Table of Contents

1.0 Introduction 2

2.0 Data Structure 3

3.0 Source Code and Output 4

Main Function 4

Login/Logout 5

Nurse 9

Add Patients according to the disability option priority on the Waiting List 9

View Patients on Waiting List 13

View Patients on Visit History List 15

Calling the Patient to be Treated 17

Delete from the front in waiting list & insert into the end in visit history list: 19

Search Patient on the Waiting List 21

Sort Waiting List 24

Doctor 29

Search and Modify Patient Record from Visit History 29

Sort and Display All Records from Visit History List in Descending Order 34

Search Patients from Visit History List 37

4.0 Conclusion 41

1.0 Introduction

This project aims to develop a patient queue based management system for the client from Klinik Sulaiman. In this project, it is requested that a C++ program to be built which contains features that help manage data of patient visiting the clinic.

The proposed system will include a login function. The user is able to login as either of the two level of users, which are nurses and doctors. The function of the system differs according to which level of user you login to the system as. For example, nurse users will be permitted to add new patients to waiting list, change the patient order on the waiting list, view waiting list, call patient to be treated, search for patient record in waiting list, and sort and display waiting list. On the other hand, doctor users can view the waiting list, search and modify patient record in visit history, sort and display patient's visit history list and search for patients from visit history list.

The project requires that the built program to implement two linked lists of patient records, which are the "waiting list" and the "visit history list" respectively. The waiting list contains records of patients who are waiting in the queue to be treated, while the visit history list contains the records of patients who has been treated and left the clinic.

Following the completed proposal documentation, below is the documentation for the finished program.

2.0 Data Structure

```
struct DoctorAndNurse {
    string userID;
    string password;
};

struct Patient {
    string patientID;
    string firstName;
    string lastName;
    int age;
    char gender;
    string phoneNo;
    string address;
    string sicknessDescription;
    string visitDate;
    string visitTime;
    char disabilityOption;
    string doctorName;
    string medicineInformation;
    Patient* nextAddress;
    Patient* prevAddress;
}*waitingHead, * visitHistoryHead, * newNode, * current, * waitingTail, * visitHistoryTail;
```

The proposed linked list will be designed for waiting list and visit history list by using doubly linked lists. However, there will be two struct created. One is used to store the user ID and password of doctor and nurse. The other struct is used to store all the patient details, such as patient ID, name, sickness description, and others.

3.0 Source Code and Output

Main Function

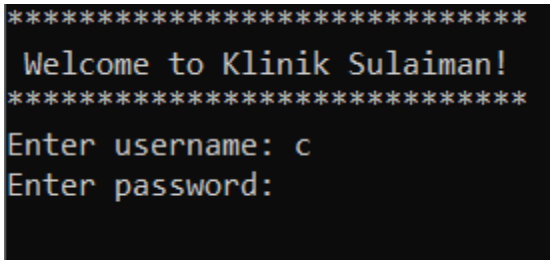
```
int main() {
    waitingHead = waitingTail = visitHistoryHead = visitHistoryTail = NULL;

    // Adding default patients into waiting list
    string defaultWaitingPatients[3][11] =
    {
        {"P4", "Lim", "Ah Kao", "28", "M", "0123456789", "No4, Jalan 4, Taman JKL", "None", "13/09/2021", "09.50", "N"},
        {"P5", "Mohammad", "Ali", "40", "M", "0129876543", "No5, Jalan 5, Taman MNO", "None", "13/09/2021", "09.40", "N"},
        {"P6", "Devi", "Darika", "12", "F", "0198765432", "No6, Jalan 6, Taman PQR", "None", "13/09/2021", "09.30", "Y"}
    };
    for (int i = 0; i < 3; i++)
    {
        insertWaitingPatient(defaultWaitingPatients[i][0], defaultWaitingPatients[i][1], defaultWaitingPatients[i][2], stoi(defaultWaitingPatients[i][3]),
            defaultWaitingPatients[i][4][0], defaultWaitingPatients[i][5], defaultWaitingPatients[i][6], defaultWaitingPatients[i][7],
            defaultWaitingPatients[i][8], defaultWaitingPatients[i][9], defaultWaitingPatients[i][10][0]);
    }

    // Adding default patients into visit history list
    string defaultVisitHistoryPatients[3][13] =
    {
        {"P1", "Tan", "Ah Mei", "58", "F", "0154887559", "No1, Jalan 1, Taman ABC", "None", "12/09/2021", "19.00", "N", "Doctor Tay", "Antibiotics"},
        {"P2", "Ahmad", "Faliq", "2", "M", "0167844877", "No2, Jalan 2, Taman DEF", "None", "12/09/2021", "09.14", "N", "Doctor Mutu", "Antibiotics"},
        {"P3", "Suresh", "Sarda", "62", "M", "0128549678", "No3, Jalan 3, Taman GHI", "None", "12/09/2021", "15.19", "N", "Doctor Tay", "Antibiotics"}
    };
    for (int i = 0; i < 3; i++)
    {
        insertVisitHistoryPatient(defaultVisitHistoryPatients[i][0], defaultVisitHistoryPatients[i][1], defaultVisitHistoryPatients[i][2],
            stoi(defaultVisitHistoryPatients[i][3]), defaultVisitHistoryPatients[i][4][0], defaultVisitHistoryPatients[i][5],
            defaultVisitHistoryPatients[i][6], defaultVisitHistoryPatients[i][7], defaultVisitHistoryPatients[i][8], defaultVisitHistoryPatients[i][9],
            defaultVisitHistoryPatients[i][10][0], defaultVisitHistoryPatients[i][11], defaultVisitHistoryPatients[i][12]);
    }

    login();
    return 0;
}
```

This is the main function of the program which will be run initially when starting the code. First of all, the waitingHead, waitingTail, visitHistoryHead, and visitHistoryTaill will be referred to null value. After that, there will be default three patients for the waiting list and three patients for the visit history list. Then, the program will start with the login function.

