FACULTY OF COMPUTING
UTM Johor Bahru

SESSION 2023/2024 SEMESTER 1

SECJ - DATA STRUCTURE AND ALGORITHMS

ASSIGNMENT 1

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

In our data structure and algorithm assignment 1, the main objectives of developing the Inventory management system are :

- To manage inventory in a warehouse
- To apply the sorting and searching technique in the system

2. Synopsis

The inventory management system is essential for one business or organization to track or manage its warehouse. Our inventory management system is designed to search or organize (sort) the inventory by its certain keys such as code, name, type, quantity, price in either ascending or descending order. These two methods are important because they not only save time but also offer convenience to users in managing the inventory effectively.

The menu is provided for the user. The user may select whether they want to sort the inventory list, search specific inventory, or exit the system.

```
~~~~~~~~~~~~~~~~~~~~~  WELCOME TO INVENTORY MANAGEMENT SYSTEM  ~~~~~~~~~~~~~~~~~~~~~

                        :::::::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

Do you want to
[1] Sorting
[2] Searching
[3] Exit
Option:
```

If the user chooses the sorting option, they will be provided several criteria to sort the inventory list including **code**, **name**, **type**, **quantity** and **price**. Then the user is asked to choose whether the sorting should be sorted in ascending or descending order. The system will then automatically sort the inventory list based on the chosen criteria.

```
<<< Sorting Process >>>
[1] By Inventory Code
[2] By Inventory Name
[3] By Inventory Type
[4] By Quantity
[5] By Price
Option:
```

Below is the example output of sorting by **Quantity** in **ascending order**:

```
::Sort By Quantity in Ascending Order

                    :::::::Inventory List:::::::

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

Below is the example output of sorting by **inventory type** in **descending order**:

```
::Sort By Inventory Type in Descending Order

                    :::::::Inventory List:::::::

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

Other than that, if the user selects the searching option, They will be presented with 2 choices, whether they want to search by **code** or **name**. Following this, the user then enter the desired code or the name they want to search for. The system will display the relevant information for the specific inventory item to the user.

```
<<< Searching Process >>>
[1] By Inventory Code
[2] By Inventory Name
Option:
```

Below is the example output of searching by **code "I003"** :

```
Enter your search key: I003

::Search By Inventory Code

                       :::::::Inventory List:::::::

-------------------------------------------------------------------------------
Inventory Code      Inventory Name      Inventory Type      Quantity       Price
-------------------------------------------------------------------------------
I003                Novel               Book WE             20             120.30
```

Below is the example output of searching by **name "Story Book"** :

```
Enter your search key: Story Book

::Search By Inventory Name

                       :::::::Inventory List:::::::

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

* Note that the name entered is not case sensitive

If the user choose to exit the system, the system will display a goodbye message to user

```
Bye Bye!!!
```

3. System Design

3.1. Pseudocode

3.1.1 Main function

1. Start
2. Set counter = 0, loop = true, create Inventory class object array
3. Open input and output file
4. if(! input)
    4.1 Print "Cannot open the input file"
    4.2 End
5. while(!input.eof())
    5.1 read inventory data from file
    5.2 Update inventory data to inventory object array
6. Print inventory object array
7. while(loop)
    7.1 Print mainMenu
    7.2 Get mChoice from user
    7.3 if(mChoice==1)
        7.3.1 call sort function
        7.3.2 display inventory object array
    7.4 else if(mChoice==2)
        7.4.1 call search function
    7.5 else if(mChoice==3)
        7.5.1 print “Bye Bye!!!”
        7.5.2 loop = false
    7.6 else
        7.6.1 print “Please enter a valid choice! :) ”
8. Store the inventory object array data into output file
9. Close input file and output file
10. End

3.1.2 Sort function

1. Start
2. Print sort menu
3. Get sort choice from user + exception handling
4. Print order menu
5. Get order choice from user + exception handling
6. Set sorted=false
7. for(p=1;p<n&&!sorted;++p)
    7.1 sorted=true
    7.2 for(x=0;x<n-p;++x)

