DSA A2 Center Point

by Lim SI NI

Submission date: 31-Dec-2023 12:16AM (UTC-0800)
Submission ID: 2265784818
File name: DSA_A2_Report_Center_Point_1.pdf (677K)
Word count: 2519
Character count: 11048

FACULTY OF COMPUTING
UTM Johor Bahru

SESSION 2023/2024 SEMESTER 1

SECJ - DATA STRUCTURE AND ALGORITHMS

ASSIGNMENT 2

Group Name: Center Point

Topics: Inventory Management System

Lecturer: MDM LIZAWATI BINTI MI YUSUF

GROUP MEMBERS:

No.	Name	Matric No.
1	LIM SI NI	A22EC0070
2	ONG KAI XUEN	A22EC0100
3	SOH FEI ZHEN	A22EC0272

Table of Content

1. Objective

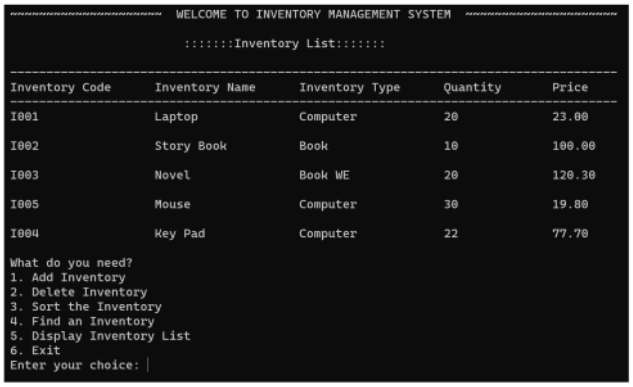
The assignment 2 is designed to implement the linked list concept into the inventory management system. In this system, we get initial inventory information from a file, and keep it in the linked list data structure, and finally store the modified linked list into file.

To use this system, user can modify the inventory information by
-adding new item into inventory
-delete item from inventory list
-search items in inventory list
-sort the inventory list
-display the entire inventory list

2. Synopsis

The implementation of linked list concept into the inventory system is a strong and also fast way to manage things like stocks or goods. It helps organize data in a convenient method. The main features of the tool are adding new things to stock, removing items, looking for inventory items that users need. Sorting list and display inventory list are also convenient functions for this system.

In the menu and welcome page, user is asked to choose the way to modify the inventory such as add, delete, sort, search, display the inventory list or exit the system.



I'll briefly explain how each function worked in the inventory system.

Feature 1: Adding Inventory

```
Please choose where to add the new inventory:
[1] Add to the front
[2] Add to the middle
[3] Add to the end
[4] Exit
Enter your choice:
```

After choosing the Add Inventory feature, users can then decide to add the inventory at front, middle or the end of the list.

When the user chooses to add new inventory in the list's middle, they will be asked for info about this item. Next, since the user chooses to insert in the middle, the user will need to enter the position of the list.

If the user entered position is invalid, such as the position is larger than the existing number of inventory in the list, the new inventory will automatically insert at the end.

```
----------New inventory Info----------
Enter Inventory Code: A333
Enter Inventory Name: Speaker
Enter Inventory Type: Accessories
Enter Quantity: 4
Enter Price: 123.90

***If the position entered is invalid, it'll automatically insert at the end.***
Enter the position to add in the middle:
```

A message and modified list will be pop out to remind user that the item has been successfully added in the list after they enter the position.

```
Enter the position to add in the middle: 3
New node has been successfully added to position 3.
```

Feature 2: Delete Inventory

After choosing the Delete Inventory feature, user can decide to delete the inventory at front, middle or the end of the list.

```
Please choose where to delete the inventory item:
[1] Delete at the front
[2] Delete at the middle
[3] Delete at the end
[4] Exit
Enter your choice:
```

When the user chooses to add new inventory in the list's middle, they will be asked for info about this item. Next, since the user choose to insert in the middle, user will need to enter the position of the list.

```
Pls enter the position of inventory node you wish to delete: (eg. 1,2,3,...)
3
The current inventory list after your delete operation be like:
                      :::::::Inventory List:::::::

--------------------------------------------------------------------------------
Inventory Code      Inventory Name      Inventory Type      Quantity       Price
--------------------------------------------------------------------------------
I001                Laptop              Computer            20             23.00

I002                Story Book          Book                10             100.00

I005                Mouse               Computer            30             19.80
```

The system will automatically display the current inventory list after deleting.

