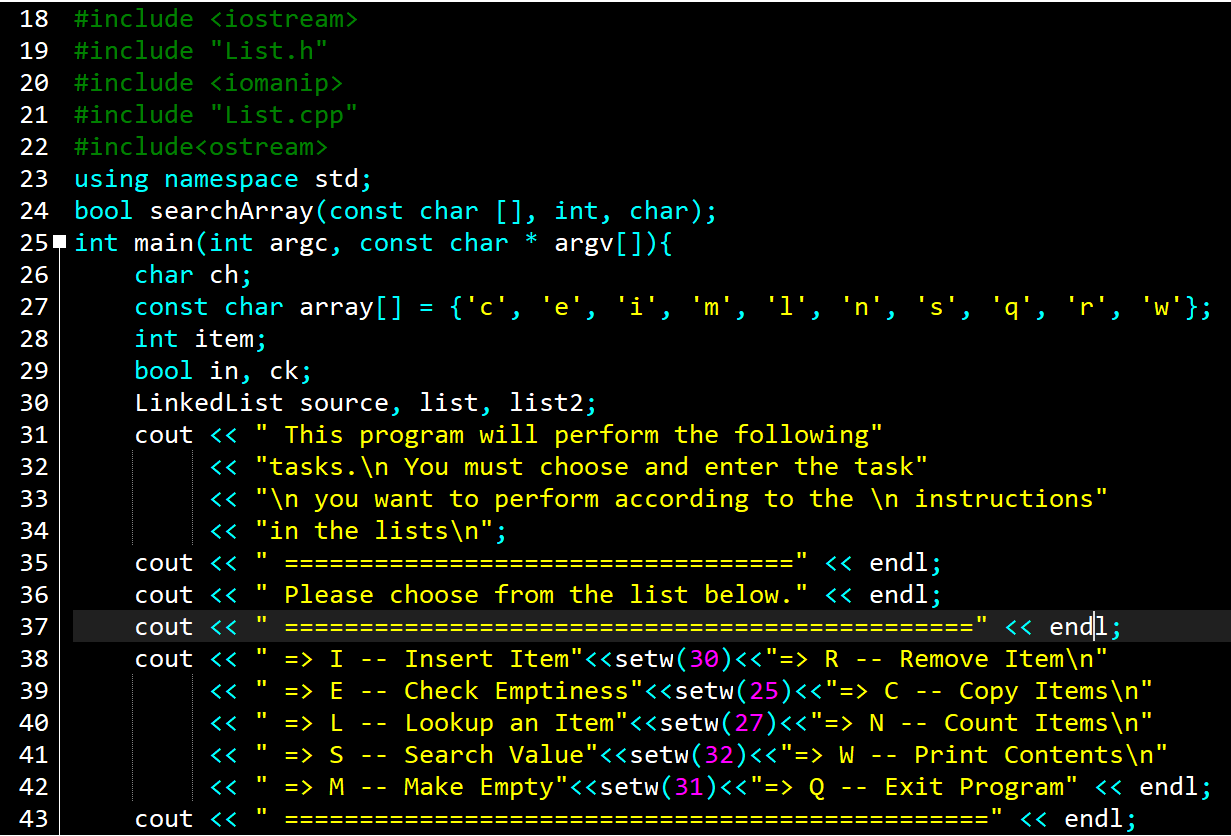


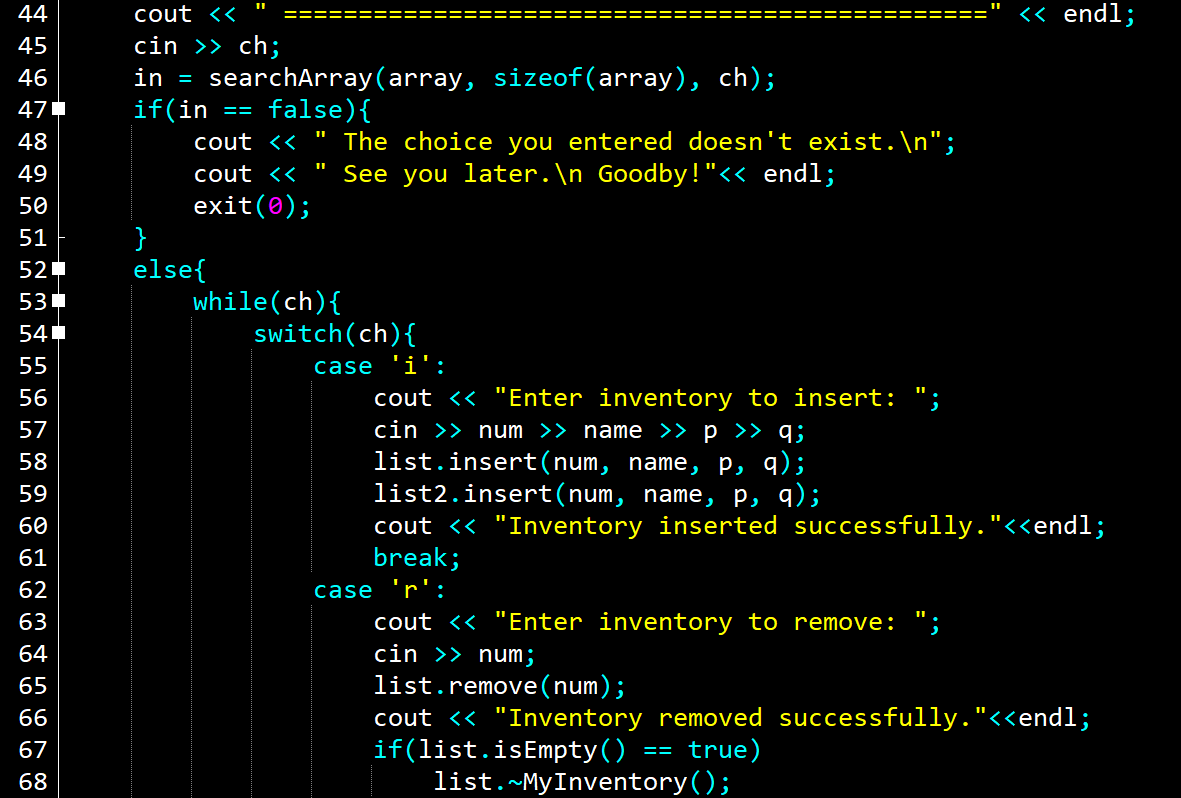


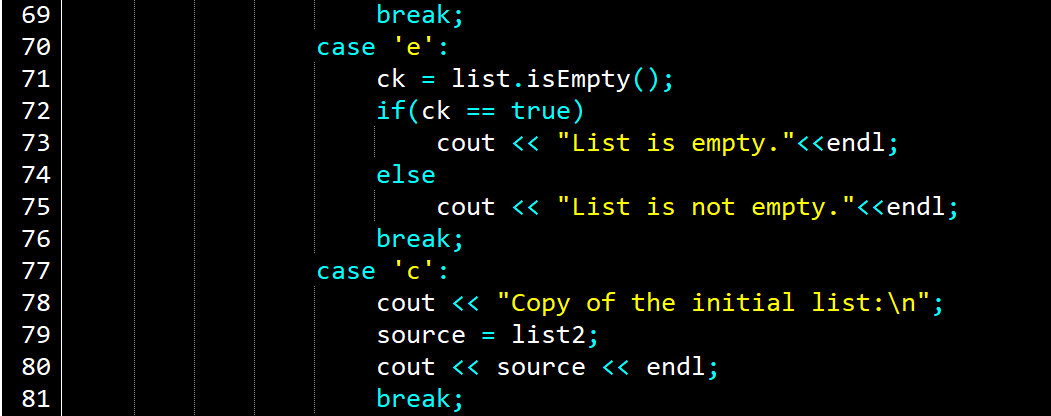