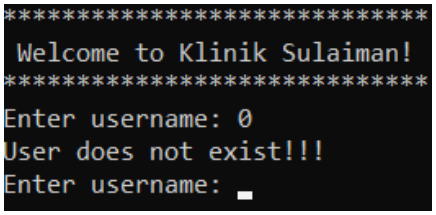
Login/Logout

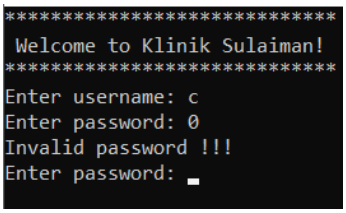
```
string user;
string pass;
struct DoctorAndNurse doctor { "a", "b" };
struct DoctorAndNurse nurse { "c", "d" };

cout << "*****************************\n";
cout << " Welcome to Klinik Sulaiman!\n";
cout << "*****************************";
cout << endl;
cout << "Enter username: ";
cin >> user;
while (!(user == doctor.userID || user == nurse.userID)) {
    cin.ignore();
    cout << "User does not exist!!!" << endl;
    cout << "Enter username: ";
    cin >> user;
}
cin.ignore();

cout << "Enter password: ";
cin >> pass;
while (!((user == doctor.userID && pass == doctor.password) || (user == nurse.userID && pass == nurse.password))) {
    cin.ignore();
    cout << "Invalid password !!!" << endl;
    cout << "Enter password: ";
    cin >> pass;
}
if (user == nurse.userID && pass == nurse.password) {
    log = 1;
}
else if (user == doctor.userID && pass == doctor.password) {
    log = 2;
}
cin.ignore();
```

The above code block is used to provide access to Klinik Sulaiman Patients' Queue based Management System. Upon starting the system, the user will be greeted with a welcome messaged where they must key in the username and password for their profile as shown below.



The credentials would be validated to see if the user profile exists in the system. If the username or password does not exist, an invalid message would pop up, alerting the user to key in the correct credentials as shown below.





```
switch (log)
{
case 1:
    system("cls");
    while (choice != 0)
    {
        cout << " _____________________________________________________________________________\n";
        cout << "|     Klinik Sulaiman Patient's Queue Bases Management System: Nurse Access   |\n";
        cout << "|-----------------------------------------------------------------------------|\n";
        cout << "|                                  Menu Options:                              |" << endl;
        cout << "|-----------------------------------------------------------------------------|\n";
        cout << "|                                                                             |\n";
        cout << "|1. Add a new patient on the waiting list                                     |" << endl;
        cout << "|2. View all patients on the original waiting list                            |" << endl;
        cout << "|3. View all patients on the original visit history list                      |" << endl;
        cout << "|4. Calling the patient to be treated                                         |" << endl;
        cout << "|5. Search patient from the waiting list based on Patient ID or First Name    |" << endl;
        cout << "|6. Sort the waiting list by patient's current visit time, display in ascending order|" << endl;
        cout << "|7. Logout                                                                    |" << endl;
        cout << "|_____________________________________________________________________________|\n";
        cout << "\nSelect your option: ";
        cin >> choice;
        cin.ignore();

        switch (choice)
        {
        case 1:
            system("cls");
            cout << "**************\n";
            cout << " Add Patients\n";
            cout << "**************\n";
            addPatient();
            break;
        case 2:
            system("cls");
            viewWaitingPatients();
            break;
        case 3:
            system("cls");
            viewVisitHistoryPatients();
            break;
        case 4:
            system("cls");
            callPatients();
            break;
        case 5:
            system("cls");
            searchWaitingList();
            break;
        case 6:
            system("cls");
            sortWaitingList();
            break;
        case 7:
            login();
            break;
        }
        system("cls");
    }
    break;
```

The above code block is used to display Nurse Access Menu that is used by the nurses in Klinik Sulaiman.

```
case 2:
    system("cls");
    while (choice != 0)
    {
        cout << " _____________________________________________________________________________________________\n";
        cout << "|     Klinik Sulaiman Patient's Queue Bases Management System: Doctor Access                 |\n";
        cout << "|------------------------------------------------------------------------------------------|\n";
        cout << "|                                    Menu Options:                                         |" << endl;
        cout << "|------------------------------------------------------------------------------------------|\n";
        cout << "|                                                                                          |\n";
        cout << "|1. View all patients on the original waiting list                                         |" << endl;
        cout << "|2. View all patients on the original visit history list                                   |" << endl;
        cout << "|3. Search Specific Patient from the patient's visit history and modify patient records    |" << endl;
        cout << "|4. Sort and display all records from the patient's visit history list in descending order  |" << endl;
        cout << "|5. Search patients from the patient's visit history list based on sickness description or firstname   |" << endl;
        cout << "|6. Logout                                                                                 |" << endl;
        cout << "|__________________________________________________________________________________________|\n";
        cout << "\nSelect your option: ";
        cin >> choice;
        cin.ignore();


        switch (choice) {
        case 1:
            system("cls");
            viewWaitingPatients();
            break;
        case 2:
            system("cls");
            viewVisitHistoryPatients();
            break;
        case 3:
            system("cls");
            searchAndModifyVisitHistory();
            break;
        case 4:
            system("cls");
            sortVisitHistory();
            break;
        case 5:
            system("cls");
            searchVisitHistory();
            break;
        case 6:
            login();
            break;
        default:
            cout << "Invalid option";
            system("cls");
        }

    }
    break;
}
```

The above code block is used to display Doctor Access Menu that is used by the Doctors in Klinik Sulaiman.

