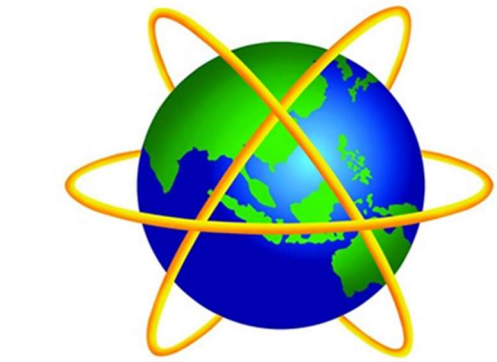

Group Member : Chan Jia Le (TP049952)
Lee Jin Heng (TP053710)
Chen Chee Kin (TP053224)

Intake Code : UC2F1908

Module Code : CT077-3-2-DSTR

Assignment Title : Data Structure

Hand-out Date : 17th February 2020

Hand-in Date : 17th April 2020

Lecturer : Mr Au Yit Wah

Contents

1.0 | Introduction

The group is consisted of three members, each one of them are from different modules. The teammates are Chan Jia Le (TP049952) from Computer Science specialism in Data Analytics, Chen Chee Kin (TP053224) from Intelligence System and Lee Jin Heng (TP053710) from Computer Science. The aim of the proposal is to propose a tuition centre management system to provide an overall view of the system to have a smooth construction soon. The tutor management system will be able to add, display records of Tutors, Search, Sort & Display Tutor details by Tutor ID, Hourly Pay Rate or Overall Performance. Thus, modifying and deleting tutor records. All in all, the system will be constructed for the purpose for eXcel Tuition Centre for a more efficient and effective way of retrieving and managing tutor record.

1.1 | Assumptions

eXcel Tuition Centre will have various outlets based on different locations in Kuala Lumpur and Selangor. Currently, the tuition centres had based their ground at 5 locations. The locations are Bukit Bintang, Bukit Jalil, Puchong, Cheras and Kepong. Each of the location will be managed by one admin manager, thus there will be total 100 tutors. the Tuition Centre Head Quarter Human Resources Department will be managing all the records.

The Tuition Centre Head Quarter Human Resources Department will be able to add, display all records, search tutor details based on location, tutor ID, subject taught or rating and modifying it if it is necessary. All the records or details of tutor displayed will be sorted by the system in an ascending order in accordance to their TutorID to allow the Human Resources Department to navigate and manage the details easily. Each TutorID is unique and will be generated by the system for consistency. Thus, if Tutor has been terminated and returned to teach, a new Tutor ID will be issued. The tutor only can be terminated after 60 days of date terminated as requested.

For quality and consistency assurance, each tutor is only allowed to be teaching one subject. There will be 5 subject selected to be taught in each tuition centre being Chinese, Malay, English, Maths, Science. The details of each Tutor will contain TutorID, Tutor Name, Date Joined, Date Terminated, Hourly rate which is around 40 to 80 and will be decided based on subject taught, Phone Number, Address, Tuition Centre details that Tutor is working on, Subject details that Tutor is teaching and lastly the rating of the Tutor 1 being bad to 5 being

excellent. The system will also allow Human Resources Department to search the record based on rating of tutor. Lastly the system will generate a weekly detail report to check the available vacancy slot.

1.2 | Data Structure and Classes

It is decided that the system will be constructed with 4 data structures as linked list or array. The data structures include Tuition Centre, Subject, Date and Tutor. Each Tutor will be included with the 3 other data structure. The reason for separating Tuition Centre, Subject and Date as individual data structure is because each Subject, Tuition Centre and date has their own additional information like name and unique code of the centre. Other than that, the purpose of this is to better sort, manage, categorize the tuition centre and subject codes, in order to meet the objective of this system which is effectively and efficiently achieve Tutor Details.

```
struct Tutor {
    int tutorID;
    string tutorFirstName;
    string tutorLastName;
    struct Date datejoined, dateTerminated;
    double hourlyPayRate;
    struct TuitionCenter center;
    struct Subject teachSub;
    string phoneNumber;
    string tutorAddress;
    int rating;
};
struct Tutor tutorDetails[100];
```

Figure 1 Data Structure of Tutor in Array of Structure

Figure 1 shows the data structure of Tutor, which includes vital details like tutor ID, first and last name, data structure for date with member variable dateJoined and dateTerminated, hourly pay rate, phone number, address, 2 data structure of tuition center and subject and lastly rating or performance. All the details are defined as the data member of Tutor to satisfy the requirement requested by the eXcel tuition center's HR department. The structure will then be initialized as an array (array of structure) in the size of 100 with a member variable name as tutorDetails. This initialization means this array of structure can contain 100 tutor and each of them contain the details in the structure.

```
struct Tutor {
    struct Tutor *previous;
    int tutorID;
    string tutorFirstName;
    string tutorLastName;
    struct Date datejoined, dateTerminated;
    double hourlyPayRate;
    struct TuitionCenter center;
    struct Subject teachSub;
    string phoneNumber;
    string tutorAddress;
    int rating;
    struct Tutor* next;
} *start, *newnode, *previous, *current, *last;
```

Figure 2 Data Structure of Tutor in Doubly Link List

Figure 2 shows the data structure in a link list form. The team plans to implement doubly link list. The details of the structure are like the one in array of structure but with an additional previous and next node. There will also be 5 pointers which include start (which is the pointer of the start node), new node (pointer for creating a new node), previous and temp (pointer used for inserting and deleting in specific location in a sorted link list) and lastly last (pointer to the last node).

```
struct Date {
    int dd;
    int mm;
    int yyyy;
};
```

Figure 3 Data Structure of Date

Figure 3 shows data structure of Date, this data structure is included inside Data structure of Tutor as shown in Figure 1, it includes the day, month and year. The extra data structure constructed and declared as two-member variable dateJoined and dateTerminated because the team plans to use the value to calculate the day that Tutor has joined as the Tutor details only can be deleted once the day tutor had left is over 60 days.

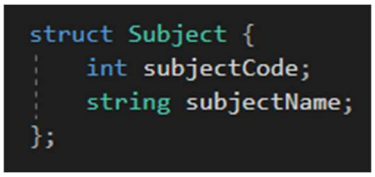

Figure 4 Data Structure of Subject

Figure 4 shows data structure of Subject, this data structure is included inside Data structure of Tutor as shown in Figure 1, it includes the subject code and subject name. The extra data structure constructed for Subject is as mentioned to better differentiate each unique subject with subject code and its name.

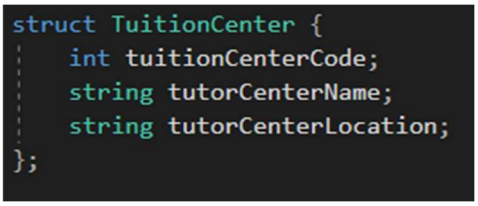

Figure 5 Data Structure of Tuition Center

Figure 5 shows the data structure of Tuition, as shown in Figure 1, this data structure is included in the data structure of Tutor, it includes details like tuition center code, name and its location. Thus, as for mentioned, it is constructed for easier differentiation of each unique tuition center for each tutor.