Feature 3: Sort Inventory

The system will automatically sort the Inventory Code in ascending when user choose to sort the inventory list.

```
Inventory List is sorted by Inventory Code in Ascending
                      :::::::Inventory List:::::::

-------------------------------------------------------------------------------
Inventory Code      Inventory Name      Inventory Type      Quantity       Price
-------------------------------------------------------------------------------
I001                Laptop              Computer            20             23.00

I002                Story Book          Book                10             100.00

I005                Mouse               Computer            30             19.80
```

Feature 4: Find an Inventory

When users choose to find an inventory, they will ask whether they want to find the information of inventory in code or in name.

```
Do you want to find the inventory details based on
[1] Inventory Code
[2] Inventory Name
[3] Exit
Option:
```

Below is the example of finding an inventory item in code.

```
--Find the Inventory--

Please enter the value of Inventory Code: I002

---Inventory Detail---
Inventory Code: I002
Inventory Name: Story Book
Inventory Type: Book
Quantity:       10
Price:          100.00
```

Feature 5: Display Inventory List

The inventory list will be displayed after the user chooses 5.

```
                    :::::::Inventory List:::::::

-------------------------------------------------------------------------------
Inventory Code      Inventory Name      Inventory Type      Quantity       Price
-------------------------------------------------------------------------------
I001                Laptop              Computer            20             23.00
I002                Story Book          Book                10             100.00
I003                Novel               Book WE             20             120.30
I005                Mouse               Computer            30             19.80
I004                Key Pad             Computer            22             77.70
```

3. System Design

3.1. Pseudocode

3.1.1. Main function

1. Start
2. Set loop = true
3. Open input file
4. Read input file to get inventory item details (code, name, type, quantity, price)
5. Update inventory item details into inventory linked list
6. Close input file
7. Print linked list
8. While(loop)
    - 8.1. Print main menu
    - 8.2. Get menuChoice
    - 8.3. If(menuChoice==1)
        - 8.3.1. Adding new inventory function
    - 8.4. Else if(menuChoice==2)
        - 8.4.1. Delete inventory function
    - 8.5. Else if(menuChoice==3)
        - 8.5.1. Sort the inventory list function
    - 8.6. Else if(menuChoice==4)
        - 8.6.1. Find an inventory based on search key function
    - 8.7. Else if(menuChoice==5)
        - 8.7.1. Display inventory list function
    - 8.8. Else if(menuChoice==6)
        - 8.8.1. loop=false
    - 8.9. Else
        - 8.9.1. Print "Please enter a valid choice! :)”
9. Open output file
10. Store linked list data into output file
11. End

3.1.2. Adding new inventory function

1. Start
2. Print adding menu
3. Get adding option
4. If adding option ==1 //add front
   4.1 if the list is empty
      4.1.1 set head==NULL
   4.2 Else
      4.2.1 set newInventory->next=head
      4.2.2 set head=newInventory
   4.3 Display successful message
5. Else if adding option==2 //add middle
   5.1 Get the position of inventory node user wish to add (pos)
      5.1.2 if pos==1 || the list is empty
         5.1.2.1 Go to step 4
      5.1.3 Else
         5.1.2.1 Set temp = head, count=1, found =false
         5.1.2.2 while temp->next!=NULL && count<pos-1 &&!found
            5.1.2.2.1 set temp=temp->next
            5.1.2.2.2 count++
         5.1.2.3 if ! found
            5.1.2.3.1 set newInventory->next = temp->next
            5.1.2.3.2 set temp->next = newInventory
            5.1.2.3.3 set found=true
   5.2 Display successful message
6. Else if adding option==3 //add end
   6.1 if the list is empty
      6.1.1 set head=newInventory
   6.2 Else
      6.2.1 While temp->next !=NULL