Depending on the type of user, different display would be output based on the credentials. The login uses a switch case, where if the user is a nurse, the case would be set to “1” causing the system to output the Nurse Access Menu whereas if the user is a docotr, the case would be set to “2” causing the system to output the Doctor Access Menu.

```
 ____________________________________________________________________________
|     Klinik Sulaiman Patient's Queue Bases Management System: Nurse Access  |
|----------------------------------------------------------------------------|
|                                  Menu Options:                             |
|----------------------------------------------------------------------------|
|                                                                            |
|1. Add a new patient on the waiting list                                    |
|2. View all patients on the original waiting list                           |
|3. Calling the patient to be treated                                        |
|4. Search patient from the waiting list based on Patient ID or First Name   |
|5. Sort the waiting list by patient's current visit time, display in ascending order|
|6. Logout                                                                   |
|____________________________________________________________________________|

Select your option:
```

```
 ____________________________________________________________________________
|     Klinik Sulaiman Patient's Queue Bases Management System: Doctor Access |
|----------------------------------------------------------------------------|
|                                  Menu Options:                             |
|----------------------------------------------------------------------------|
|                                                                            |
|1. View all patients on the original waiting list                           |
|2. Search Specific Patient from the patient's visit history and modify patient records|
|3. Sort and display all records from the patient's visit history list in descending order|
|4. Search patients from the patient's visit history list based on sickness description or firstname|
|5. Logout                                                                   |
|____________________________________________________________________________|

Select your option:
```

Nurse

Add Patients according to the disability option priority on the Waiting List

```
void insertWaitingPatient(string patientID, string firstName, string lastName,
    int age, char gender, string phoneNo, string address, string sicknessDescription,
    string visitDate, string visitTime, char disabilityOption)
{
    newNode = new Patient;
    newNode->patientID = patientID;
    newNode->firstName = firstName;
    newNode->lastName = lastName;
    newNode->age = age;
    newNode->gender = gender;
    newNode->phoneNo = phoneNo;
    newNode->address = address;
    newNode->sicknessDescription = sicknessDescription;
    newNode->visitDate = visitDate;
    newNode->visitTime = visitTime;
    newNode->disabilityOption = disabilityOption;
    newNode->doctorName = "NULL";
    newNode->medicineInformation = "NULL";
    newNode->nextAddress = NULL;
    newNode->prevAddress = NULL;

    //situation 1: list is empty (adding any patients)
    if (waitingHead == NULL)
    {
        waitingHead = waitingTail = newNode;
    }
    else
    {
        // Check whether the new patient is disabled or not
        if (newNode->disabilityOption == 'Y') {
            if (waitingHead->disabilityOption == 'Y') {
                //situation 2: list with disabled patients (adding disabled patients)
                current = waitingHead->nextAddress;
                // Loop the waiting list until find the non-disabled patient
                while (current->disabilityOption == 'Y') {
                    current = current->nextAddress;
                }
                current->prevAddress->nextAddress = newNode;
                newNode->prevAddress = current->prevAddress;
                current->prevAddress = newNode;
                newNode->nextAddress = current;
            }
            else {
                //situation 3: list without disabled patients (adding disabled patients)
                // If the first patient is not disabled, directly add the new disabled patient in the first node
                newNode->nextAddress = waitingHead;
                waitingHead->prevAddress = newNode;
                waitingHead = newNode;
            }
        }
        else {
            //situation 4: list not empty (adding normal patients)
            //Insert from the end in waiting list
            newNode->prevAddress = waitingTail;
            waitingTail->nextAddress = newNode;
            waitingTail = newNode;
        }
    }
    ::index++;
    ::waitingPatientSize++;
}
```