2.0 | System Design and Workflows

2.1 | Flow Chart Diagram

2.1.1 | Main Menu

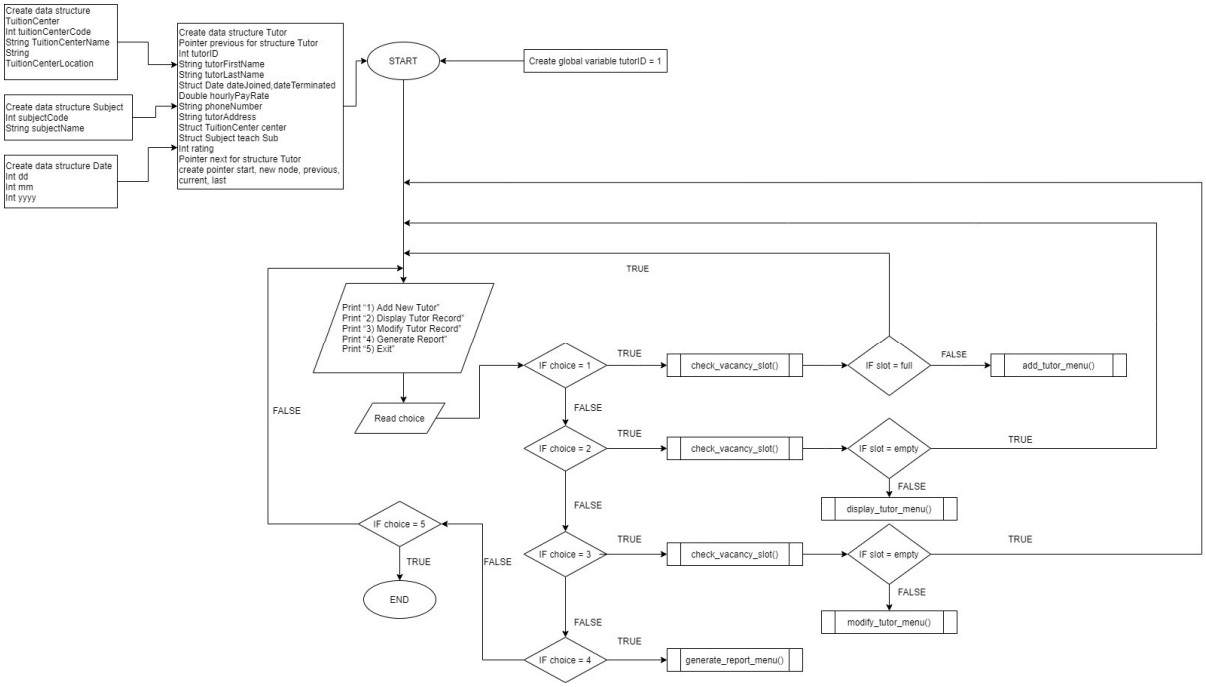

Figure 2.1.1.1 Main Menu for Link List

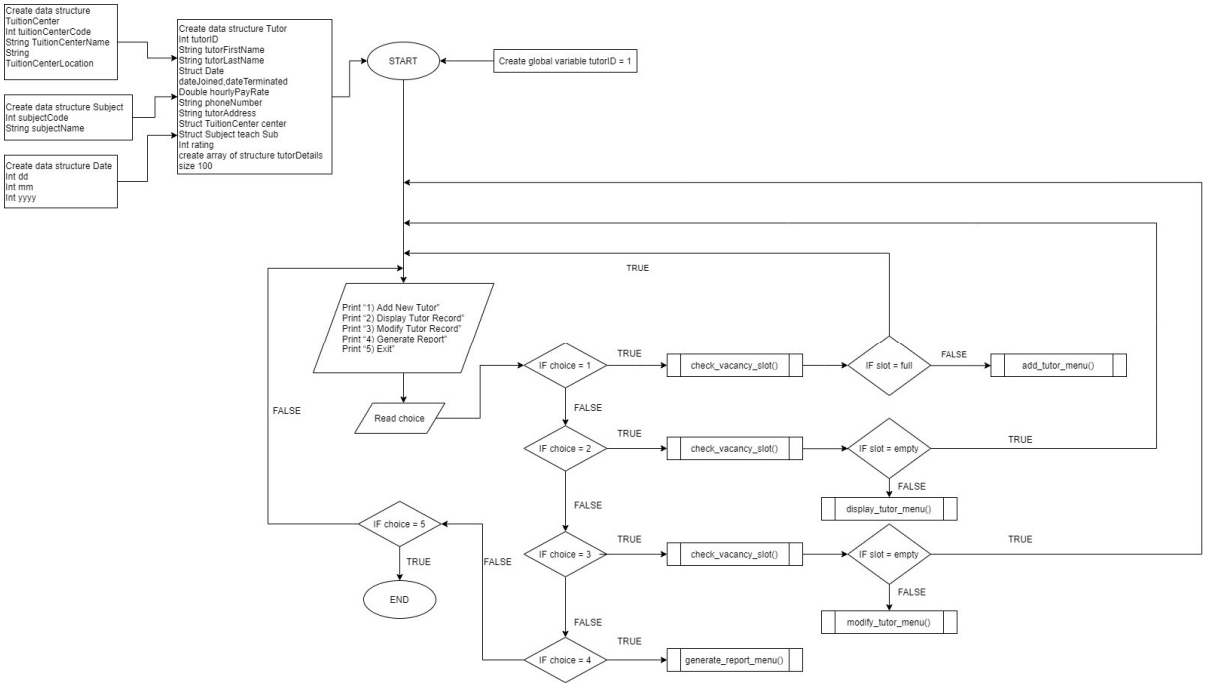

Figure 2.1.1.2 Main Menu for Array of Structure

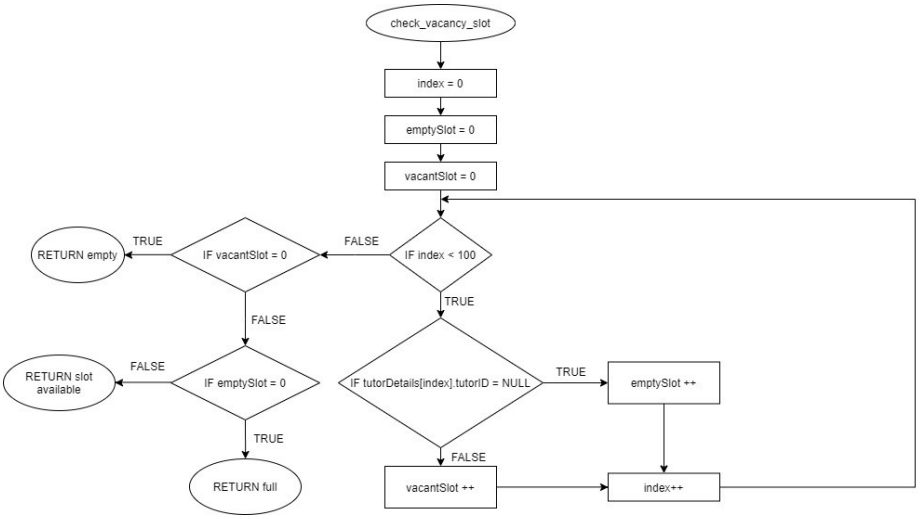

Figure 2.1.1.3 check vacancy slot at Main Menu for Array of Structure

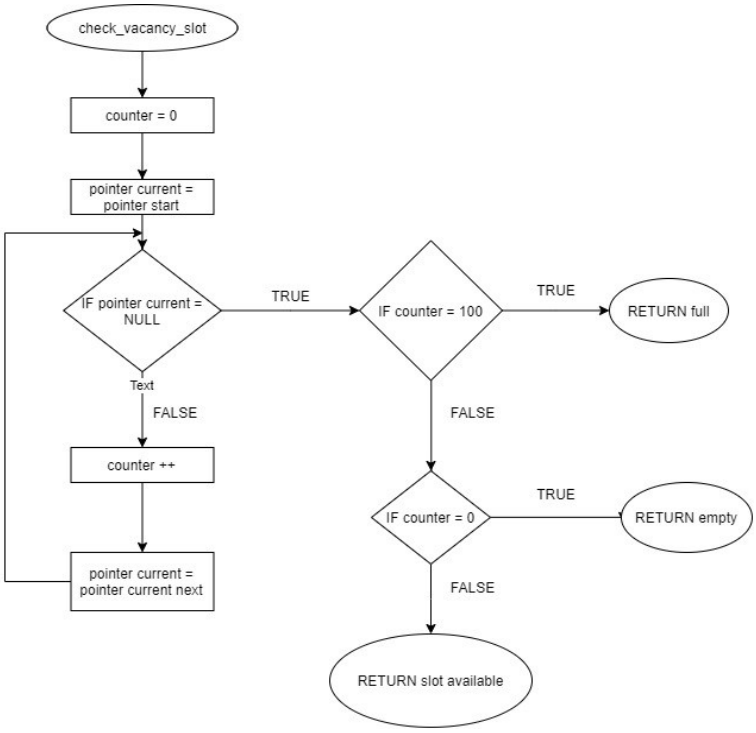

Figure 2.1.1.4 check vacancy slot at Main Menu for Linked List

Figure 2.1.1.1 and Figure 2.1.1.2 shows the main menu of the system for array of structure and linked list respectively. The program will first ask user input for each specific functionality which includes add tutor, display tutor records, modify tutor records (included with delete record), generate report or exit. Before entering add tutor menu, the program will contain a check vacancy slot function to ensure it is not full. Furthermore, if user wishes to enter display tutor record or modify tutor record, the check vacancy slot function will be implemented in order to ensure the tutor records is not empty. Figure 2.1.1.3 and Figure 2.1.1.4 shows the function check vacancy slot for both array of structure and linked list.

2.1.2 | Add Tutor Menu