6.2.1.1 set temp=temp->next

6.2.2 set newInventory->next=NULL

6.2.3 set temp->next=newInventory

6.3 Display successful message

7. Else

7.1 Go to step 8 of main function

8. Print the inventory list after adding operation
9. End

3.1.3. Delete inventory function

1. Start
2. If list is empty
    2.1. Print “Error: The inventory list is empty. Cannot delete any inventory anymore”
    2.2. Go to step 8 of main function
3. Print delete menu
4. Get delete option
5. If delete option == 1 //delete front
    5.1. Set temp=head;
    5.2. head = head->next;
    5.3. delete temp;
6. Else if delete option ==2 //delete middle
    6.1. Get the position of inventory node user wish to delete (delPos)
    6.2. If (delPos==1)
        6.2.1. Go to step 5.1
    6.3. Else
        6.3.1. temp = head
        6.3.2. for(i=0; i<(pos-1)&&temp;i++)
            6.3.2.1. temp=temp->next;
        6.3.3. if(!temp)
            6.3.3.1. Print “Error: Your entered position of node that to be deleted is out of the range of current list.”
        6.3.4. Else if (temp->next->next==NULL) //delete back
            6.3.4.1. delete temp->next;
            6.3.4.2. temp->next=NULL;
        6.3.5. Else
            6.3.5.1. prev=temp;