The above code block is used to add the patient into the waiting list together with their details. First, a newNode is created for the patient. Then, the details of the patient is read and stored based on the input from the nurse. The list is checked to see if it is empty. If the list is empty, the waitingHead and waitingTail is reffered to the newNode. If the list isn't empty, the disability of the patient is checked. If the patient is not disabled, the prevAddress would be referred to the newNode, the tail would be referred to the nextAddress. Finally, the tail will be assigned to the newNode. This will cause the patient to be added to the end of the list.

```
void addPatient() {
    int decision = 1;

    string patientID;
    string firstName;
    string lastName;
    int age;
    char gender;
    string phoneNo;
    string address;
    string sicknessDescription;
    string visitDate;
    string visitTime;
    char disabilityOption;

    regex n("[A-Za-z]+");
    regex p("^(\\d{3}[- .]?){2}\\d{4}$");
    regex d("([0-9]{2})/([0-9]{2})/([0-9]{4})");
    regex t("([01]?[0-9]|2[0-3]):[0-5][0-9]");
    regex k("[A-Za-z1-9.,]+");

    while (decision != 0)
    {
        patientID = "P" + to_string((::index + 1));
        cout << "Enter the Patient's First Name: ";
        cin >> firstName;
        while (!regex_match(firstName, n) || firstName.empty()) {
            cout << endl << "Invalid Name !!!" << endl;
            cout << "Enter the Patient's First Name: ";
            cin.ignore();
            cin >> firstName;
        }
        cin.ignore();

        cout << endl << "Enter the Patient's Last Name: ";
        cin >> lastName;
        while (!regex_match(lastName, n) || lastName.empty()) {
            cout << endl << "Invalid Name !!!" << endl;
            cout << "Enter the Patient's Last Name: ";
            cin.ignore();
            cin >> lastName;
        }
        cin.ignore();

        cout << endl << "Enter the Patient's Age: ";
        bool valid = false;
        do {
            cin >> age;
            if (cin.fail() || age <= 0) {
                cout << endl << "Invalid Age!!!" << endl;
                cout << "Enter the Patient's Age: ";
                cin.clear();
                cin.ignore();
            }
            else {
                valid = true;
            }
        } while (!valid);
        cin.ignore();
```

```
        cout << endl << "Enter the Patient's Gender ('M' for Male, 'F' for Female): ";
        cin >> gender;
        gender = toupper(gender);
        while (!((gender == 'M') || (gender == 'F'))) {
            cout << endl << "Invalid Gender !!!" << endl;
            cout << "Enter the Patient's Gender ('M' for Male, 'F' for Female): ";
            cin.ignore();
            cin >> gender;
            gender = toupper(gender);
        }
        cin.ignore();

        cout << endl << "Enter the Patient's Phone Number: ";
        cin >> phoneNo;
        while (!regex_match(phoneNo, p) || phoneNo.empty()) {
            cout << endl << "Invalid Phone Number !!!" << endl;
            cout << "Enter the Patient's Phone Number: ";
            cin.ignore();
            cin >> phoneNo;
        }
        cin.ignore();

        cout << endl << "Enter the Patient's Address: ";
        getline(cin, address);
        while (address.empty() || !regex_match(address, k)) {
            cout << endl << "Invalid Address !!!" << endl;
            cout << "Enter the Patient's Address: ";
            getline(cin, address);
        }

        cout << endl << "Enter Description about Patient's Sickness: ";
        getline(cin, sicknessDescription);
        while (sicknessDescription.empty() || !regex_match(sicknessDescription, k)) {
            cout << endl << "Invalid sickness description !!!" << endl;
            cout << "Enter Description about Patient's Sickness: ";
            getline(cin, sicknessDescription);
        }

        cout << endl << "Enter the Patient's Visit Date (Example = 01/02/2000): ";
        cin >> visitDate;
        while (!regex_match(visitDate, d) || visitDate.empty()) {
            cout << endl << "Invalid Date !!!" << endl;
            cout << "Enter the Patient's Visit Date (Example = 01/02/2000): ";
            cin.ignore();
            cin >> visitDate;
        }
        cin.ignore();

        cout << endl << "Enter the Patient's Visit Time (Example = 22.22): ";
        cin >> visitTime;
        while (!regex_match(visitTime, t) || visitTime.empty()) {
            cout << endl << "Invalid Time !!!" << endl;
            cout << "Enter the Patient's Visit Time (Example = 22.22): ";
            cin.ignore();
            cin >> visitTime;
        }
        cin.ignore();
    cout << endl << "Is the Patient disabled ? ('Y' for Yes, 'N' for No): ";
    cin >> disabilityOption;
    disabilityOption = toupper(disabilityOption);
    while (!((disabilityOption == 'Y') || (disabilityOption == 'N'))) {
        cout << endl << "Invalid Option !!!" << endl;
        cout << "Is the Patient disabled ? ('Y' for Yes, 'N' for No): ";
        cin.ignore();
        cin >> disabilityOption;
        disabilityOption = toupper(disabilityOption);
    }
    cin.ignore();


    insertWaitingPatient(patientID, firstName, lastName, age, gender, phoneNo, address, sicknessDescription, visitDate, visitTime, disabilityOption);
    cout << endl;

    cout << "Would you like to add another patient? Type: 1 for Yes, 0 for No. \n";
    cin >> decision;
    cin.ignore();
  }
  system("cls");
}
```

The code in code block 5 is used to the read the input from the nurse about the details of the patient and add the patient to the waiting list. The input is validated to ensure that the correct patient detail is stored. Below are the output and errors for adding patient details.

```
**********
 Add Patients
**********
Enter the Patient's First Name: John

Enter the Patient's Last Name: Mayer

Enter the Patient's Age: 22

Enter the Patient's Gender ('M' for Male, 'F' for Female): M

Enter the Patient's Phone Number: 012-222-3456

Enter the Patient's Address: No.12, Jalan Ampang

Enter Description about Patient's Sickness: Fever

Enter the Patient's Visit Date (Example = 01/02/2000): 12/12/2021

Enter the Patient's Visit Time (Example = 22.22): 22.22

Is the Patient disabled ? ('Y' for Yes, 'N' for No): N

Would you like to add another patient? Type: 1 for Yes, 0 for No.
```

```
Invalid Name !!!
Enter the Patient's First Name:
```

```
Invalid Name !!!
Enter the Patient's Last Name:
```

```
Invalid Age!!!
Enter the Patient's Age:
```

```
Invalid Gender !!!
Enter the Patient's Gender ('M' for Male, 'F' for Female):
```

```
Invalid Phone Number !!!
Enter the Patient's Phone Number:
```

```
Invalid Address !!!
Enter the Patient's Address:
```

```
Invalid sickness description !!!
Enter Description about Patient's Sickness:
```

```
Invalid Date !!!
Enter the Patient's Visit Date (Example = 01/02/2000):
```

```
Invalid Time !!!
Enter the Patient's Visit Time (Example = 22.22):
```

```
Invalid Option !!!
Is the Patient disabled ? ('Y' for Yes, 'N' for No):
```