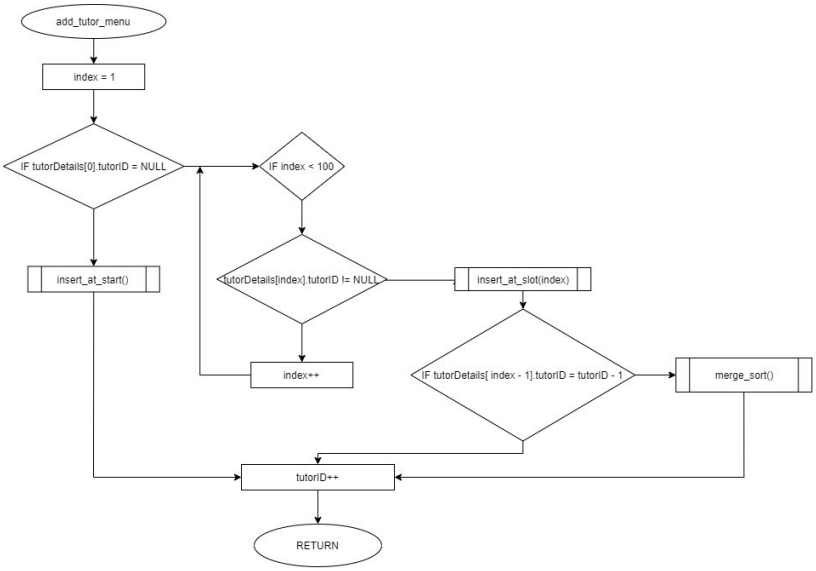

Figure 2.1.2.1 Add Tutor Menu for Array of Structure

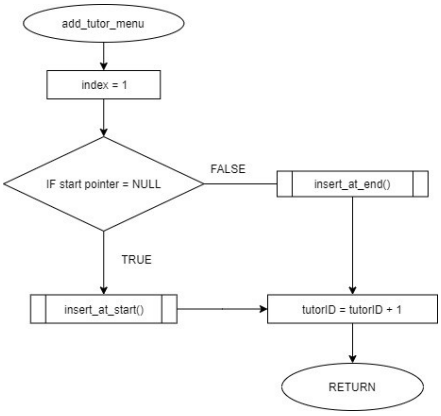

Figure 2.1.2.2 Add Tutor for Linked List

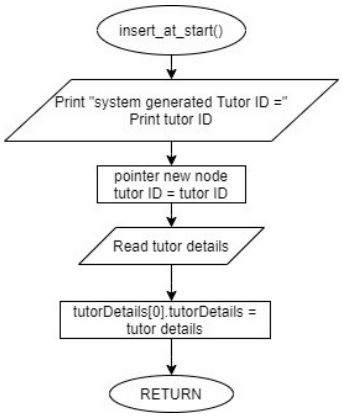

Figure 2.1.2.3 insert at start function for add tutor menu (array of structure)

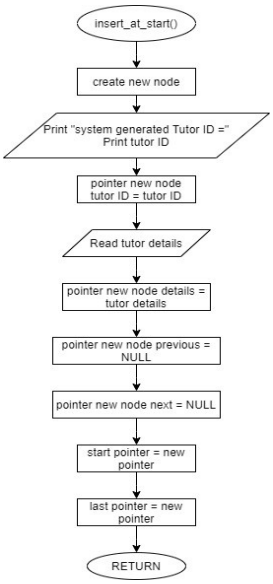

Figure 2.1.2.4 insert at start function for add tutor menu (linked list)

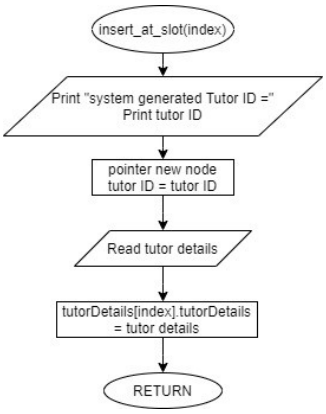

Figure 2.1.2.5 insert at slot function for add tutor menu (array of structure)

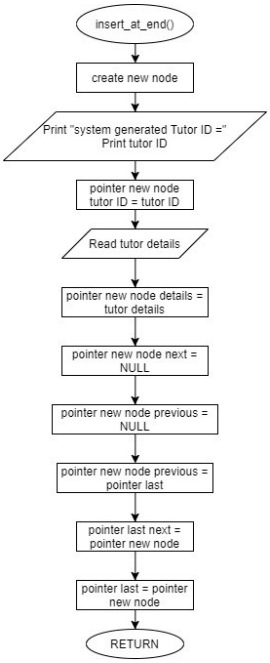

Figure 2.1.2.6 insert at end function for add tutor menu (linked list)

Add tutor menu is constructed as shown in Figure 2.1.2.1 and Figure 2.1.2.2. Add tutor menu allows user to add new tutor details to the system. The program will first identify if the first element of array or linked list is empty. If it is empty the program will insert the user input tutor details to the first element of array or linked list. If its not, the program will insert by searching for empty slot in array of structure and into the last element if its linked list. The other figures show how the input will be processed and insert into the program.