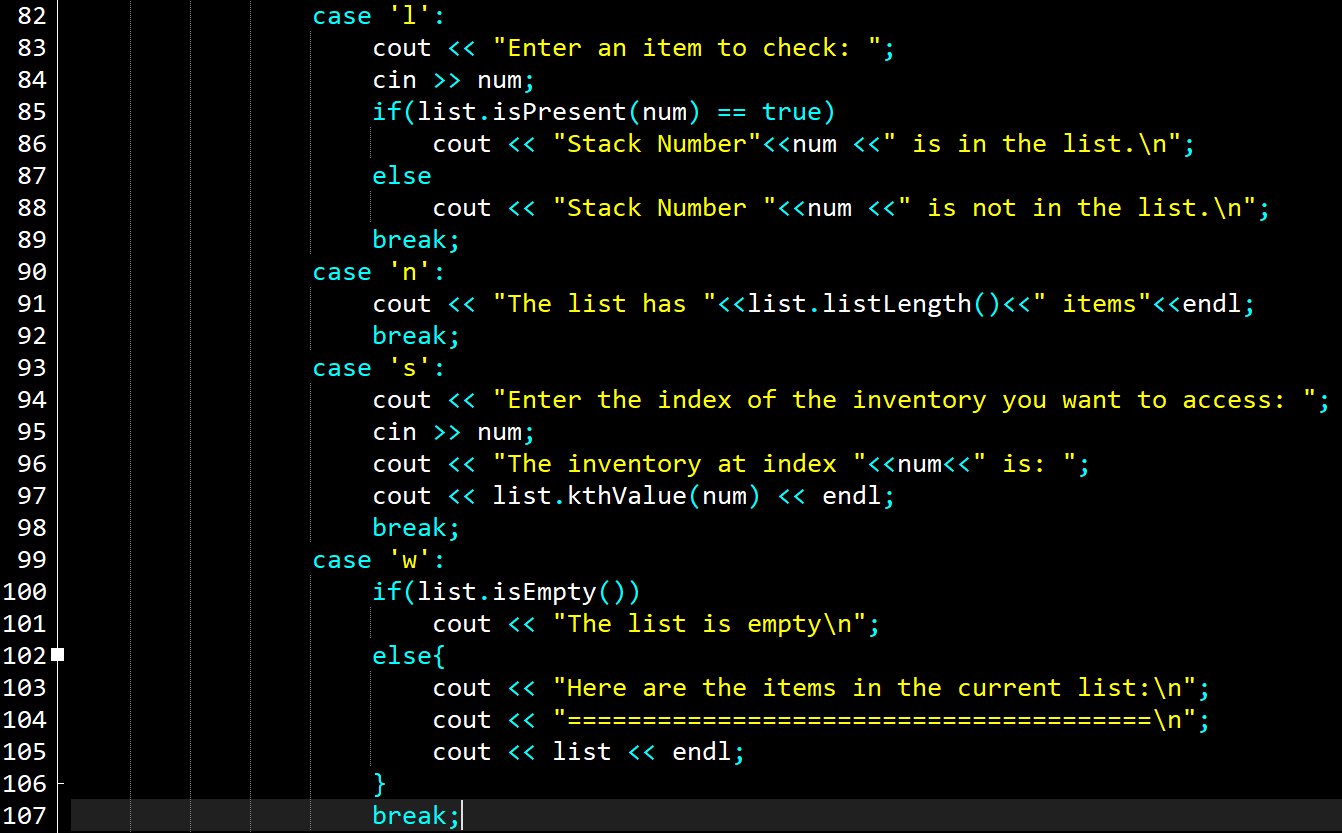
## **The testing file (main.cpp)**

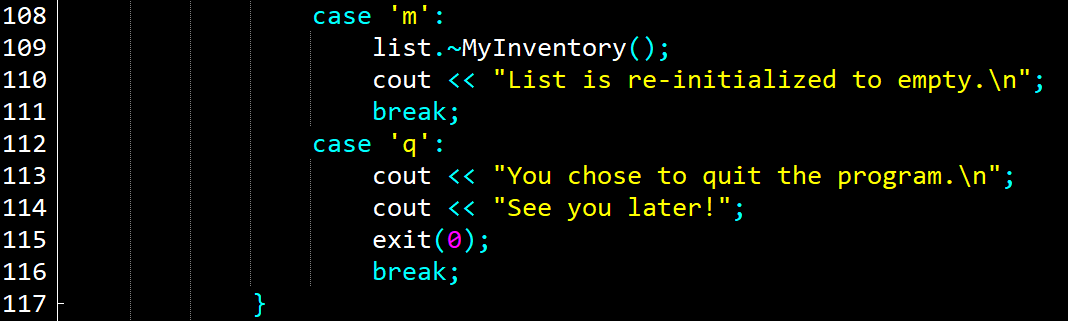