            6.3.5.2. temp=temp->next

6.3.5.3. prev-next=temp->next

6.3.5.4. delete temp

7. Else if delete option ==3 //delete end

7.1. temp=head;

7.2. while(temp->next->next != NULL)

7.2.1. temp=temp->next

7.3. delete temp->next

7.4. temp->next=NULL

8. Else

8.1. Go to step 8 of main function

9. Print the inventory list after delete operation

10. End

3.1.4. Sort the inventory list function

1. Start
2. If (head == NULL || head->next == NULL)
    2.1. Go to step 8 of main function
3. Set sortedList = NULL
4. Set curr = head
5. While (curr)
    5.1. Set next = curr->next
    5.2. If (sortedList == NULL || curr->getCode() < sortedList->getCode())
        5.2.1. curr->next = sortedList
        5.2.2. sortedList = curr
    5.3. Else
        5.3.1. Set temp = sortedList
        5.3.2. While (temp->next != NULL && temp->next->getCode < curr->getCode())
            5.3.2.1. temp = temp->next
        5.3.3. curr->next = temp->next
        5.3.4. temp->next = curr
    5.4. curr = next
6. head = sortedList
7. Print "Inventory List is sorted by Inventory Code in Ascending"
8. Display sorted list
9. End

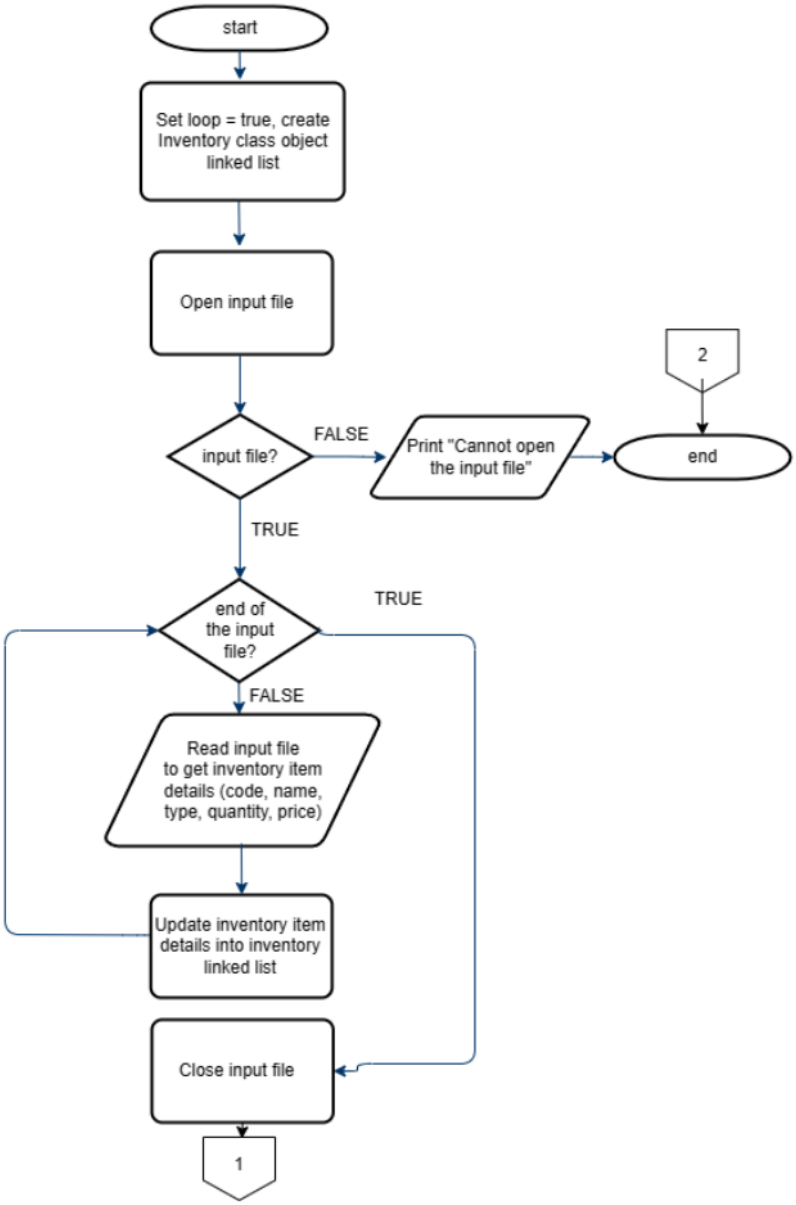
3.1.5. Find an inventory based on search key function