2.1.3 | Display Tutor Menu

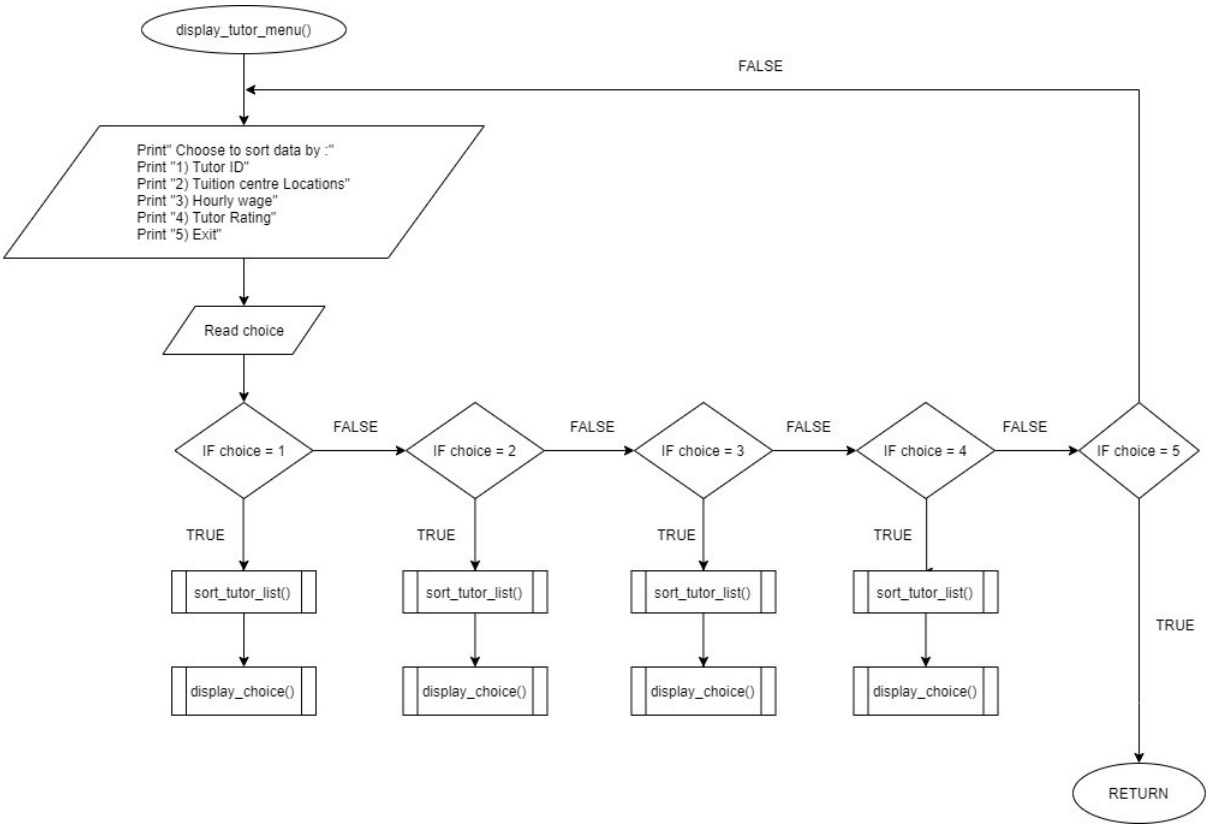

Figure 2.1.3.1 Display Tutor Menu

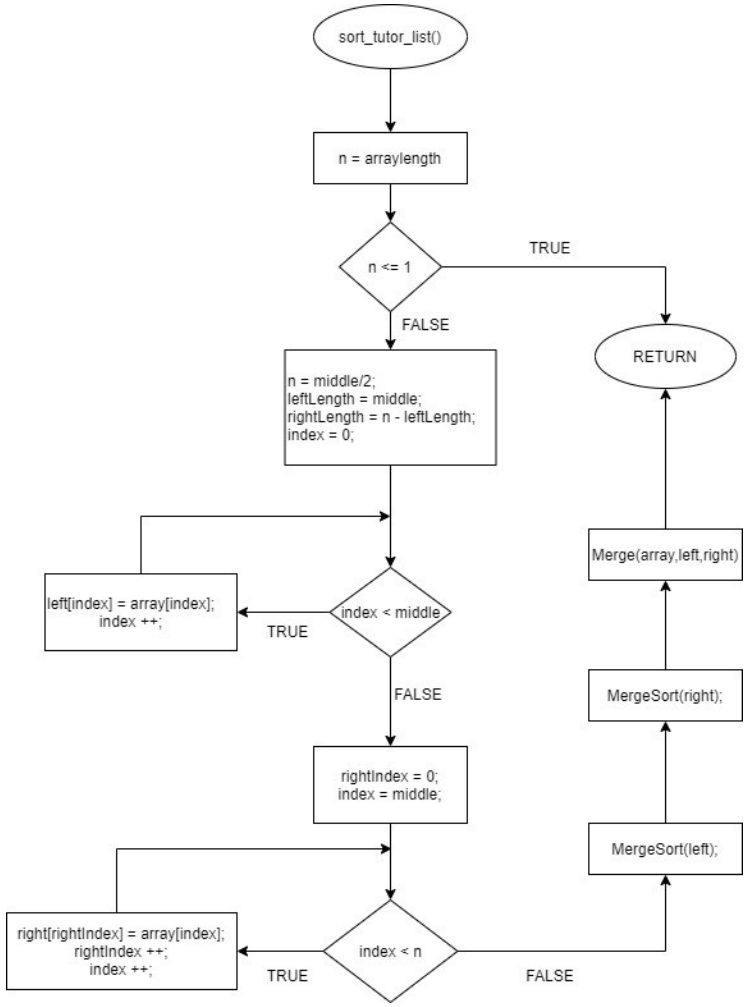

Figure 2.1.3.2 Sort Tutor List for Display Tutor Menu (Array of Structure)

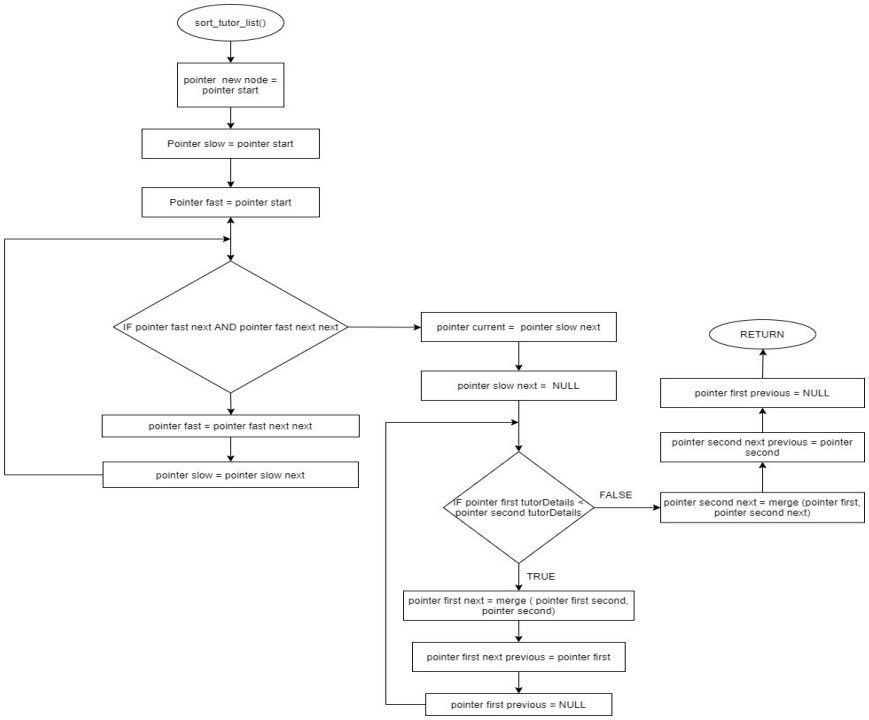

Figure 2.1.3.3 Sort Tutor List for Display Tutor Menu (Linked List)

