FACULTY OF COMPUTING
UTM Johor Bahru

FACULTY OF COMPUTING

SESSION 2023/2024

SEMESTER 1

SECJ2013-04 DATA STRUCTURE AND ALGORITHM

ASSIGNMENT 2 - DATA STRUCTURE OPERATION

LECTURER: DR. LIZAWATI BINTI MI YUSUF

NO	NAME	MATRIC NO
1	NUR HAFIZAH BINTI JAFRI	A22EC5022
2	EDDY KOH WEI HEN	A22EC0154
3	NURSYUHADA BINTI BADREN	A22EC0253

Table of Contents

1. Objectives

Our objectives for this assignment are:

a. To store and arrange customer information using linked list structure.

b. Keep record of booking details, such as customers' name, age, IC number, phone number, room number and check-in date, with each customer node.

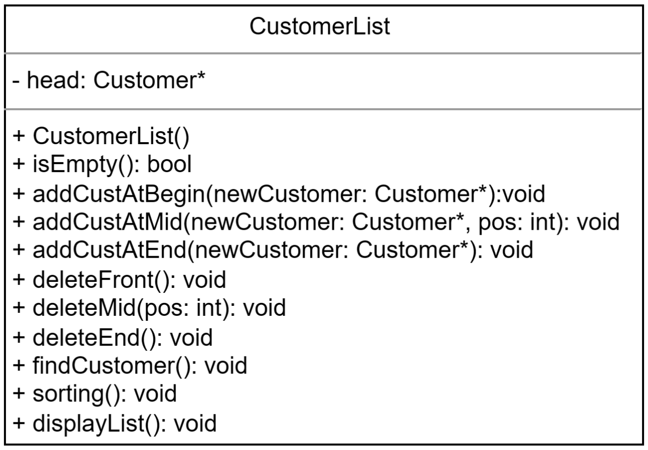
c. To allow changeable functions such as adding, deleting, finding, sorting customers from the list.

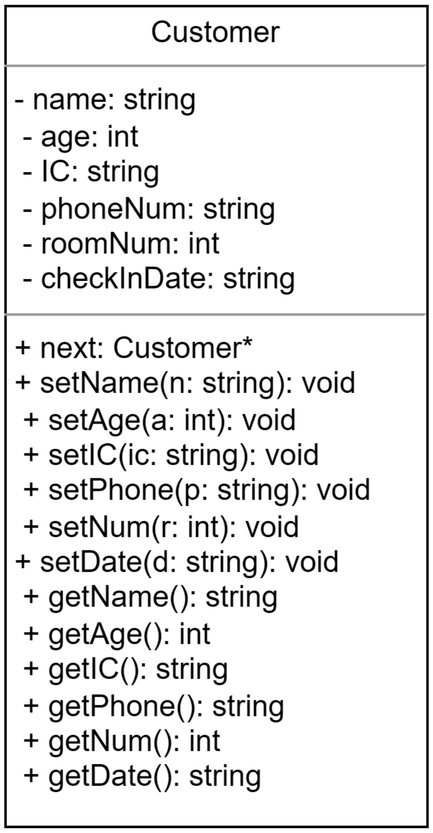
2. Synopsis of the project

This hotel booking system uses a constant change of linked lists, allowing for the easy handling of customer information in a way similar to the changeable movement of customers throughout a hotel. For example, when customers check in their rooms, it easily inserts new customers by entering their data at any location they want within the list. When customers check out, their data is easily removed from the list, creating an easy checkout procedure. This ability to change, which is similar to the constant change of hotel operations, provides an easy and effective experience for both staff and customers.