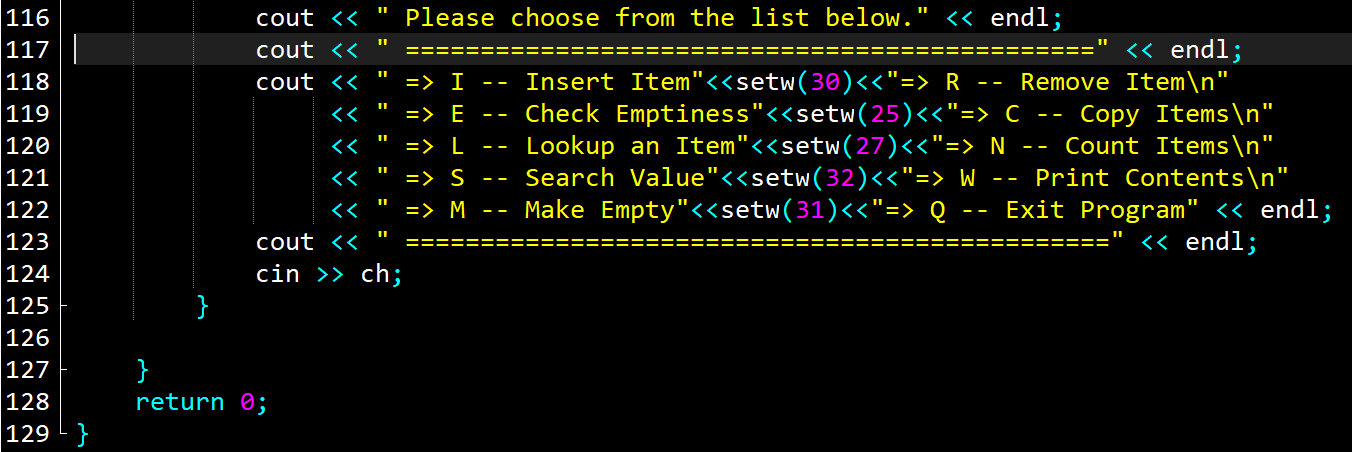


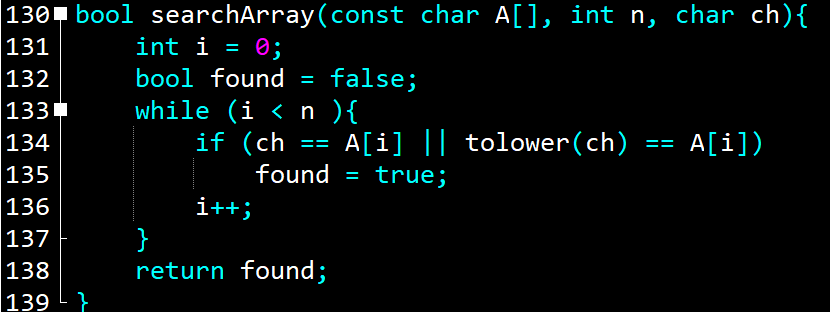






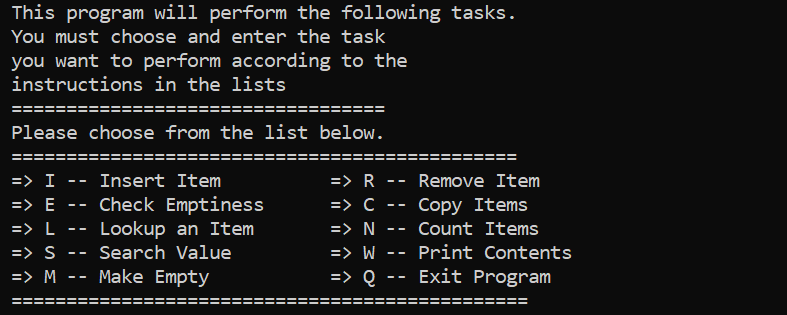


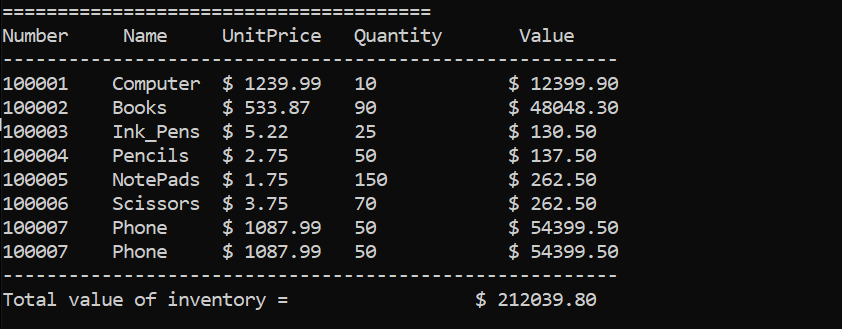