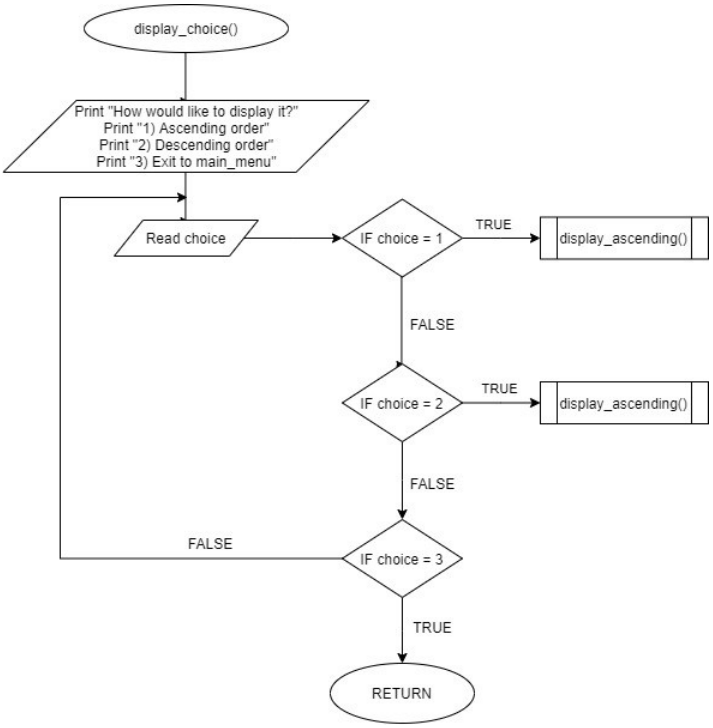

Figure 1.1.3.4 Display Choice for Display Tutor Menu (Linked List and Array of Structure)

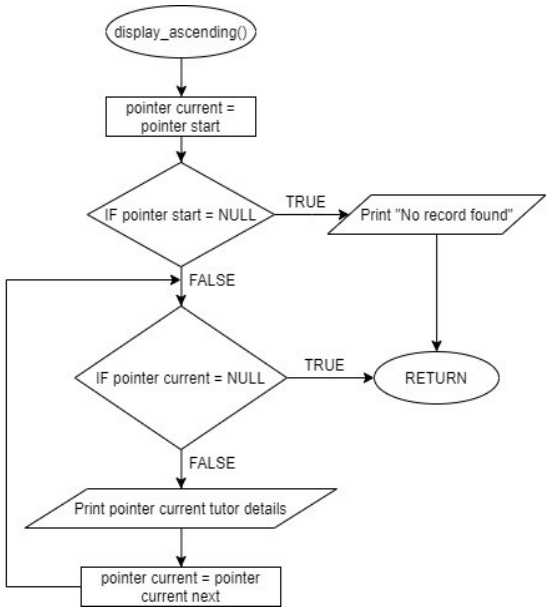

Figure 2.1.3.5 Display Ascending function for display choice (Linked List)

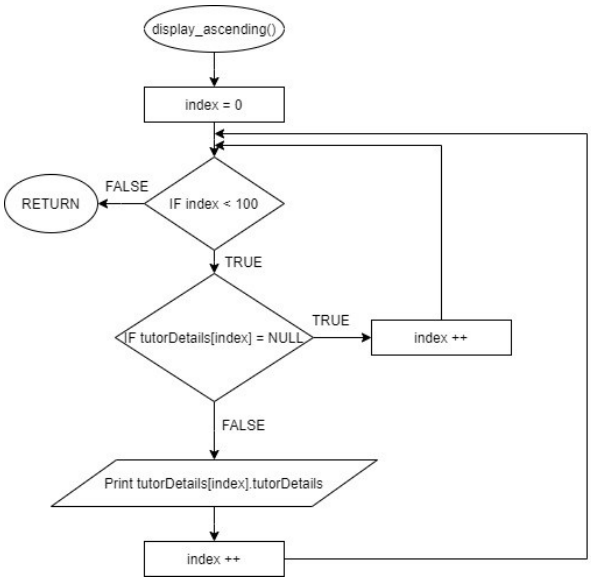

Figure 2.1.3.6 Display Ascending function for display choice (Array of Structure)

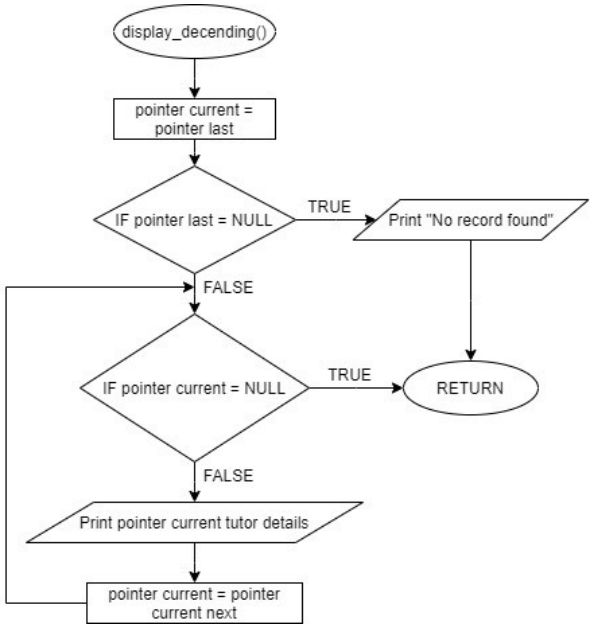

Figure 2.1.3.6 Display Descending function for display choice (Linked List)

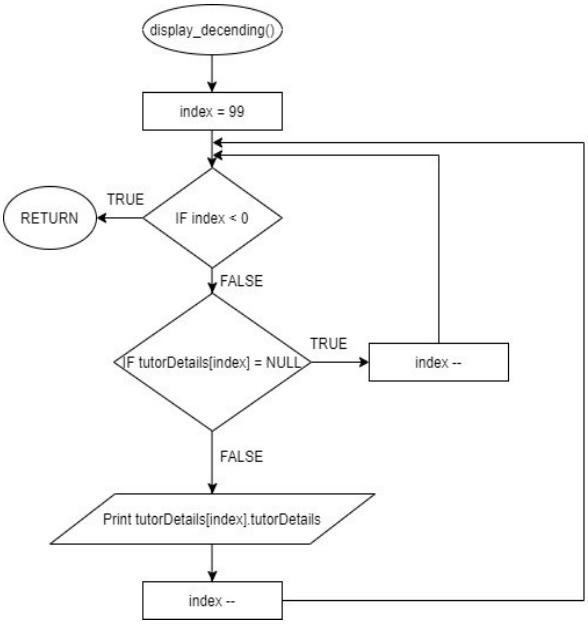

Figure 2.1.3.7 Display Descending function for display choice (Array of Structure)

Figure 2.1.3.1 is the menu for Display Tutor Details. The user will be asked to input the details that they wish to be sorted by either by tutor id, rating, hourly pay rate and so on. The system will then implement a merge sort function based on the system, array of structure or linked list. After the details are sorted based on user's desire. It will then execute the display choice function, shown in figure 2.1.3.4. The system will then request user to display either in ascending or descending order. Once user inputs the desired outcome, the system will then execute the function display either by ascending or descending as shown in figure 2.1.3.5 to figure 2.1.3.7. after finishing displaying all the tutor details. It will then return to the main menu.