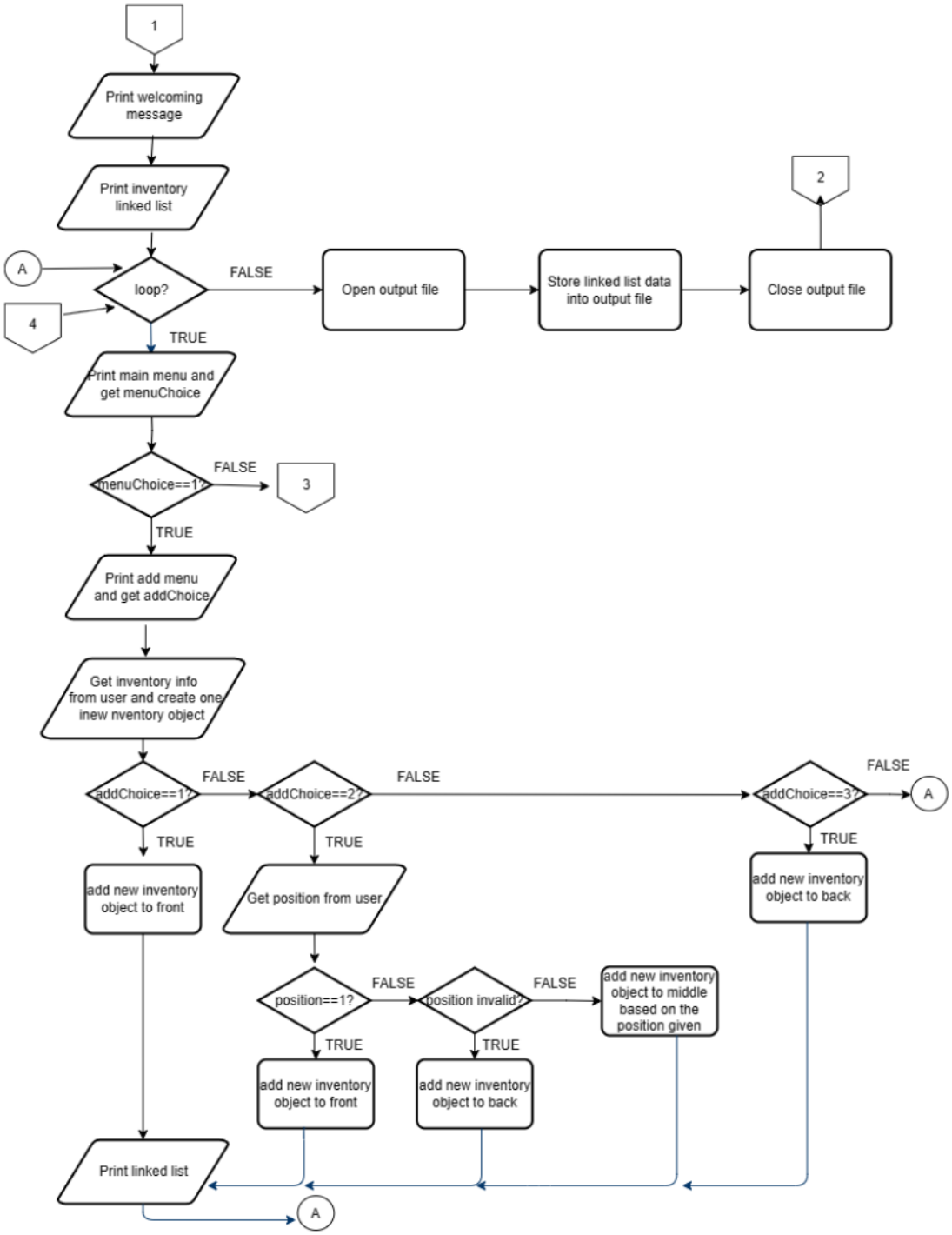
1. Start
2. Set temp = head
3. Set found = false
4. Print finding user menu
5. Get menuChoice
6. If (menuChoice == 1)
    - 6.1. Print finding based on inventory code page
    - 6.2. Get user input inventoryCode
    - 6.3. Convert the input inventoryCode to uppercase
    - 6.4. While (!found && temp->next != NULL)
        - 6.4.1. If (temp->getCode() == inventoryCode)
            - 6.4.1.1. found = true
            - 6.4.1.2. Print the inventory detail
        - 6.4.2. temp = temp->next
    - 6.5. Go to step 8 of main function
7. Else if(menuChoice == 2)
    - 7.1. Print finding based on inventory name page
    - 7.2. Get user input inventoryName
    - 7.3. Convert the input inventoryName to uppercase
    - 7.4. While (!found && temp->next != NULL)
        - 7.4.1. If (temp->getName() == inventoryName)
            - 7.4.1.1. found = true
            - 7.4.1.2. Print the inventory detail
        - 7.4.2. temp = temp->next
    - 7.5. Go to step 8 of main function
8. Else
    - 8.1. Go to step 8 of main function
9. If(!found)
    - 9.1. Print “The entered value is invalid!”
10. End