3. Design

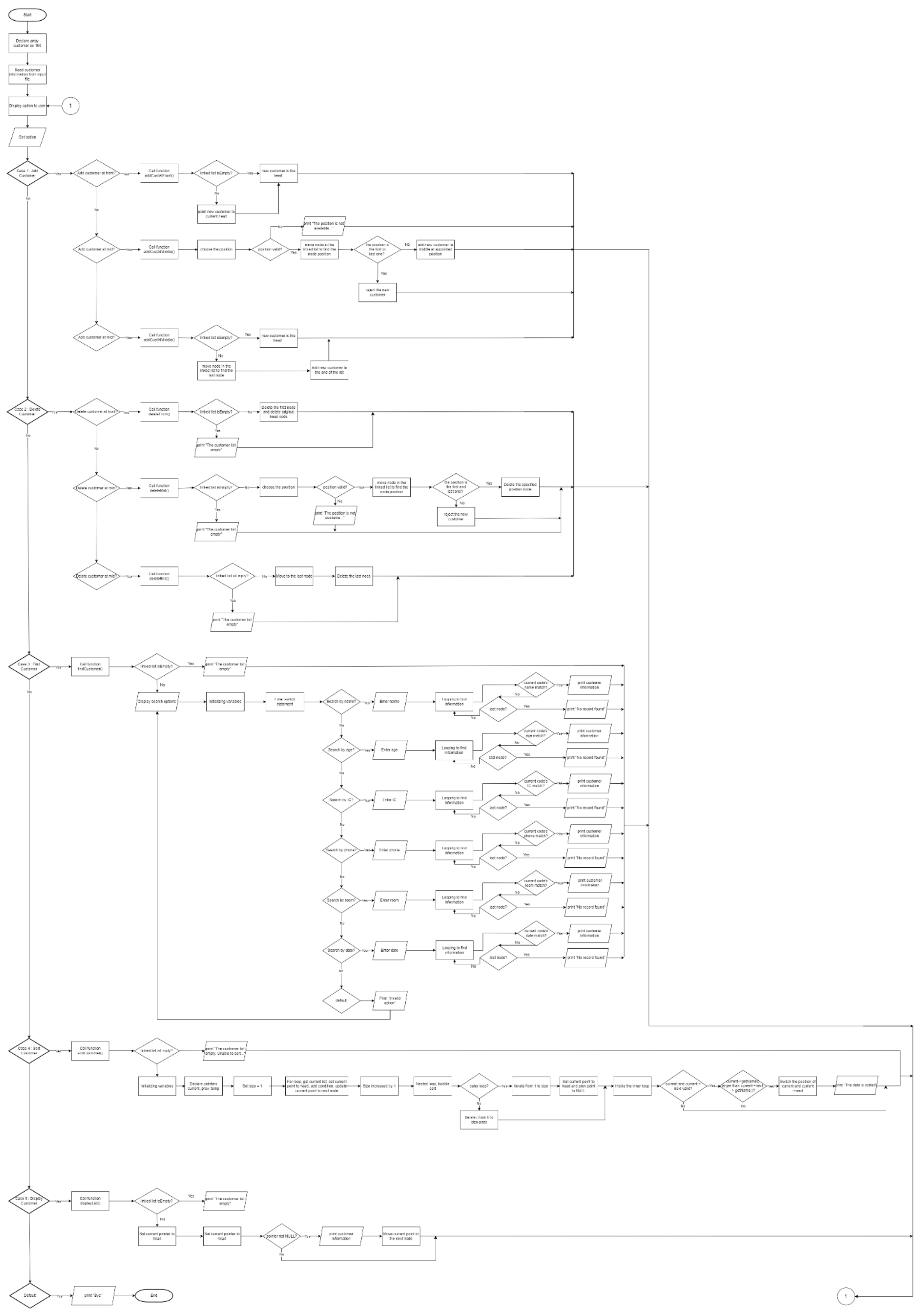
3.1 Class Diagram


CustomerList
- head: Customer*
+ CustomerList()
+ isEmpty(): bool
+ addCustAtBegin(newCustomer: Customer*):void
+ addCustAtMid(newCustomer: Customer*, pos: int): void
+ addCustAtEnd(newCustomer: Customer*): void
+ deleteFront(): void
+ deleteMid(pos: int): void
+ deleteEnd(): void
+ findCustomer(): void
+ sorting(): void
+ displayList(): void