# **Test Results**

## **Tests result for inventery file read in.**

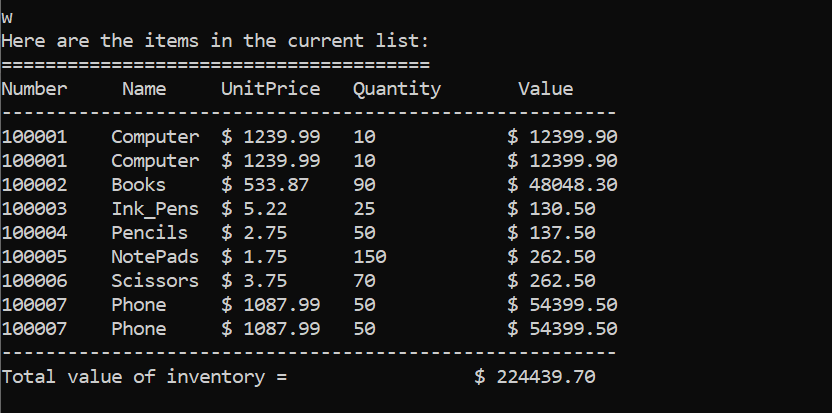




## **Tests result for insert() function.**

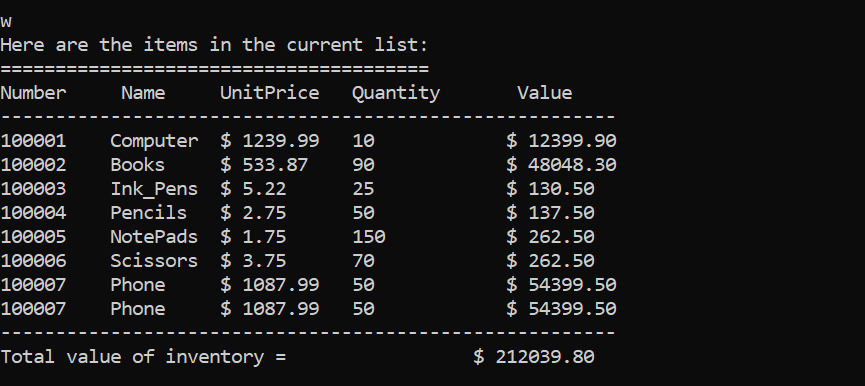
* The second inventor below was inserted.