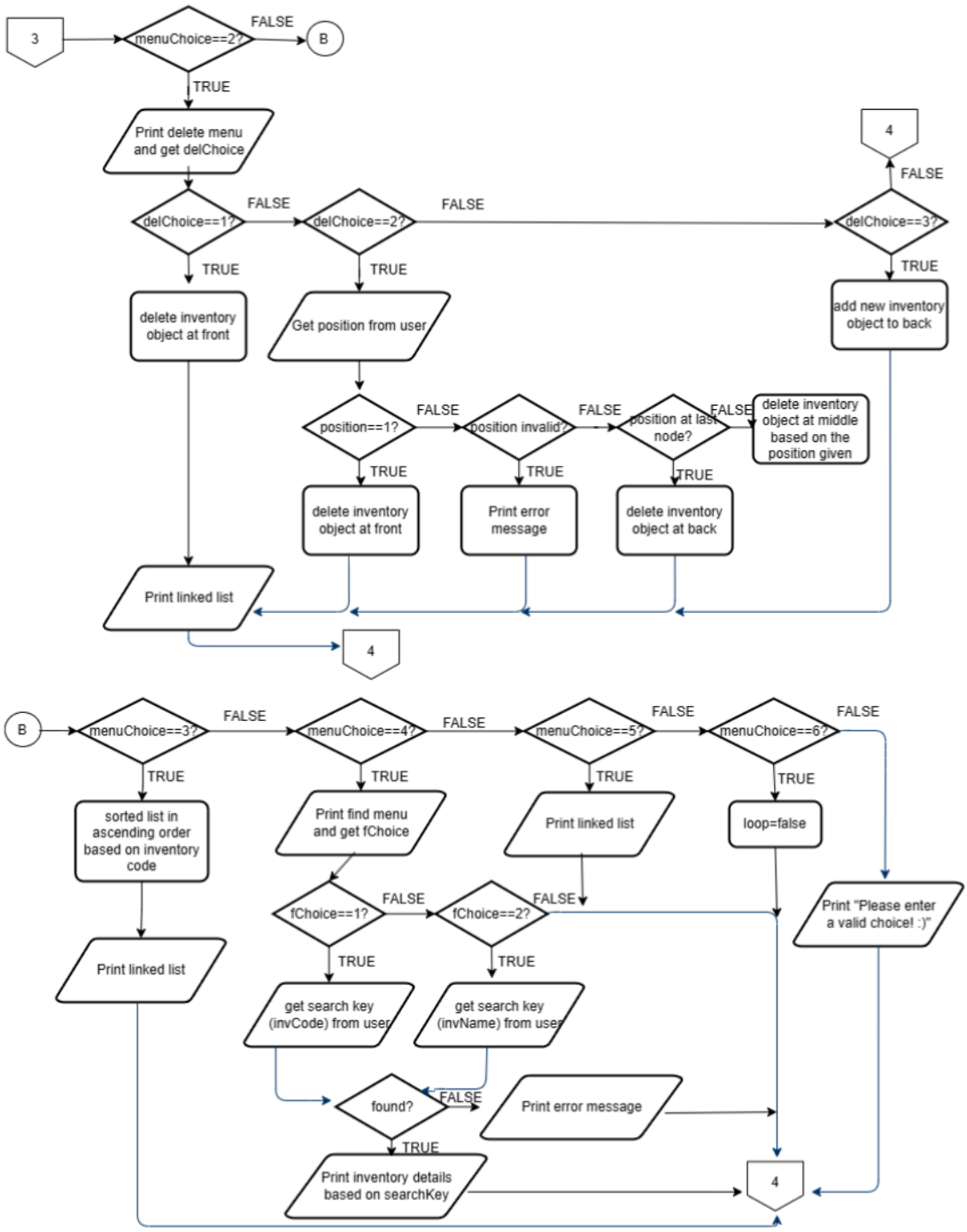
3.1.6. Display inventory list function

1. Start
2. set temp=head
3. Display “Inventory Code”
4. Display“Inventory Name”
5. Display “Inventory Type”
6. Display “ Quantity”
7. Display “Price”
8. While (temp!=NULL)
   2.1 Display Inventory Code (temp->getCode)
   2.2 Display Inventory Name (temp->getName)
   2.3 Display Inventory Type (temp->getType)
   2.4 Display Quantity (temp->getQuantity)
   2.5 Display Price (temp->getPrice)
   2.6 temp=temp->next
9. End