3.2 Flowchart

Flowchart Link : https://app.diagrams.net/#G1_IaB1tLsIodmKWIh0TbjkgSj5COdUm9j



Start
Declare array customer as 100
Read customer information from input file
Display option to user
1
Get option
Case 1 : Add Customer
Add customer at front?
Call function addCustAtFront()
linked list isEmpty?
new customer is the head
point new customer to current head
Add customer at mid?
Call function addCustAtMiddle()
choose the position
position valid?
print "The position is not available..."
move node in the linked list to find the node position
the position is the first or last one?
add new customer in middle at appointed position
reject the new customer
move node in the linked list to find the last node
Add new customer to the end of the list
Case 2 : Delete Customer
Delete customer at front?
Call function deleteFront()
Delete the first node and delete original head node
print "The customer list empty"
Delete customer at mid?
Call function deleteMid()
the position is the first and last one?
Delete the specified position node
Call function deleteEnd()
Move to the last node
Delete the last node
Case 3 : Find Customer
Call function findCustomer()
Display search options
Initializing variables
Enter switch statement
Search by name?
Enter name
Looping to find information
current node's name match?
print customer information
last node?
print "No record found"
Search by age?
Enter age
Search by IC?
Enter IC
Search by phone?
Enter phone
Search by room?
Enter room
Search by date?
Enter date
default
Print "Invalid option"
Case 4 : Sort Customer
Call function sortCustomer()
print "The customer list empty. Unable to sort..."
Declare pointers current, prev, temp
Set size = 1
For loop, get current list, set current point to head, add condition, update current point to next node
Size increased by 1
Nested loop, bubble sort
outer loop?
Iterate from 1 to size
Iterate j from 0 to size-pass
Set current point to head and prev point to NULL
Inside the inner loop
current and current->next valid?
current->getName() larger than current->next->getName()?
Switch the position of current and current->next
print "The data is sorted"
Case 5 : Display Customer
Call function displayList()
Set current pointer to head
pointer not NULL?
Move current point to the next node
Default
print "Bye"
End
Yes
No

4. Design Description

4.1 Main function

1. Declare array ‘customer’ of 100 ‘Customer’
2. Read customer information from an input file.
3. Switch case: Case 1: Add Customer
    - 3.1 Select to add customer at the front of the list
        - 3.1.1 call function addCustAtBegin()
    - 3.2 Select to add customer in the middle of any position of the list
        - 3.2.1 call function addCustAtMid()
    - 3.3 Select to add customer at end of the list
        - 3.3.1 call function addCustAtEnd()
4. Switch case: Case 2 : Delete Customer
    - 4.1 Select to delete first customer from the list
        - 4.1.1 call function deleteFront()
    - 4.2 Select to delete customer in the middle of any position of the list
        - 4.2.1 call function deleteMid()
    - 4.3 Select to delete last customer from the list
        - 4.3.1 call function deleteEnd()
5. Switch case: Case 3 : Find Customer
    - 5.1 call function findCustomer() to search for customers based on specified criteria
6. Switch case: Case 4 : Sort Customer
    - 6.1 call function sorting() to sort customer list based on names using bubble sort algorithm
7. Switch case: Case 5 : Display Customer
    - 7.1 Call the function displayList() to display the entire list of customers.

8. Case Default : Quit

    8.1 Display “Bye” and quit the program/system.

9. The program will infinite loop until user key in quit option

4.2 Add Function

4.2.1 AddCustAtBegin()

1. Check if the linked list is empty
2. If the linked list is empty, make the new customer at the head of the list.
3. If the linked list is not empty, point the new customer to the current head.
4. Make the new customer the new head of the list.

4.2.2 AddCustAtMid()

```
1. Check if the position is valid or not
2. Move the linked list to find the node at specified position.
   2.1 If the position is the last one
           2.1.1 Reject the new customer
   2.2 If the position is the first one
           2.2.1 Reject the new customer
3. Add the new customer to the middle of the list by setting its next pointer to the node at
   specified position and setting the next pointer of the previous node to the new customer.
```

4.2.3 AddCustAtEnd()

```
1. Check if the linked list is empty.
2. If the list is empty,
           2.1 Make the new customer the head of the list.
3. If the list is not empty,
           2.1 Move the linked list to find the last node.
```

4. Add the new customer to the end of the list by setting the next pointer of the last node to the new customer.

4.3 Delete Function

4.3.1 deleteFront()

1. Check if the linked list is empty.
2. If the list is empty
    - 2.1 Inform user the list is empty and return
3. If the linked is not empty,
    - 3.1 Delete the first node of the list by setting the head of the list to the next node, and delete the original head node.