View Patients on Waiting List

```
void viewWaitingPatients()
{
    current = waitingHead;

    cout << "***************\n";
    cout << " View Patients\n";
    cout << "***************\n\n";

    while (current != NULL)
    {

        cout << "===============" << endl;
        cout << "Patient ID: " << current->patientID << endl;
        cout << "===============" << endl;
        cout << "First Name: " << current->firstName << "  " << "Last Name: " << current->lastName << endl;
        cout << "Age: " << current->age << endl;
        cout << "Gender: " << current->gender << endl;
        cout << "Phone Number: " << current->phoneNo << endl;
        cout << "Address: " << current->address << endl;
        cout << "Sickness Description: " << current->sicknessDescription << endl;
        cout << "Visit Date: " << current->visitDate << "  " << "Visit Time: " << current->visitTime << endl;
        cout << "Disability: " << current->disabilityOption << endl;
        cout << "Doctor's Name: " << current->doctorName << endl;
        cout << "Medicine Information: " << current->medicineInformation << endl << endl;

        current = current->nextAddress;
    }

    cout << "End of Patient List!" << endl;
    system("pause");
    system("cls");
}
```

The code block above is used to view all the patients that are in the waiting list. First, the current pointer is referred to the waitingHead pointer to allow the system to read the list. The current pointer is checked to see if its NULL, indicating that it has come to end of the list. If the current pointer is not NULL, the system will display the patients the waiting list including their details. After the patient's information is displayed, the current pointer will be moved to the nextAddress for the next patient. This process will be looped till all the patients are displayed or when the current pointer has reached till the end. When the pointer has reached the end of the list, a message "End of Patient List ! " will be displayed showing that there are no more patients to be displayed and the function is stopped. Below is the output for viewing all patients.

```
***************
 View Patients
***************

===============
Patient ID: P6
===============
First Name: Devi  Last Name: Darika
Age: 12
Gender: F
Phone Number: 0198765432
Address: No6, Jalan 6, Taman PQR
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.30
Disability: Y
Doctor's Name: NULL
Medicine Information: NULL

===============
Patient ID: P4
===============
First Name: Lim  Last Name: Ah Kao
Age: 28
Gender: M
Phone Number: 0123456789
Address: No4, Jalan 4, Taman JKL
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.50
Disability: N
Doctor's Name: NULL
Medicine Information: NULL

===============
Patient ID: P5
===============
First Name: Mohammad  Last Name: Ali
Age: 40
Gender: M
Phone Number: 0129876543
Address: No5, Jalan 5, Taman MNO
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.40
Disability: N
Doctor's Name: NULL
Medicine Information: NULL

End of Patient List!
Press any key to continue . . .
```