7.2.1 if(sort choice ==1)
    7.2.1.1 sortType=Inventory Code
    7.2.1.2 if(oderChoice==1)
        7.2.1.2.1 if(data[x].getCode()>
            data[x+1].getCode())
            7.2.1.2.1.1 temp=data[x]
            7.2.1.2.1.2 data[x]=data[x+1]
            7.2.1.2.1.3 data[x+1] = temp
            7.2.1.2.1.4 sorted=false
    7.2.1.3 else
        7.2.1.3.1 if(data[x].getCode()<
            data[x+1].getCode())
            7.2.1.3.1.1 temp=data[x]
            7.2.1.3.1.2 data[x]=data[x+1]
            7.2.1.3.1.3 data[x+1] = temp
            7.2.1.3.1.4 sorted=false
7.2.2 else if(sort choice ==2)
    7.2.2.1 sortType=Inventory Name
    7.2.2.2 if(oderChoice==1)
        7.2.2.2.1 if(data[x].getName()>
            data[x+1].getName())
            7.2.2.2.1.1 temp=data[x]
            7.2.2.2.1.2 data[x]=data[x+1]
            7.2.2.2.1.3 data[x+1] = temp
            7.2.2.2.1.4 sorted=false
    7.2.2.3 else
        7.2.2.3.1 if(data[x].getName()<
            data[x+1].getName())
            7.2.2.3.1.1 temp=data[x]
            7.2.2.3.1.2 data[x]=data[x+1]
            7.2.2.3.1.3 data[x+1] = temp
            7.2.2.3.1.4 sorted=false
7.2.3 else if(sort choice ==3)
    7.2.3.1 sortType=Inventory Type
    7.2.3.2 if(oderChoice==1)
        7.2.3.2.1 if(data[x].getType()>
            data[x+1].getType())
            7.2.3.2.1.1 temp=data[x]
            7.2.3.2.1.2 data[x]=data[x+1]
            7.2.3.2.1.3 data[x+1] = temp
            7.2.3.2.1.4 sorted=false
    7.2.3.3 else
        7.2.3.3.1 if(data[x].getType()<
            data[x+1].getName())

7.2.3.3.1.1 temp=data[x]
7.2.3.3.1.2 data[x]=data[x+1]
7.2.3.3.1.3 data[x+1] = temp
7.2.3.3.1.4 sorted=false

7.2.4 else if(sort choice ==4)

7.2.4.1 sortType=Quantity

7.2.4.2 if(oderChoice==1)

7.2.4.2.1 if(data[x].getQuantity()>
data[x+1].getQuantity())
7.2.4.2.1.1 temp=data[x]
7.2.4.2.1.2 data[x]=data[x+1]
7.2.4.2.1.3 data[x+1] = temp
7.2.4.2.1.4 sorted=false

7.2.4.3 else

7.2.4.3.1 if(data[x].getQuantity()<
data[x+1].getQuantity())
7.2.4.3.1.1 temp=data[x]
7.2.4.3.1.2 data[x]=data[x+1]
7.2.4.3.1.3 data[x+1] = temp
7.2.4.3.1.4 sorted=false

7.2.5 else

7.2.5.1 sortType=Price

7.2.5.2 if(oderChoice==1)

7.2.5.2.1 if(data[x].getPrice()>
data[x+1].getPrice())
7.2.5.2.1.1 temp=data[x]
7.2.5.2.1.2 data[x]=data[x+1]
7.2.5.2.1.3 data[x+1] = temp
7.2.5.2.1.4 sorted=false

7.2.5.3 else

7.2.5.3.1 if(data[x].getPrice()<
data[x+1].getPrice())
7.2.5.3.1.1 temp=data[x]
7.2.5.3.1.2 data[x]=data[x+1]
7.2.5.3.1.3 data[x+1] = temp
7.2.5.3.1.4 sorted=false