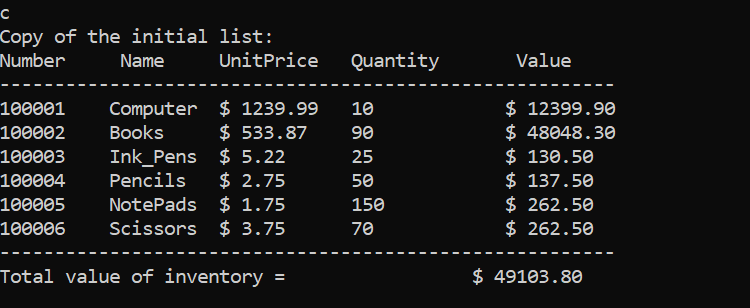


## **Tests result for remove() function.**

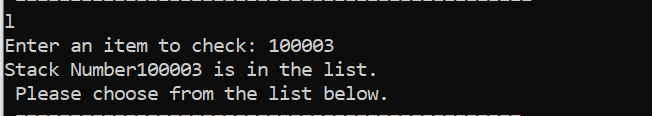
* Let us remove inventory **100001** now.

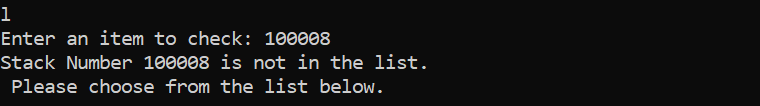


## **Tests result for copy () function.**

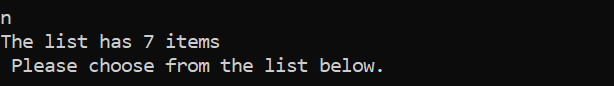


## **Tests result for isPresent() function.**

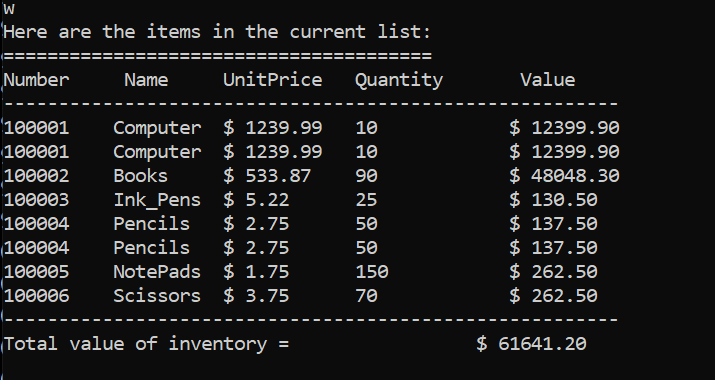


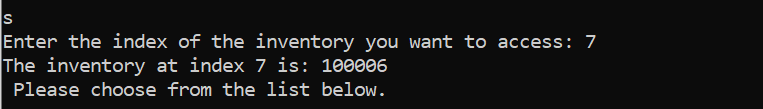


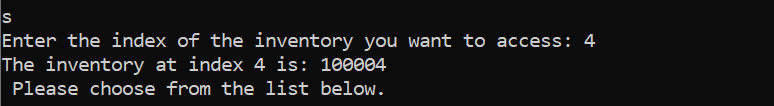
## **Tests result for listLength() function.**



## **Tests result for kthValue() function.**





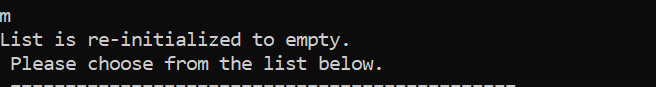


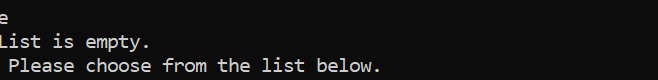
***Note***: If we try to access an index that is not in the list the function will exhaust searching that index and exit the program as shown in the above screenshot.

## **Tests result for write () function**

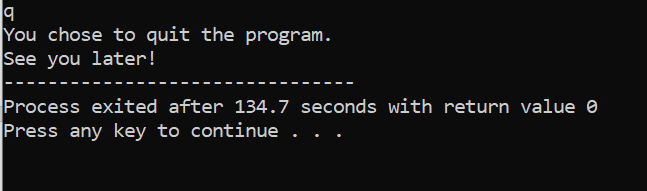
I am not running a separate test for the write () function since I am running it for almost every other step as part of checking the program to see if it is doing what it supposed to do. Please look at the end of the other programs where I occasionally run the write() function.

## **Tests result for makeEmpty () function**





## **Tests result for quit**



Moreover, if the user enters a choice that is not listed the program will announce that and exit as shown below.