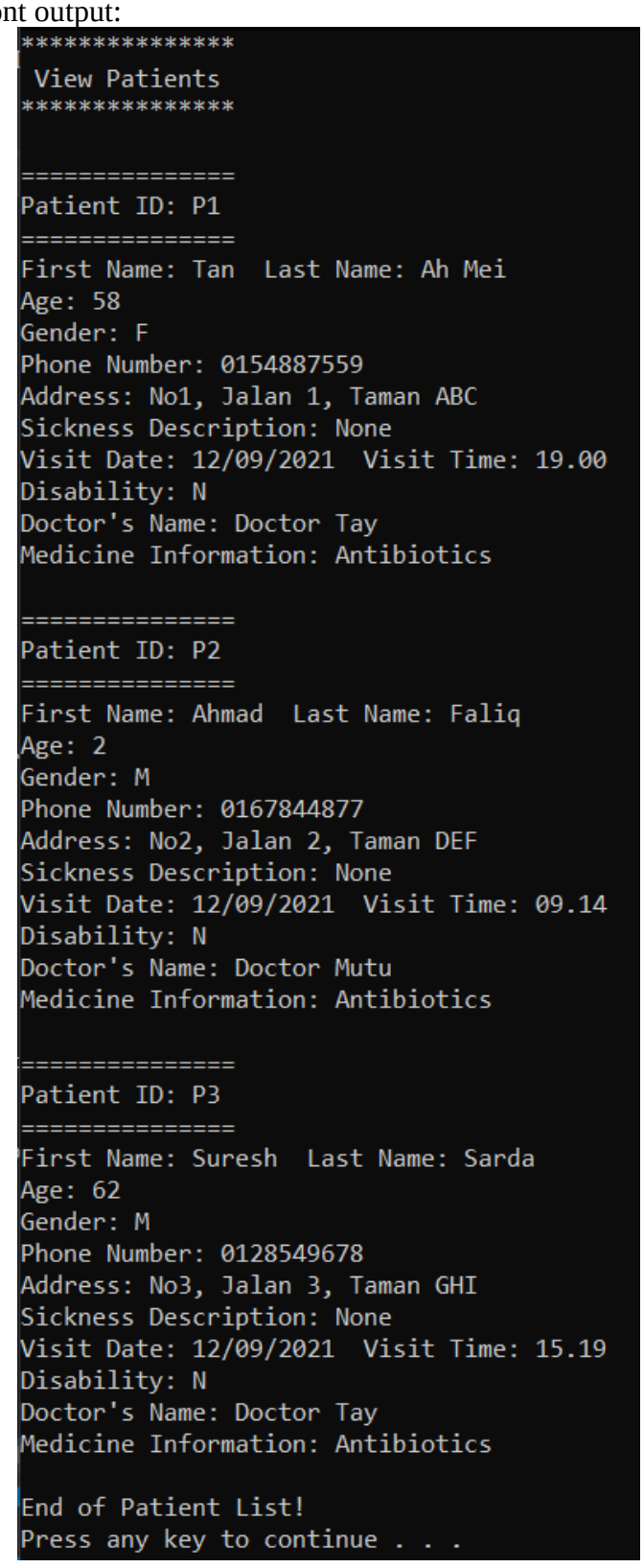
View Patients on Visit History List

Code Section:

```
void viewVisitHistoryPatients()
{
    current = visitHistoryHead;

    cout << "***************\n";
    cout << " View Patients\n";
    cout << "***************\n\n";

    while (current != NULL)
    {

        cout << "===============" << endl;
        cout << "Patient ID: " << current->patientID << endl;
        cout << "===============" << endl;
        cout << "First Name: " << current->firstName << "  " << "Last Name: " << current->lastName << endl;
        cout << "Age: " << current->age << endl;
        cout << "Gender: " << current->gender << endl;
        cout << "Phone Number: " << current->phoneNo << endl;
        cout << "Address: " << current->address << endl;
        cout << "Sickness Description: " << current->sicknessDescription << endl;
        cout << "Visit Date: " << current->visitDate << "  " << "Visit Time: " << current->visitTime << endl;
        cout << "Disability: " << current->disabilityOption << endl;
        cout << "Doctor's Name: " << current->doctorName << endl;
        cout << "Medicine Information: " << current->medicineInformation << endl << endl;

        current = current->nextAddress;
    }

    cout << "End of Patient List!" << endl;
    system("pause");
    system("cls");
}
```

This function is used to display all the patients on the visit history list in a single page. First of all, current pointer refers to the visitHistoryHead which is the first node in the visit history list. Display the patient details inside the current pointer. Loop the visit history list by refer current to current's next address until the current equal to Null value, which means it comes to the end of the list. After ended the loop, the system will pause until users press any key to continue. The system will clear the screen before displaying the return to the main menu.

Display from the front output:



This is the default patients' detail in the visit history list while running the program. When new patient comes in, it will directly add the patient's details at the end of the list.

Calling the Patient to be Treated

Code Section:

```
void callPatients() {
    //Delete from the front in waiting list
    //situation 1: list is empty
    if (waitingHead == NULL) {
        cout << "There is no patient on the waiting list." << endl;
        system("pause");
        return;
    }
    //situation 2: list not empty
    current = waitingHead;
    waitingHead = waitingHead->nextAddress;

    //situation3: determine whether the current head is it equal to NULL value
    if (waitingHead == NULL) {
        waitingTail = NULL;
    }
    else {
        waitingHead->prevAddress = NULL;
    }
    current->nextAddress = NULL;
    current->prevAddress = NULL;

    //Insert from the end in visit history list
    //situation 1: list empty
    if (visitHistoryHead == NULL)
    {
        visitHistoryHead = visitHistoryTail = current;
    }
    //situation 2: list not empty
    else
    {
        current->prevAddress = visitHistoryTail;
        visitHistoryTail->nextAddress = current;
        visitHistoryTail = current;
    }
    ::waitingPatientSize--;
    ::visitHistorySize++;

    cout << "The patient of " << current->patientID << " is called to be treated." << endl;
    cout << "Waiting List Patient: " << waitingPatientSize << endl;
    cout << "Visit History List Patient: " << visitHistorySize << endl;
    system("pause");
}
```

This function is used to calling the patient to be treated, which includes two steps. The first one is deleted the patient from the front on the waiting list; the second one is inserted that particular patient from the end on the visit history list.

First of all, the program will check whether the waitingHead is empty or not. If waitingHead is empty, it will display the output “There is no patient on the waiting list” and return to the main menu. On the other hand, it will store the waitingHead (first node of waiting list) in the current pointer and refer waitingHead to the waitingHead’s next address. After that, check again whether the waitingHead is empty or not. If waitingHead is empty, refer waitingTail to null value; else, refer waitingHead’s previous address to null value. This process will ensure the waiting list is still arranged in neat although the first node is taken out from the list. After that, make sure the current’s next and previous address are equal to null value to make it become a single node.

Then, check the visitHistoryHead is empty or not. If visitHistoryHead is empty, it will directly make the visitHistoryHead and visitHistoryTail refer to the current pointer and make it as the first node. On the other hand, it will make the current’s previous address to the visitHistoryTail and visitHistoryTail’s next address to the current pointer. After that, refer visitHistoryTail to the current pointer. This process will ensure the visit history list is still arranged in neat although added a new node.

Lastly, decrease the size of waiting list and increase the size of visit history list by 1.