4.3.2 deleteMid()

1. Check if the linked list is empty.
2. Reject if position is not valid.
3. If the list is empty
    - 3.1 Inform user the list is empty and return
4. Move the node in the linked list to the specified position.
5. Reject if position is the last one.
6. Reject if the position is the first one.
7. Delete the node at specified position by setting the next pointer of the previous node to the node after the node to be deleted and then deleting the node.

4.3.3 deleteEnd()

1. Check if the linked list is empty.
2. If the linked list is empty,
    - 2.1 Inform user the list is empty and return
3. If the linked is not empty,
    - 3.1 Move to the last node of the linked list.
    - 3.2 Delete the last node of the linked list.

4.4 Find Function

4.4.1 findCustomer()

```
1. Check if the linked list is empty.
2. If the linked list is empty,
   2.1 Inform user the list is empty and unable to search, then return
3. Display search options
4. Initializing variables
   4.1 Set current pointer to the head of the list
   4.2 Set prev pointer to NULL
   4.3 Set found flag to false
5. Enter switch statement after user select the search option
   5.1 Search by name
           5.1.1 Enter name to search
           5.1.2 If current node's name matches search name
                   5.1.2.1 Set found to true to break the loop after current loop
           5.1.3 Move prev to current
           5.1.4 Move current to next node
   5.2 Search by Age
           5.2.1 Enter age to search
           5.2.2 If current node's name matches search name
                   5.2.2.1 Set found to true to break the loop after current loop
           5.2.3 Move prev to current
           5.2.4 Move current to next node
   5.3 Search by IC Number
           5.3.1 Enter IC number to search
           5.3.2 If current node's name matches search name
                   5.3.2.1 Set found to true to break the loop after current loop
           5.3.3 Move prev to current
           5.3.4 Move current to next node
   5.4 Search by Phone Number
           5.4.1 Enter phone number to search
           5.4.2 If current node's name matches search name
                   5.4.2.1 Set found to true to break the loop after current loop
           5.4.3 Move prev to current
           5.4.4 Move current to next node
   5.5 Search by Room Number
           5.5.1 Enter room number to search
```

```
        5.5.2 If current node’s name matches search name
            5.5.2.1 Set found to true to break the loop after current loop
        5.5.3 Move prev to current
        5.5.4 Move current to next node
    5.6 Search by  Check In Date
        5.6.1 Enter check in date to search
        5.6.2 If current node’s name matches search name
            5.6.2.1 Set found to true to break the loop after current loop
        5.6.3 Move prev to current
        5.6.4 Move current to next node
    5.7 Default
        5.7.1 Inform the user the option is invalid and let them choose again
6.  Display results
    6.1 If a match is found
        6.1.1 Print customer information
    6.2 If no match is found
        6.2.1 Inform user not record found
```

4.5 Sort Function

4.5.1 sorting()

```
1.  Check if the linked list is empty
2.  If the linked list is empty
    2.1 Inform user the list is empty and unable to sort then return
3.  Initializing variables
    3.1 Declare pointers current, prev and temp
    3.2 Set size = 1
4.  For loop is used to get current list size, set current point to head of the list and add
    condition of looping if current is not NULL, update current point to the next node
    4.1 size increase by 1
5.  Nested loop for bubble sort
    5.1 For outer loop, iterate from 1 to size
        5.1.1 Set current point to head and prev point to NULL in every outer loop
    5.2 For inner loop, iterating j from 0 to size-pass
6.  Inside the inner loop
    6.1 Check if current and current->next is valid and if current->getName() is larger than
        current->next->getName()
        6.1.1 switch the position of current and current->next
7. Inform the user the data is sorted.
```

4.6 Display Function

4.6.1 displayList()

```
1. Check if the linked list is empty
   1.1 If the customer list is empty
        1.1.1 Inform user the list is empty and need to insert data first then return
2. Set the current pointer point to the head of the list.
3. Looping while current pointer is not NULL
   3.1 Print customer information
   3.2 Move current point to the next node in the list
```