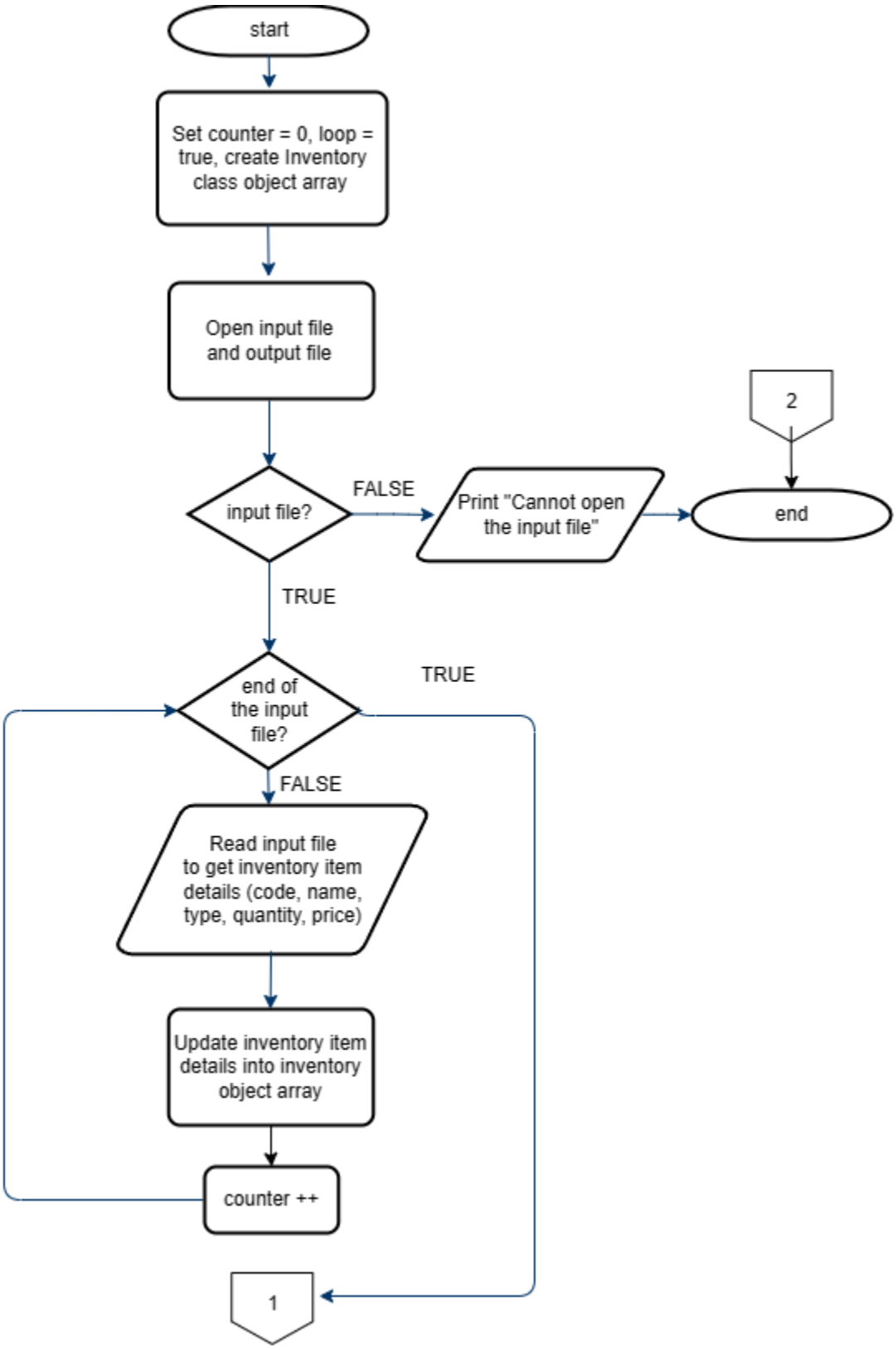
8. End

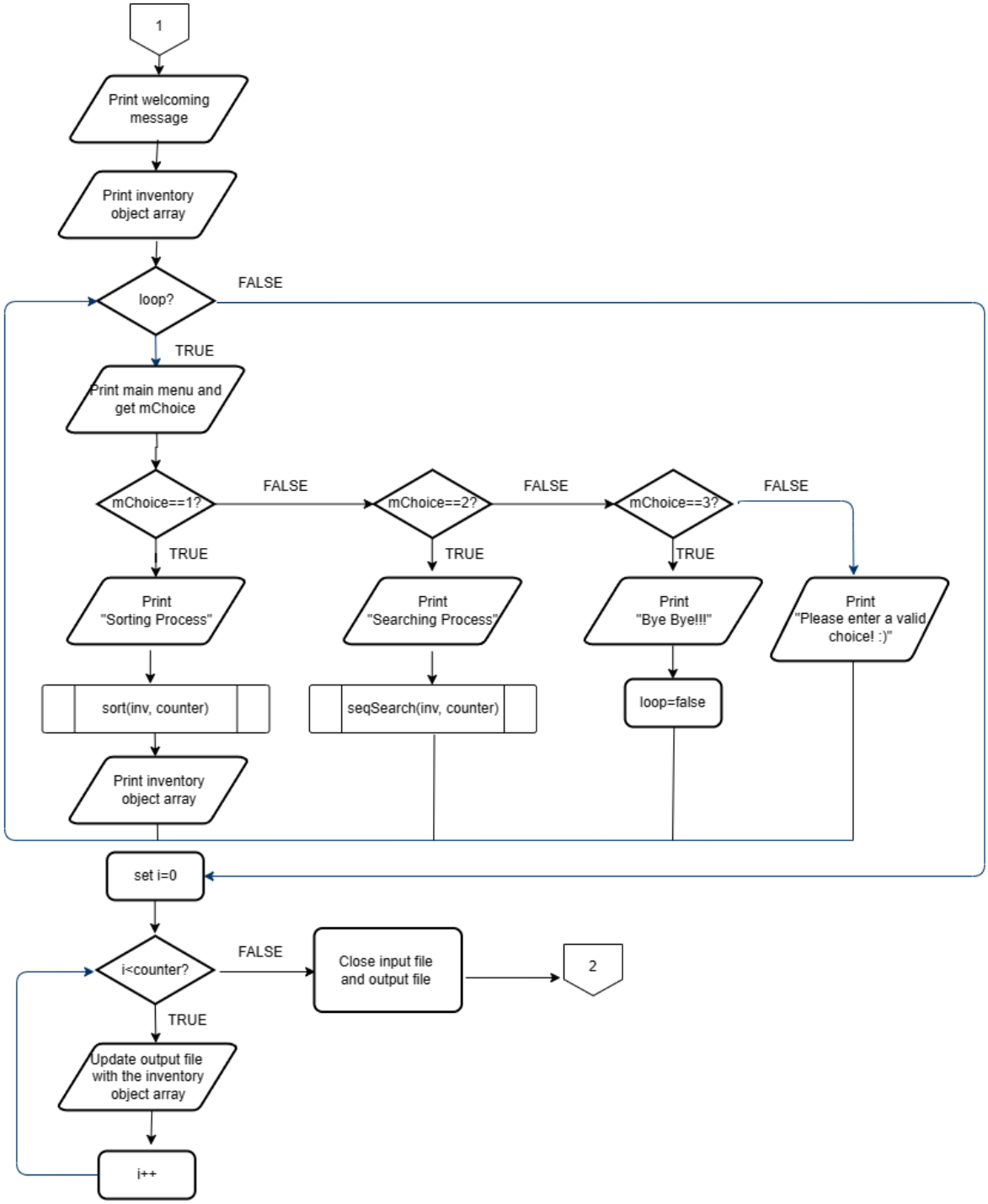
3.1.3 Search function

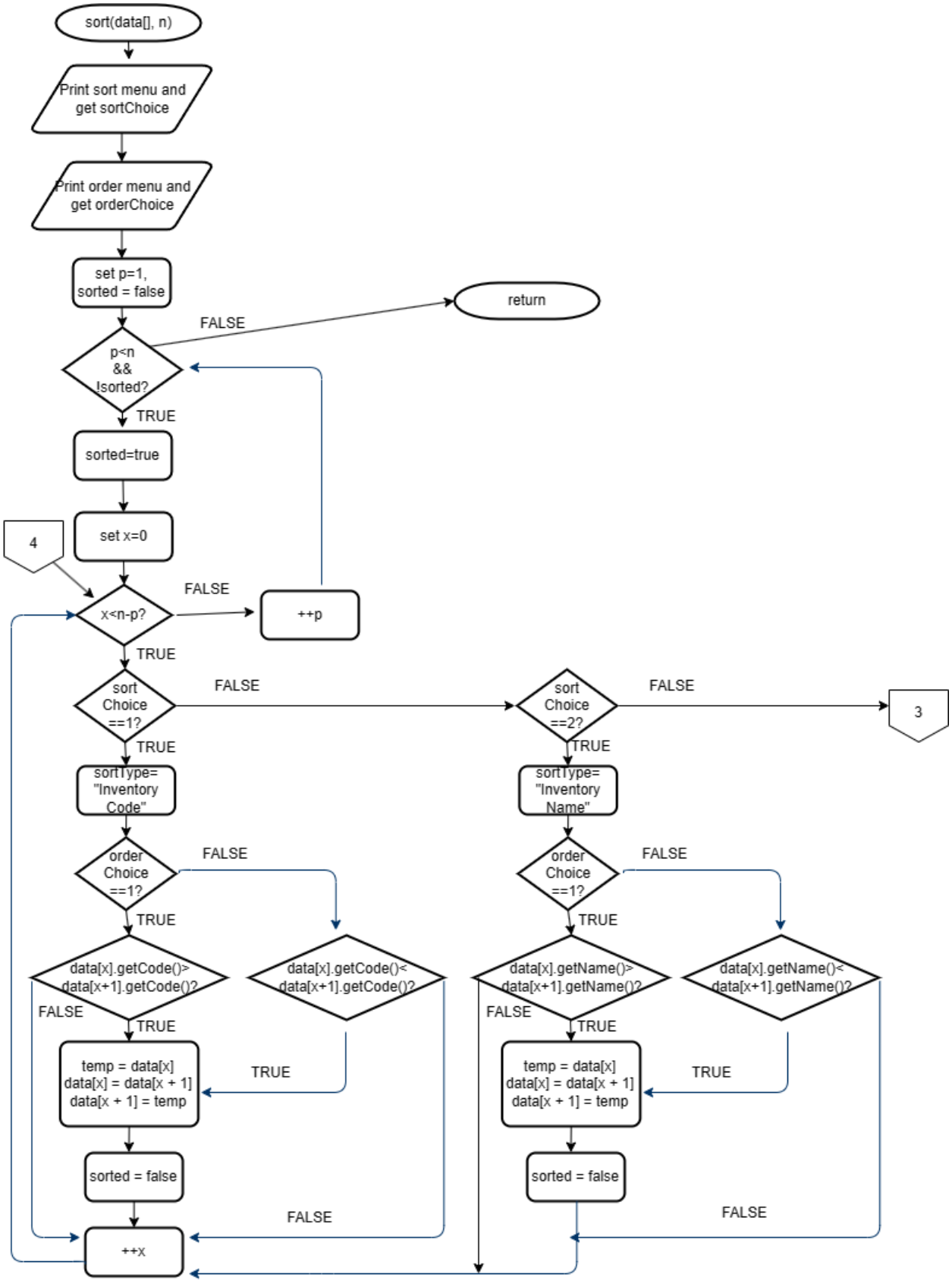
1. Start
2. set found = false, index=-1
3. Print search menu
4. Get search choice from user + exception handling