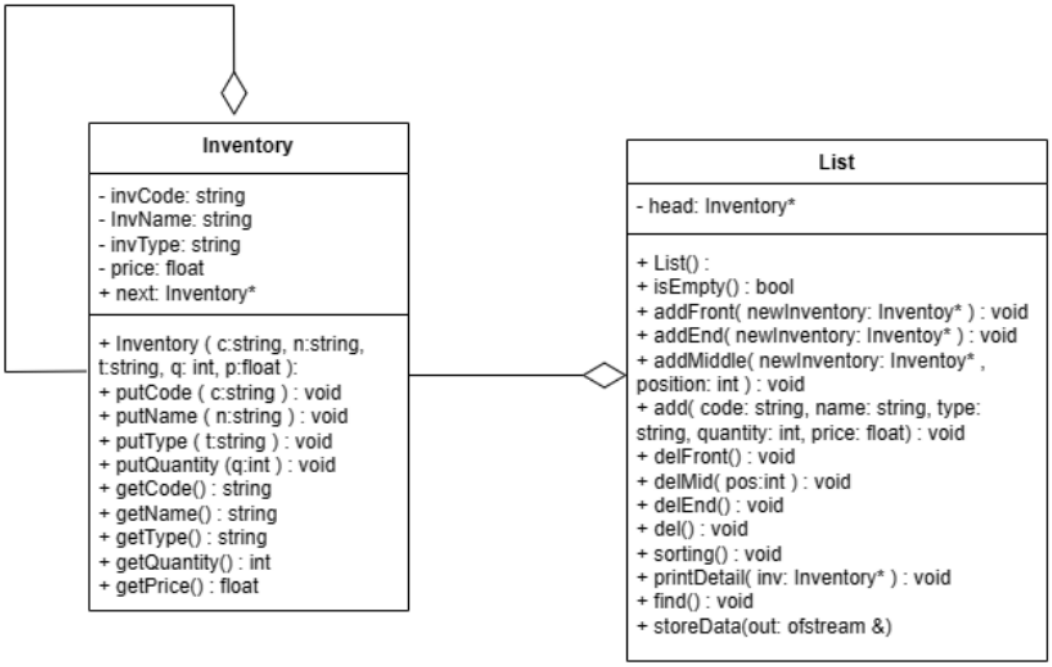
3.2. Flow Chart




1
Print welcoming message
Print inventory linked list
A
loop?
FALSE
Open output file
Store linked list data into output file
Close output file
2
4
TRUE
Print main menu and get menuChoice
menuChoice==1?
FALSE
3
TRUE
Print add menu and get addChoice
Get inventory info from user and create one inew nventory object
addChoice==1?
FALSE
addChoice==2?
FALSE
addChoice==3?
FALSE
A
TRUE
TRUE
TRUE
add new inventory object to front
Get position from user
add new inventory object to back
position==1?
FALSE
position invalid?
FALSE
add new inventory object to middle based on the position given
TRUE
TRUE
add new inventory object to front
add new inventory object to back
Print linked list
A


3
menuChoice==2?
FALSE
B
TRUE
Print delete menu and get delChoice
delChoice==1?
FALSE
delChoice==2?
FALSE
delChoice==3?
FALSE
4
FALSE
TRUE
TRUE
TRUE
delete inventory object at front
Get position from user
add new inventory object to back
position==1?
FALSE
position invalid?
FALSE
position at last node?
FALSE
delete inventory object at middle based on the position given
TRUE
TRUE
TRUE
delete inventory object at front
Print error message
delete inventory object at back
Print linked list
4
B
menuChoice==3?
FALSE
menuChoice==4?
FALSE
menuChoice==5?
FALSE
menuChoice==6?
FALSE
TRUE
TRUE
TRUE
TRUE
sorted list in ascending order based on inventory code
Print find menu and get fChoice
Print linked list
loop=false
Print "Please enter a valid choice! :)"
fChoice==1?
FALSE
fChoice==2?
FALSE
Print linked list
TRUE
TRUE
get search key (invCode) from user
get search key (invName) from user
found?
FALSE
Print error message
TRUE
Print inventory details based on searchKey
4

3.3. Class Diagram



4. Data Structure Operation Description

4.1. Linked List Implementation

In this case, we declare a class named “Inventory” which acts as a node and a class named “List” to store nodes. Both of these classes will function as a linked list.

In tha main function, we declare an List object named InvList which purpose is to store all the inventory nodes. At first, we retrieve data from “input.txt” file and store it into the invList by using addEnd() function where it will link the new node (inventory) at the end of the linked list (InvList).

In this program, we have 5 functions and the first function is adding new inventory where users can choose to add inventory in front, in the specific position or at the end of the InvList.

Adding Function

addFront()  ``` void addFront(Inventory *newInventory) { if (isEmpty()) { head = newInventory; } else { newInventory->next = head; head = newInventory; } } ```	• If the list is empty, set head = newInventory • Else, newInventory->next = head->next and head = newInventory
addEnd()  ``` void addEnd(Inventory *newInventory) { Inventory *temp = head; if (isEmpty()) head = newInventory; else { while (temp->next != NULL) { temp = temp->next; } newInventory->next = NULL; temp->next = newInventory; } } ```	• If the list is empty, set head = newInventory • Else, loop until find the last node in the linked list, set newInventory->next = NULL and insert the newInventory after the last node
addMiddle()	• If the list is empty, set head = newInventory • If (position == 1), addFront() • while(NO achieve the last node in the list

	AND count < position-1 AND NO found), temp will become its next value and increase count by 1 • Then, the newInventory will be inserted at the position • If the entered position value is invalid, it will auto inserted at the end of list

The second function is deleting inventory function where users can choose to delete inventory in front, in the specific position or at the end of the InvList.