2.1.4 | Modify Tutor Menu

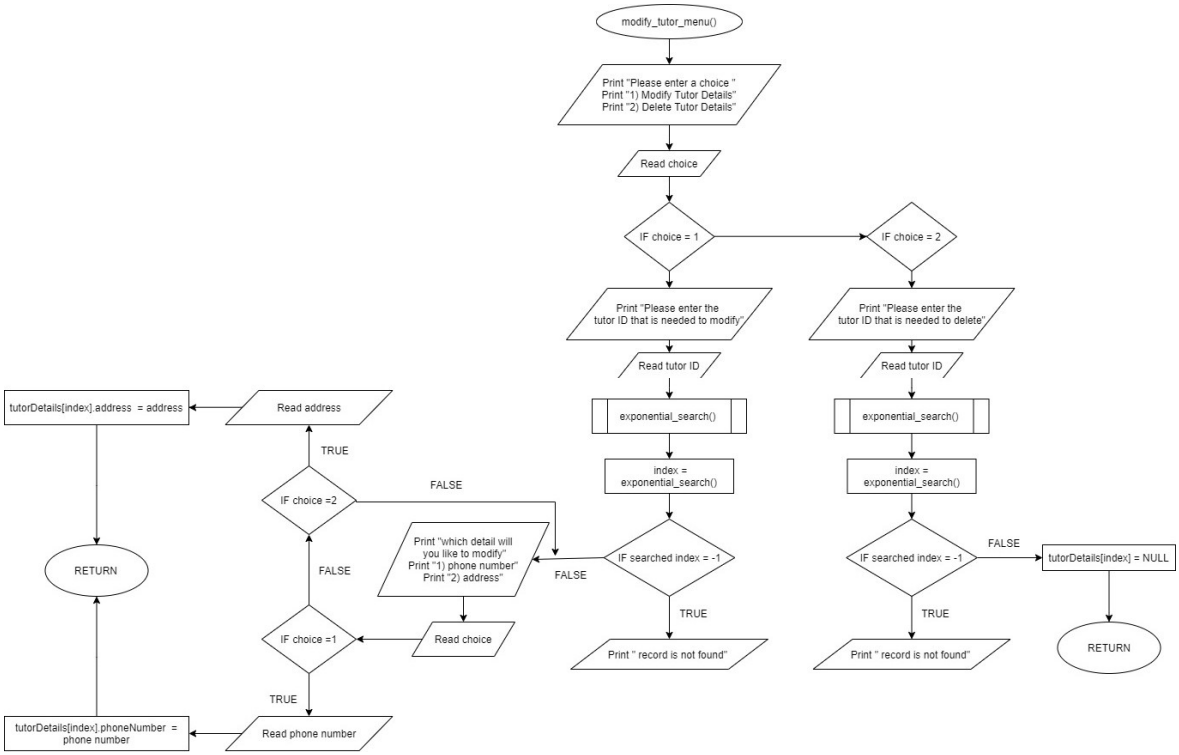

Figure 2.1.4.1 Modify Tutor Menu for Array of Structure

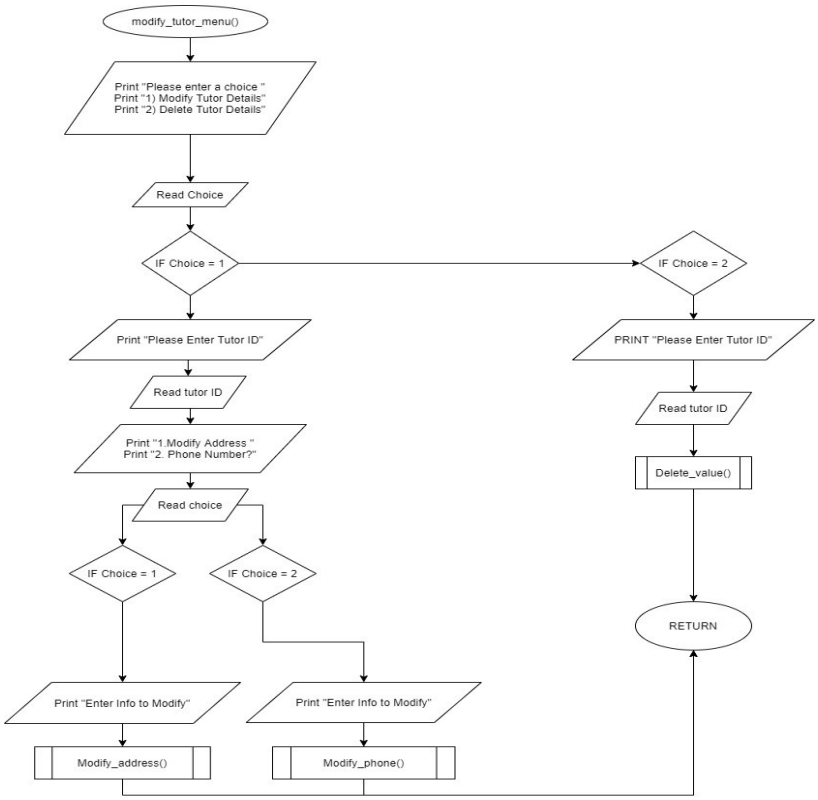

Figure 2.1.4.2 Modify Tutor Menu for Linked List

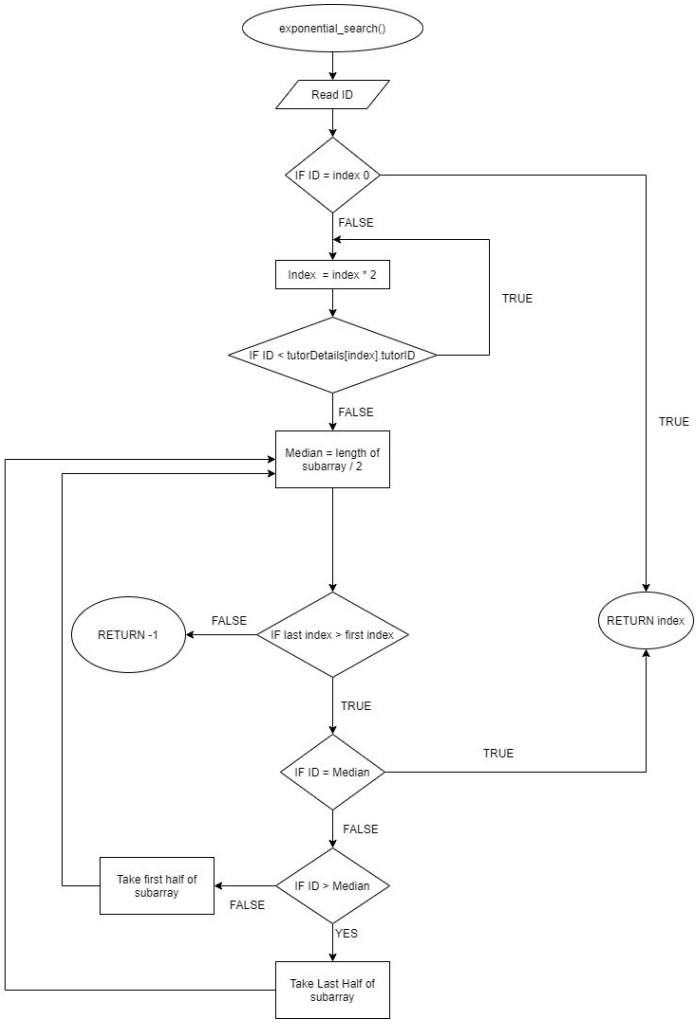

Figure 2.1.4.3 Exponential Search for Modify Tutor Menu (Array of Structure)