Delete from the front in waiting list & insert into the end in visit history list:

Before calling

Waiting List

```
**************
 View Patients
**************

==============
Patient ID: P6
==============
First Name: Devi  Last Name: Darika
Age: 12
Gender: F
Phone Number: 0198765432
Address: No6, Jalan 6, Taman PQR
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.30
Disability: Y
Doctor's Name: NULL
Medicine Information: NULL

==============
Patient ID: P4
==============
First Name: Lim  Last Name: Ah Kao
Age: 28
Gender: M
Phone Number: 0123456789
Address: No4, Jalan 4, Taman JKL
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.50
Disability: N
Doctor's Name: NULL
Medicine Information: NULL

==============
Patient ID: P5
==============
First Name: Mohammad  Last Name: Ali
Age: 40
Gender: M
Phone Number: 0129876543
Address: No5, Jalan 5, Taman MNO
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.40
Disability: N
Doctor's Name: NULL
Medicine Information: NULL

End of Patient List!
Press any key to continue . . .
```

Visit History List

```
**************
 View Patients
**************

==============
Patient ID: P1
==============
First Name: Tan  Last Name: Ah Mei
Age: 58
Gender: F
Phone Number: 0154887559
Address: No1, Jalan 1, Taman ABC
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 19.00
Disability: N
Doctor's Name: Doctor Tay
Medicine Information: Antibiotics

==============
Patient ID: P2
==============
First Name: Ahmad  Last Name: Faliq
Age: 2
Gender: M
Phone Number: 0167844877
Address: No2, Jalan 2, Taman DEF
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 09.14
Disability: N
Doctor's Name: Doctor Mutu
Medicine Information: Antibiotics

==============
Patient ID: P3
==============
First Name: Suresh  Last Name: Sarda
Age: 62
Gender: M
Phone Number: 0128549678
Address: No3, Jalan 3, Taman GHI
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 15.19
Disability: N
Doctor's Name: Doctor Tay
Medicine Information: Antibiotics

End of Patient List!
Press any key to continue . . .
```

```
The patient of P6 is called to be treated.
Waiting List Patient: 2
Visit History List Patient: 4
Press any key to continue . . .
```

After calling

Waiting List

```
**************
 View Patients
**************

===============
Patient ID: P4
===============
First Name: Lim  Last Name: Ah Kao
Age: 28
Gender: M
Phone Number: 0123456789
Address: No4, Jalan 4, Taman JKL
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.50
Disability: N
Doctor's Name: NULL
Medicine Information: NULL

===============
Patient ID: P5
===============
First Name: Mohammad  Last Name: Ali
Age: 40
Gender: M
Phone Number: 0129876543
Address: No5, Jalan 5, Taman MNO
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.40
Disability: N
Doctor's Name: NULL
Medicine Information: NULL

End of Patient List!
Press any key to continue . . .
```

Visit History List

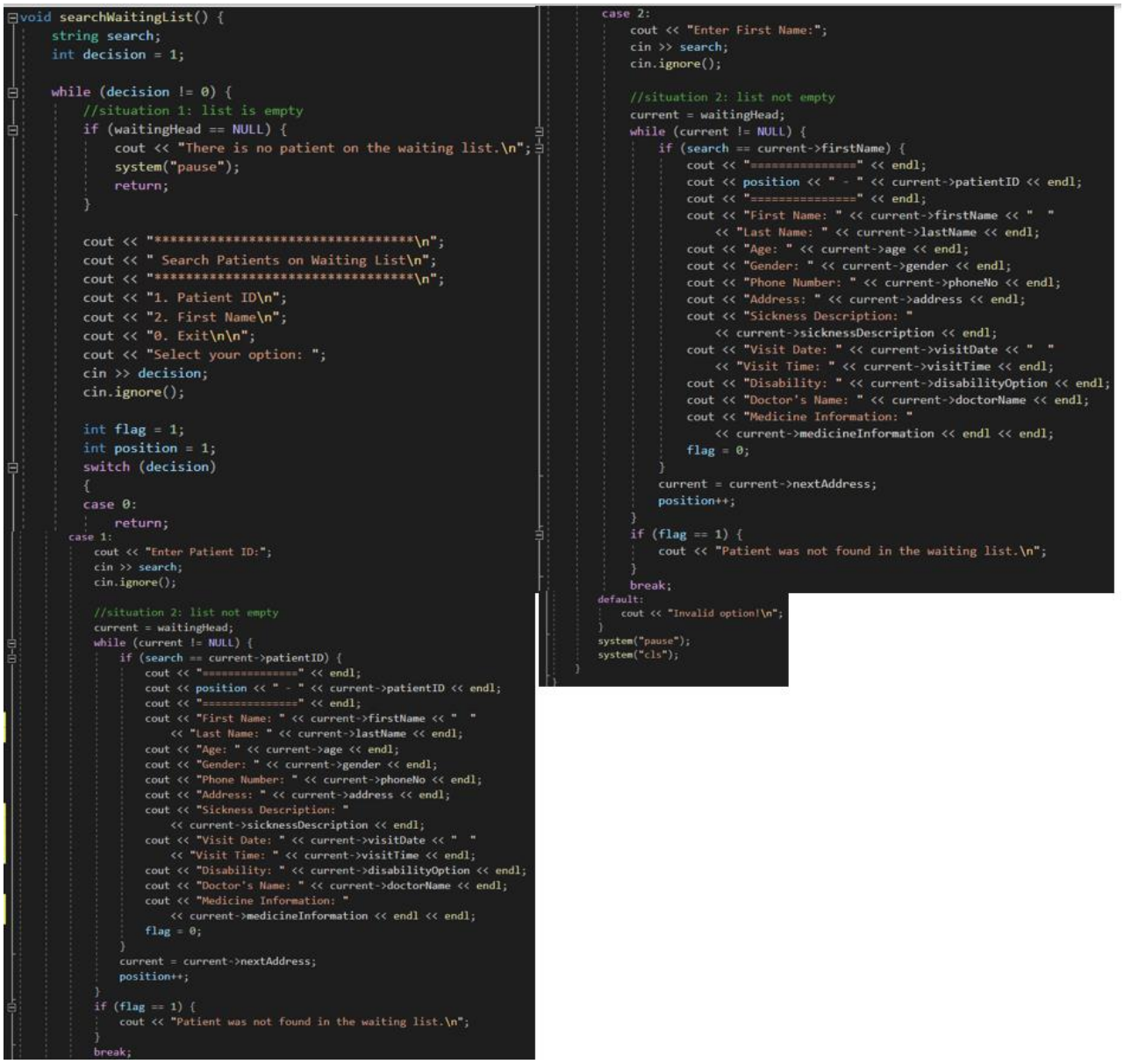
```
**************
 View Patients
**************

===============
Patient ID: P1
===============
First Name: Tan  Last Name: Ah Mei
Age: 58
Gender: F
Phone Number: 0154887559
Address: No1, Jalan 1, Taman ABC
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 19.00
Disability: N
Doctor's Name: Doctor Tay
Medicine Information: Antibiotics

===============
Patient ID: P2
===============
First Name: Ahmad  Last Name: Faliq
Age: 2
Gender: M
Phone Number: 0167844877
Address: No2, Jalan 2, Taman DEF
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 09.14
Disability: N
Doctor's Name: Doctor Mutu
Medicine Information: Antibiotics

===============
Patient ID: P3
===============
First Name: Suresh  Last Name: Sarda
Age: 62
Gender: M
Phone Number: 0128549678
Address: No3, Jalan 3, Taman GHI
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 15.19
Disability: N
Doctor's Name: Doctor Tay
Medicine Information: Antibiotics
===============
Patient ID: P6
===============
First Name: Devi  Last Name: Darika
Age: 12
Gender: F
Phone Number: 0198765432
Address: No6, Jalan 6, Taman PQR
Sickness Description: None
Visit Date: 13/09/2021  Visit Time: 09.30
Disability: Y
Doctor's Name: NULL
Medicine Information: NULL

End of Patient List!
Press any key to continue . . .
```

```
C:\Users\System Manager\Desktop\Assignment\Debug\Assignment.exe
There is no patient on the waiting list.
Press any key to continue . . .
```