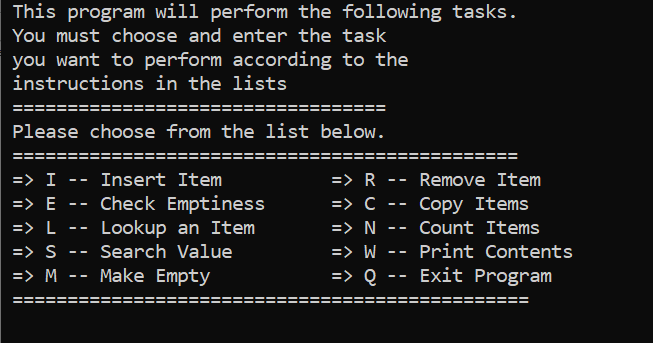
# **User document**

This program can perform different tasks on a linked list as shown in the menu below. In order to run the program, you must perform the following steps.

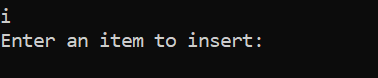
* The program name is main.cpp. on the terminal enter the following command to compile and run the program.

g++ -o main main.cpp

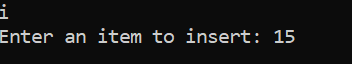
* The program will compile and open the following window:



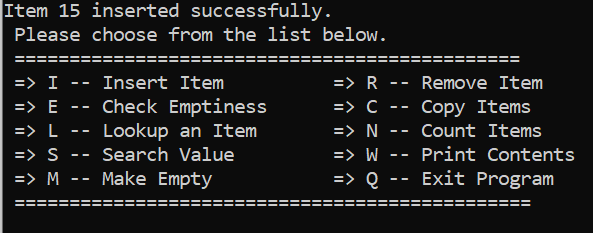
* Once the window opens, make a choice from the displayed menu. For example to insert an item type i or I and then enter.



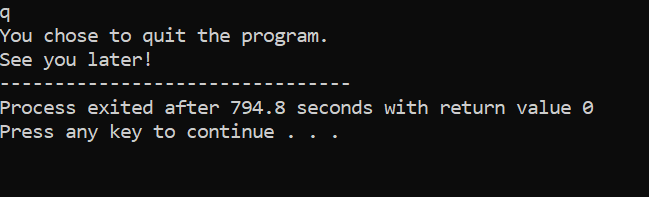
* Next, type the item you want to insert and then enter. For example, type 15 and enter.



* The program will announce that the item is entered successfully and display the menu to make the next choice.



* If you want to repeat the insert repeat the above procedure; otherwise make the next selection.
* The program will perform in the same manner for all other tasks as for insert. Hence, all the other eight functions will perform in the same manner.
* Feel free to play around with the other choices (alphabets) and see what the program is meant to do.
* If you wish to exit the program, type q (Q) and enter.



* Now you can close the window.

Here are very important points while using this program

1. You must insert integer values only. If you try to enter something else other than an integer, the program may crash.
2. Do not try to access the ends of the list. If your list has 3 nodes only and you try to access the 4th node, the program will stop and exit.
3. The program will save a copy of the current list you are working with. You can just type c(C) and access that copy. Of course, once you exit the program that copy will not exits.

# **Summery**

The project implements a linked list operation such as inserting a new item, removing an item from the list, making the list empty, checking for the presence of an item, displaying a copy of the original item that contains all the elements of the list and so on. The knowledge of linked lists and their implementation is very crucial because data are often stored and retrieved as lists.

This program can further be improved by making the necessary changes to make the program accommodate various types of data such as characters and strings. This will make the program a more useful data structure where we can store important records such as student records. Furthermore, data could also be made available as a file and a permanent copy of that file is kept with all current updates included while we still have the old data for reference.

I have gained a significant level of confidence and the necessary knowledge to work with linked lists by completing this project. I believe, this project is one of the projects I would keep working on and refer to the most even the future programming computations. [[1]](#footnote-1)

1. This materials in this document is mostly from the previous project due to the close similarity of the two projects. [↑](#footnote-ref-1)