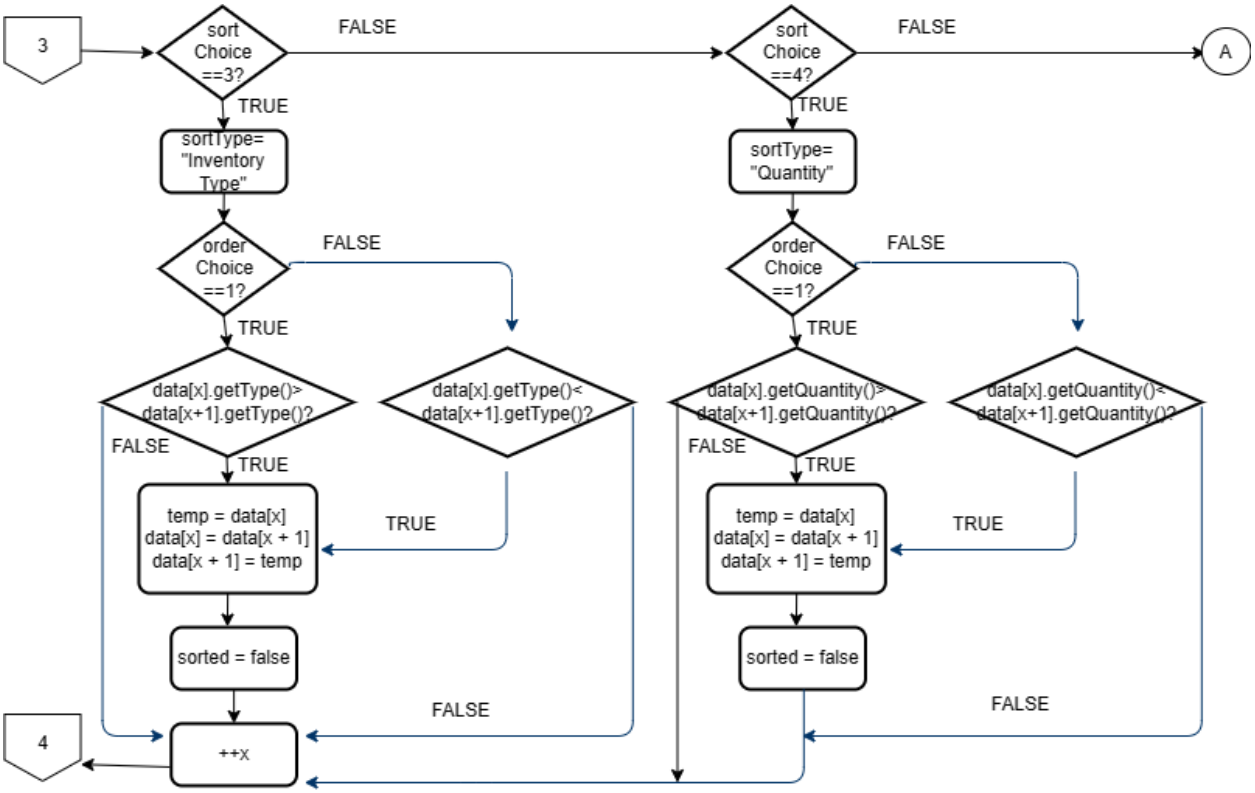
5. Get search key from user
6. if(search choice ==1)
    6.1 searchType=Inventory Code
    6.2 for(i=0;i<n;i++)
        6.2.1 if data[i].getCode()== searchKey
            6.2.2 index=i
            6.2.3 found=true
7. Else
    7.1 searchType=Inventory Name
    7.2 for(i=0;i<n;i++)
        7.2.1 if data[i].getName()== searchKey
            7.2.2 index=i
            7.2.3 found=true
8. If(found)
    8.1 Print data[index]
9. Else
    9.1 Print “Cannot Find!!!”
10. End