Deleting Function

delFront() `void delFront()` `{` `  Inventory *temp = head;` `  head = head->next;` `  delete temp;` `}`	• New head will become the next node of head • And the previous head is deleted
delMid() 	• If the position is 1, delFront() • Else, loop and find the node which position is smaller than pos-1 • If the node found is second last node, the last node will be deleted • Else, the node with the pos position will be deleted
delEnd() `void delEnd()` `{` `  Inventory *temp = head;` `  while (temp->next->next != NULL)` `  temp = temp->next;` `  delete temp->next;` `  temp->next = NULL;` `}`	• Loop the node until find the second last node • Modify the new last node becomes the second last node • Delete the previous last node

The third function is sorting where it will sort the invList based on Inventory Code in ascending order.

If the list is empty, it will quit the sorting

```
    if (head == NULL || head->next == NULL)
    {
        return;
    }
```

Else, it will compare the inventory code value from sortedList and curr

- If the inventory code from curr is smaller than inventory code from sorted list, inventory code from curr will be stored in front of inventory code from sorted list
- Else, set a temp = sortedList and start looping with condition (NO achieve the end of list AND temp->next inventory Code value is smaller than the inventory code from curr), temp will go to next value. Until out of looping, store the inventory code from curr at the next of the inventory code of temp
- Then, move the curr to be curr->next value
- And lastly update the head with sortedList

```
Inventory *sortedList = NULL;
Inventory *curr = head;

while (curr)
{
    Inventory *next = curr->next;

    if (sortedList == NULL || curr->getCode() < sortedList->getCode())
    {
        curr->next = sortedList;
        sortedList = curr;
    }
    else
    {
        Inventory *temp = sortedList;
        while (temp->next != NULL && temp->next->getCode() < curr->getCode())
        {
            temp = temp->next;
        }
        curr->next = temp->next;
        temp->next = curr;
    }
    curr = next;
}

head = sortedList;
```

Finally, display the sorted InvList

```
    cout << "Inventory List is sorted by Inventory Code in Ascending\n";
    displayList();
}
```

The fourth function is a finding function where users can find the inventory based on inventory code or inventory name.

It will loop the list since it has not achieved the last node in the list and show the details of the inventory if the value of temp->getCode() is the same as the sKey that is input by users.

```
if (fChoice == 1)
{
    cout << "Inventory Code: ";
    cin >> sKey;
    sKey = changeToUpper(sKey);
    while (temp->next != NULL && found == false)
    {
        if (temp->getCode() == sKey)
        {
            found = true;
            printDetail(temp);
        }
        temp = temp->next;
    }
}
```

The fifth function is displayList function which will display all the inventory node in the InvList linked list.

It will loop the list since it has not achieved the last node in the list and display the data by calling the accessor of the inventory node.

```
Inventory *temp = head;
while (temp != NULL)
{
    cout << left << setw(20) << temp->getCode()
         << setw(20) << temp->getName()
         << setw(20) << temp->getType()
         << setw(15) << temp->getQuantity()
         << setw(10) << fixed << setprecision(2) << temp->getPrice() << endl
         << endl;

    temp = temp->next;
}
```

5. Conclusion

In conclusion, a pointer linked list is a useful method to store data. It is flexible without limited size as an array so that we don't waste the memory space. When we apply a pointer link list concept in our system, it provides more freedom to users in managing the data in the Inventory Management System.

In our system, users can choose to add new inventory in front, at the specific position or at the end of the inventory list. It is going the same as deleting the inventory. Besides that, it also helps to sort the list based on the inventory code in ascending order to provide an easier view to users. At the same time, it also allows users to search certain inventory based on their inventory code or inventory name because both of these values are unique. Lastly, it also can print the existing inventory in the inventory list.

This program can be concluded that is more integrity, efficiency and usability than the previous program in Assignment 1.

DSA A2 Center Point

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		