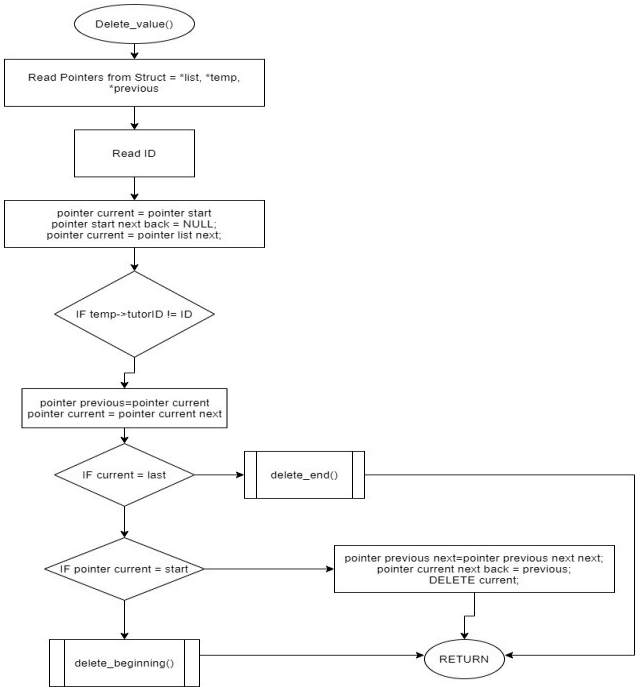

Figure 2.1.4.4 Delete value for Modify Tutor Menu (Linked List)

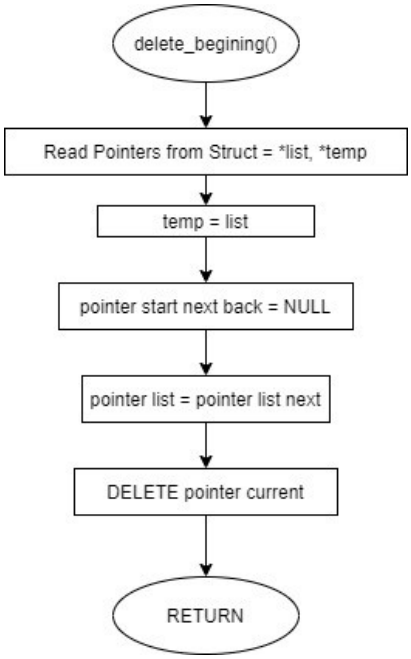

Figure 2.1.4.5 Delete beginning function for Delete Value (Linked List)

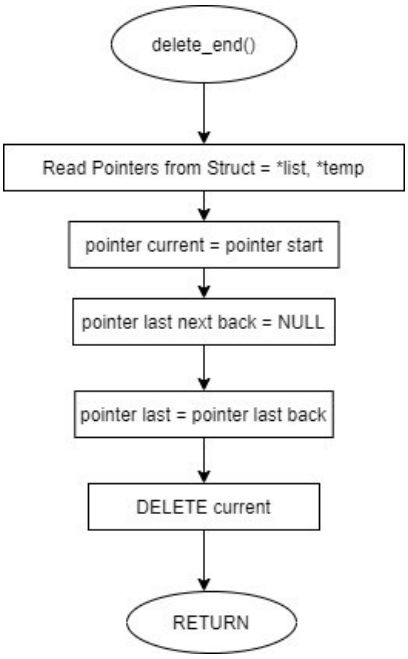

Figure 2.1.4.6 Delete end function for Delete Value (Linked List)

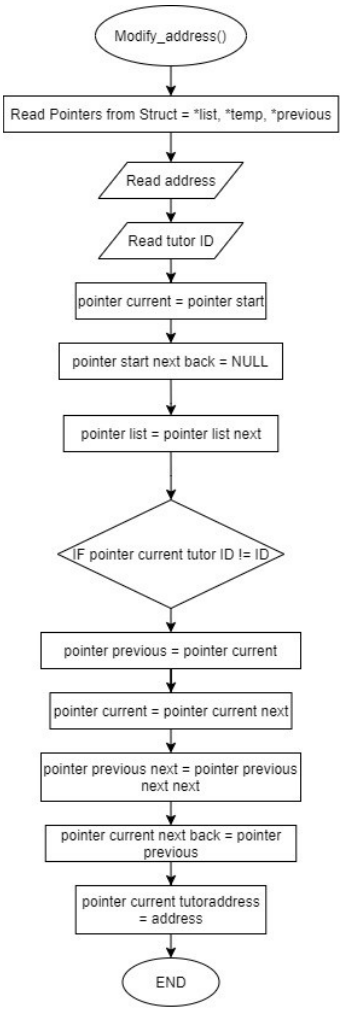

Figure 2.1.4.7 Modify address for Modify Tutor Menu (Linked List)

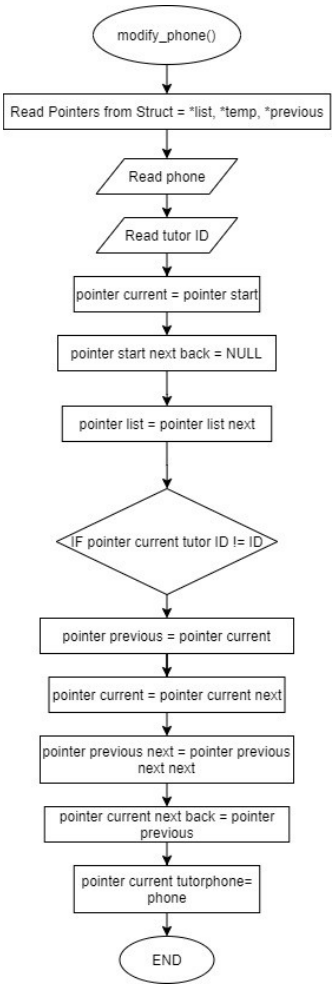

Figure 2.1.4.8 Modify address for Modify Tutor Menu (Linked List)

Figure 2.1.4.1 and Figure 2.1.4.2 are the flow chart for Modifying Tutor Menu. Modifying Tutor Menu is the menu for user to modify Tutor's phone or address or Deleting a tutor detail. For array of structure it will perform an exponential search to search for the desired Tutor by using Tutor ID and the program will then ask user to input the desired phone number or address or delete the tutor details. The program will then return to the main menu. For Linked list, the program will ask the same user input and the program will transverse through the linked list to identify the position of desired tutor details and modify it or delete accordingly. The other figures show the function that is embedded inside the modify tutor menu.