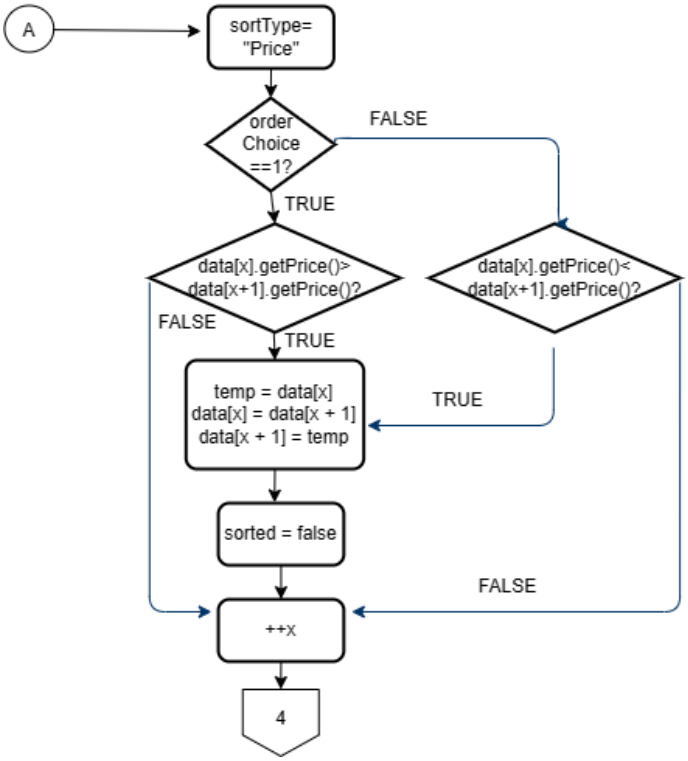
3.2. Flow Chart

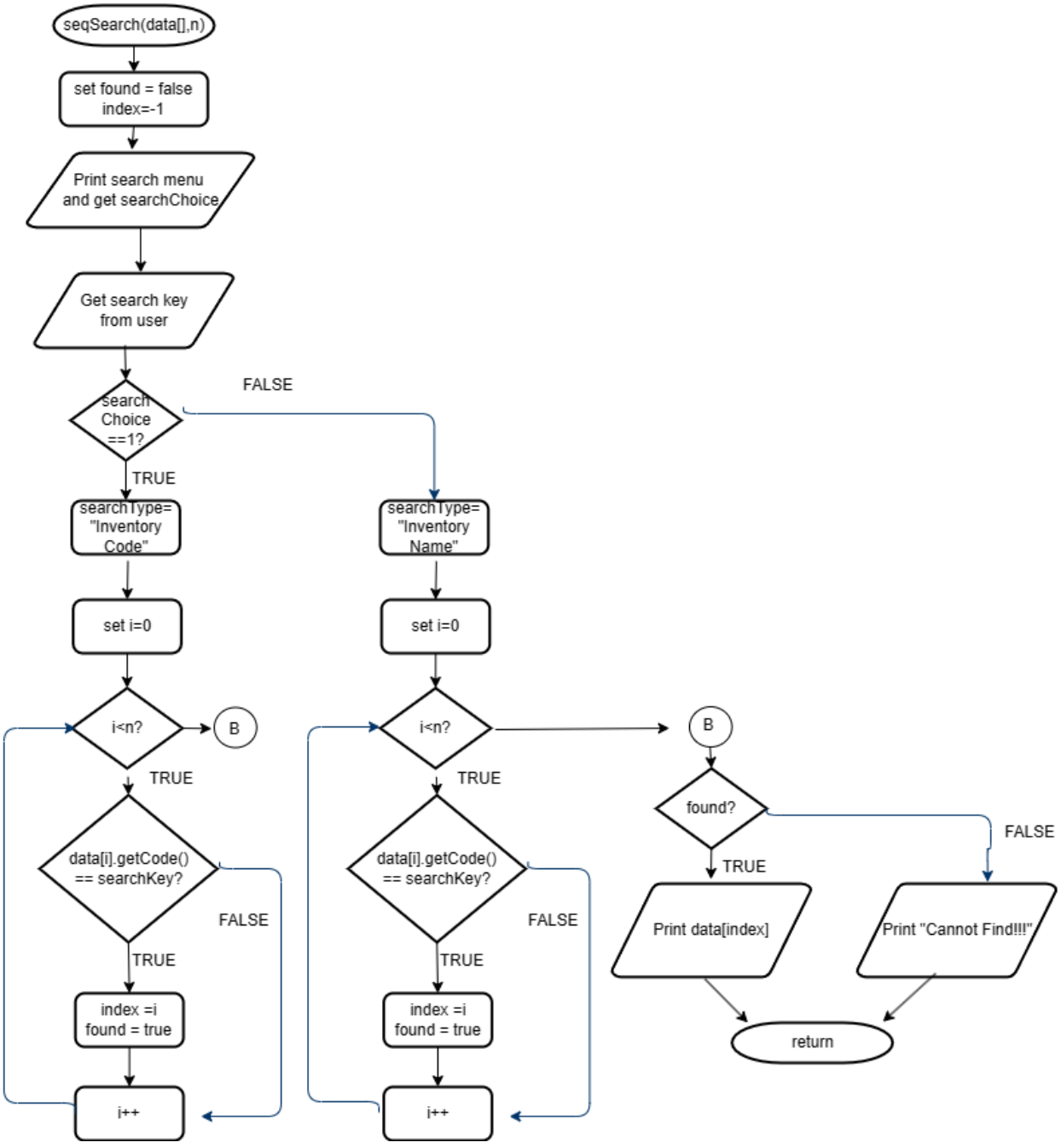



1
Print welcoming message
Print inventory object array
loop?
FALSE
TRUE
Print main menu and get mChoice
mChoice==1?
FALSE
mChoice==2?
FALSE
mChoice==3?
FALSE
TRUE
TRUE
TRUE
Print "Sorting Process"
Print "Searching Process"
Print "Bye Bye!!!"
Print "Please enter a valid choice! :)"
sort(inv, counter)
seqSearch(inv, counter)
loop=false
Print inventory object array
set i=0
i<counter?
FALSE
Close input file and output file
2
TRUE
Update output file with the inventory object array
i++

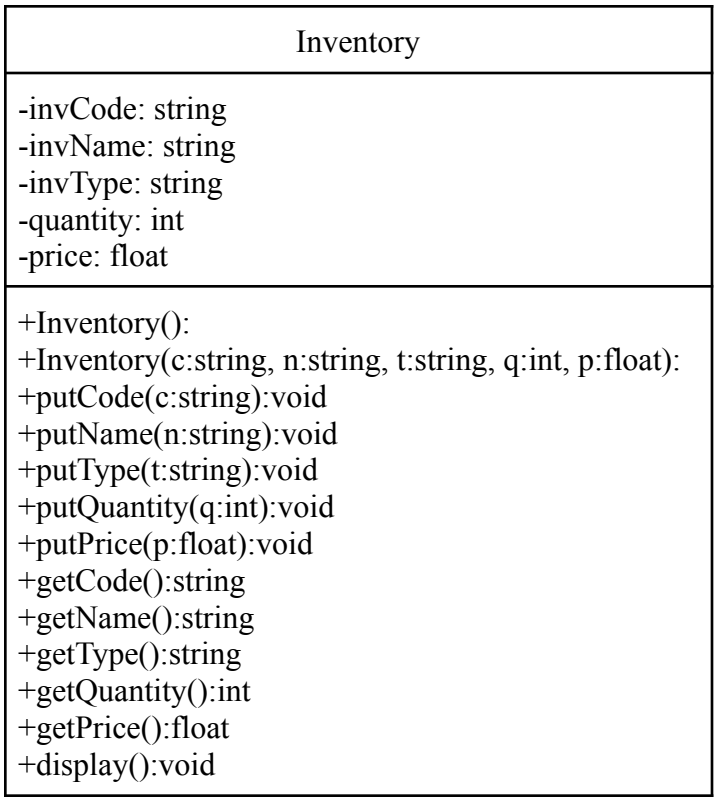

sort(data[], n)
Print sort menu and get sortChoice
Print order menu and get orderChoice
set p=1, sorted = false
return
FALSE
p<n && !sorted?
TRUE
sorted=true
set x=0
4
FALSE
x<n-p?
++p
TRUE
sort Choice ==1?
FALSE
sort Choice ==2?
FALSE
3
TRUE
sortType= "Inventory Code"
sortType= "Inventory Name"
order Choice ==1?
FALSE
TRUE
data[x].getCode()> data[x+1].getCode()?
data[x].getCode()< data[x+1].getCode()?
data[x].getName()> data[x+1].getName()?
data[x].getName()< data[x+1].getName()?
FALSE
TRUE
temp = data[x] data[x] = data[x + 1] data[x + 1] = temp
TRUE
sorted = false
FALSE
++x


3
sort Choice ==3?
FALSE
sort Choice ==4?
FALSE
A
TRUE
sortType= "Inventory Type"
TRUE
sortType= "Quantity"
order Choice ==1?
FALSE
order Choice ==1?
FALSE
TRUE
TRUE
data[x].getType()> data[x+1].getType()?
data[x].getType()< data[x+1].getType()?
data[x].getQuantity()> data[x+1].getQuantity()?
data[x].getQuantity()< data[x+1].getQuantity()?
FALSE
TRUE
FALSE
TRUE
temp = data[x] data[x] = data[x + 1] data[x + 1] = temp
TRUE
temp = data[x] data[x] = data[x + 1] data[x + 1] = temp
TRUE
sorted = false
sorted = false
FALSE
FALSE
4
++x


A
sortType= "Price"
order Choice ==1?
FALSE
TRUE
data[x].getPrice()> data[x+1].getPrice()?
data[x].getPrice()< data[x+1].getPrice()?
FALSE
TRUE
temp = data[x] data[x] = data[x + 1] data[x + 1] = temp
TRUE
sorted = false
FALSE
++x
4


seqSearch(data[],n)
set found = false
index=-1
Print search menu
and get searchChoice
Get search key
from user
search
Choice
==1?
FALSE
TRUE
searchType=
"Inventory
Code"
searchType=
"Inventory
Name"
set i=0
set i=0
i<n?
B
TRUE
data[i].getCode()
== searchKey?
FALSE
TRUE
index =i
found = true
i++
i<n?
B
TRUE
data[i].getCode()
== searchKey?
FALSE
TRUE
index =i
found = true
i++
found?
FALSE
TRUE
Print data[index]
Print "Cannot Find!!!"
return

3.3. Class Diagram



4. Data Structure Operation Description

4.1 Sorting

In implementing our inventory management system, we use **improved bubble sort** technique to sort our inventory.

Improved bubble sort is an enhanced sorting technique from bubble sort (to minimize unnecessary iteration). The improved bubble sort is used to iteratively compare and swap the adjacent elements until the entire array is sorted. The sorting is based on user chosen criteria such as inventory code, name, type, quantity, or price. The sorting order can be ascending or descending. The function "**sort**" consists of 4 important parts (function, data, loops). I will briefly describe the data structure operation that happened in this function.

void sort(Inventory data[], int n)
The "sort" function receives an array of Inventory objects (data) and the number of elements in the array (n) as parameters.

int sortChoice = menu();
int orderChoice = sortMenu();
Inside the sort function, the sortChoice and orderChoice receive the value of the user chosen type of sorting (inventory code, name, type, quantity, or price) and the order (ascending or descending).

for (int p = 1; (p < n) && !sorted; ++p)
This is an external loop which to check whether the list is sorted in order to improve the efficiency of the bubble sort. The sorting process continues until the array is sorted or no swaps are made during the pass.

for (int x = 0; x < n - p; ++x)
This is the internal loop, which is used to control the swap if they are not in order. As it finished its execution, the largest/smallest in the array will be moved at the top (depending on the order chosen).
In addition, inside the internal loop, there are several switch case that refer to the sortChoice data. The specific section of the switch statement is executed depending on the sortChoice value (user's choice of sorting criteria).

```
switch (sortChoice)
{
case 1:
    sortType = "Inventory Code";
    if (orderChoice == 1)
    {
        if (data[x].getCode() > data[x + 1].getCode())
        {
            Inventory temp = data[x];
            data[x] = data[x + 1];
            data[x + 1] = temp;
            sorted = false;
        }
    }
```

As shown in figure above, the "if" statement in case 1 ***if (data[x].getCode() > data[x + 1].getCode())*** is used to compare the adjacent elements.

4.2 Searching

In implementing our inventory management system, we use **sequential search** technique to search the information of our inventory.

In sequential search, every element in the array will be checked until the search key is found or the process reaches the last element. In the inventory management system, the user is prompted to enter their desired searching criteria whether by inventory code or inventory name. This means every element in the selected list will be checked by the system using the user entered key (name or code). If the corresponding keyword is found, the system will display the information of the inventory item.

In the implementation of our inventory management system, **void seqSearch(Inventory data[], int n)** is used as a function to perform sequential sort. The function receives an array of Inventory objects (data) and the number of elements in the array (n) as parameters.

int searchChoice = menuSearch();
searchChoice data is used to acknowledge the number searching type entered by the user. The system then prompts the user to enter the searching key.

Initially the index is set to -1 , found is set to false.

```
switch (searchChoice)
{
case 1:
    searchType = "Inventory Code";
    for (int i = 0; i < n; i++)
    {
        if (data[i].getCode() == searchKey)
        {
            index = i;
            found = true;
        }
    }
    break;
```

if (data[i].getCode() == searchKey) is used to check the elements one by one inside the data array. If the key is matched, the sequential search is performed successfully since the found is now set to true.

```
if (found)
{
    data[index].display();
    cout << endl;
}
else
{
    cout << "Cannot Find!!!\n";
}
cout << endl;
```

Finally, the information of the array will be displayed based on the index found.

5. Conclusion

Sorting and searching techniques are both the techniques that are implemented in the inventory management system so that the information in the system is more organised and easy to find. Finding a suitable approach to implement in an inventory management system is important because it can improve the productivity of an organization.

By completing this assignment, our team has gained much knowledge and practical experience regarding how to implement sorting and searching functions in an inventory management system. Starting from design, implementation of codes, until we get the output of the system, we truly gained a deeper understanding on how algorithms work in our real life system.