2.1.5 | Generate Report Menu

Figure 2.1.5.1 generate report menu for Array of Structure

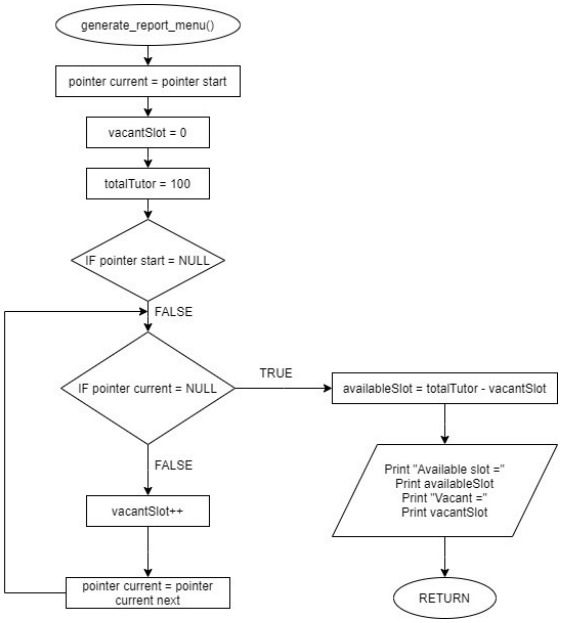

Figure 2.1.5.2 generate report menu for Linked List

Figure 2.1.5.1 and Figure 2.1.5.2 are both flow charts for generate report menu. One is constructed with the array of structure and one with Linked List. Generate report menu is constructed for the sake of easy report generation, it will go through the array of structure or linked list to calculate the available slot for vacancy and the occupied number and print it out for user's information.

2.2 | Algorithms Implemented

2.2.1 | Merge Sort

The sorting method the team has propose for this project will be the merge sort in a doubly link-list. The merge sort is essentially a sorting algorithm which works by recursively sort the subarray array[p..q] and recursively sort the subarray array[q+1..r]. The solutions to the sub-problems are then combined to produce a solution to the original problem. So, the summary of the Merge sort is, it will first divide the array into equal halves, which will be then combined into a sorted manner.

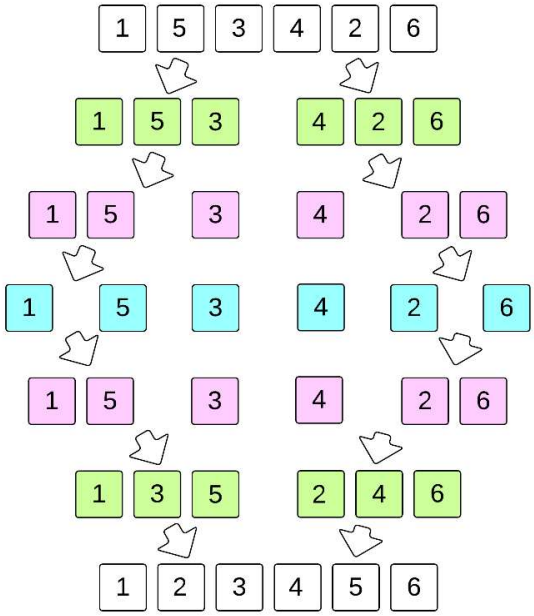

Figure 2.2.1.1: Example of Merge Sort *(Sehgal, 2018)*

As an example shown in figure 2.2.1.1, the first subarray is actually the full array, array[0..5] (p=0 and q=5), as such they have more than 2 elements which is not a base case. It will be divided and r=3. We will end up with 2 arrays at this point, in this case it would be array[0..2], that contains [1,5,3] and array[3..5] that contains [4,2,6]. This step continues with p=0 and q=2, compute r=1, recursively sort array[0..1] ([1,5]) and array[2] ([3]). Since the array [2] has a single element it will no longer be divided further and array[0..1]([1,5]) will be divided as p=0 and q=1, compute r=1. As such the arrays are now having single elements which are base cases. This will then start the merging prosses which uses when $p \geq q$ then it is sorted, if not $p < q$. This process will then proceed until the array is fully merged hence the array is then sorted.

Unlike insertion and bubble sort. It also does not go through the whole list several times which can save computational time. Furthermore, it has consistent running time which result in a more even average processing time. Thus, the requirement of this system is to handle a

relatively large array of structure, so in order to make it more effective and efficient. The merge sort algorithm is then been chosen by the team.

Merge sort algorithm will be implemented and utilized whenever user wish to display Tutor Details based on Tutor ID, Hourly Pay Rate or Overall Performance and the algorithm will then be utilize and sort according to the user choice. Other than that, the merge sort algorithm will also be used before search algorithm is implemented because a sorted array is required for the search algorithm to run effectively.

2.2.2 | Exponential Search

The searching algorithm the team will be using is named as Exponential Search. Exponential search can search for a specific element in each sorted array. This search algorithm contains 2 steps and will require binary search to work. First, the compiler will find the range where the element is present. After that, binary search will be conducted to find out the specific element. The idea behind this algorithm is to first create a subarray with a size 1, it will then compare the subarray's last element with the element we are searching. The size will then be expanded by double its size (e.g. 2 to 4, 4 to 8) until the last element is greater than the element we are searching. Once an index I is found, we can assume that the element needs to be found must be present between I/2 and I because the element to be find is very close to the last element of a subarray.

For example, given the sorted array:

2	4	6	7	8	9	11	21	25	34
0	1	2	3	4	5	6	7	8	9

Element wanted: 21

To find the element wanted using exponential algorithm, the system will first check if the first element which is index 0 is the element we are searching for. If the element is not what we want, the system will start searching the array from index 1 which is 4 and multiple the starting index by 2. (From 4 to 6, 6 to 8, 8 to 25) Since 25 is greater than 21, the range of

subarray for searching will be from index 4 to index 8 as the range is from index I/2 to I where index I is the last index of subarray. This ends the job for exponential search and the data is transferred to binary search algorithm.

Subarray from exponential search before choosing:

8	9	11	21	25
4	5	6	7	8

Subarray after choosing:

21	25
7	8

This is where binary is implemented. Binary search in simple term, will find the median of the array and check if the element wanted is greater or smaller and choose which half of the array to search. The steps will then keep on repeating to shorten the array until the wanted element is found.

For our example, the median is 11 which 21 is greater than 11 so the last half of the sub array (index 7 to 8) is chosen. Then rinse and repeat, eventually the array will be short enough to find the element we wanted which is 21 on index 7.

The reason we are using this algorithm is because it combines two searching algorithms, the exponential search and binary search. With two of these searching algorithms, the computational time and power needed for searching arrays can be significantly reduced.

This algorithm will be implemented when user is searching a specific Tutor by using Tutor ID or searching Tutors based on the desired performance/ rating. Furthermore, the searching algorithm will also be used if user desires to modify or eliminate specific Tutor, it will also require the desired Tutor ID to modify.

3.0 | Task Distribution

	Chan Jia Le TP049952	**Chen Chee Kin** TP053324	**Lee Jin Heng** TP053710
1. Introduction	34%	33%	33%
2. Assumption	34%	33%	33%
3. Data Structure	34%	33%	33%
4. Flow Chart	34%	33%	33%
5. Searching Algorithm	34%	33%	33%
6. Sorting Algorithm	34%	33%	33%

4.0 | References

[1] Geeks for Geeks, 2019. *Exponential Search.* [Online]
Available at: https://www.geeksforgeeks.org/exponential-search/
[Accessed 8 April 2020].

[2] Holamyself, 2019. *Merge sort, advantages and disadvantages.* [Online]
Available at: https://getrevising.co.uk/grids/merge-sort-advantages-and-disadvantages
[Accessed 11 April 2020].

[3] Sehgal, K., 2018. *A Simplified Explanation of Merge Sort.* [Online]
Available at: https://medium.com/karuna-sehgal/a-simplified-explanation-of-merge-sort-77089fe03bb2
[Accessed 10 April 2020].

[4] Tech Delight , 2019. *Exponential Search.* [Online]
Available at: https://www.techiedelight.com/exponential-search/
[Accessed 7 April 2020].

[5] Tutorials point, 2018. *Exponential Search.* [Online]
Available at: https://www.tutorialspoint.com/Exponential-Search
[Accessed 1 April 2020].