The figures above show the differences on waiting list and visit history list after called the patient to be treated. The output is also included the number of patients in the waiting list and visit history list. If the waiting list is empty, the program will directly display “There is no patient on the waiting list.”

Search Patient on the Waiting List

Code Section:

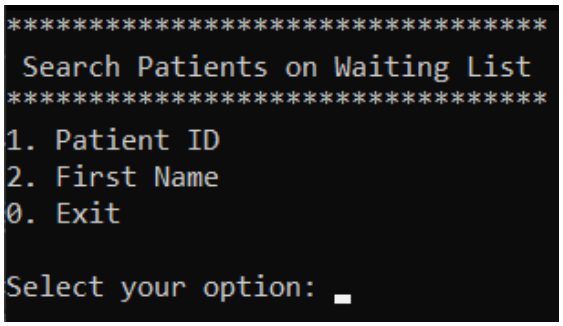


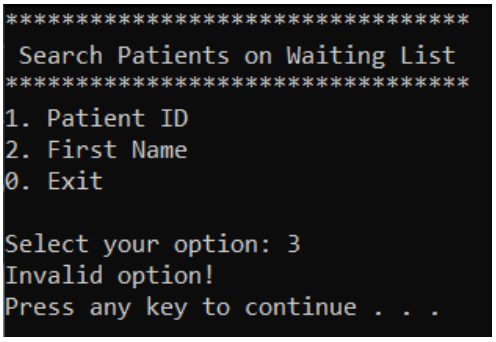
```
void searchWaitingList() {
    string search;
    int decision = 1;

    while (decision != 0) {
        //situation 1: list is empty
        if (waitingHead == NULL) {
            cout << "There is no patient on the waiting list.\n";
            system("pause");
            return;
        }

        cout << "*******************************\n";
        cout << " Search Patients on Waiting List\n";
        cout << "*******************************\n";
        cout << "1. Patient ID\n";
        cout << "2. First Name\n";
        cout << "0. Exit\n\n";
        cout << "Select your option: ";
        cin >> decision;
        cin.ignore();

        int flag = 1;
        int position = 1;
        switch (decision)
        {
        case 0:
            return;
        case 1:
            cout << "Enter Patient ID:";
            cin >> search;
            cin.ignore();

            //situation 2: list not empty
            current = waitingHead;
            while (current != NULL) {
                if (search == current->patientID) {
                    cout << "==============" << endl;
                    cout << position << " - " << current->patientID << endl;
                    cout << "==============" << endl;
                    cout << "First Name: " << current->firstName << "  "
                        << "Last Name: " << current->lastName << endl;
                    cout << "Age: " << current->age << endl;
                    cout << "Gender: " << current->gender << endl;
                    cout << "Phone Number: " << current->phoneNo << endl;
                    cout << "Address: " << current->address << endl;
                    cout << "Sickness Description: "
                        << current->sicknessDescription << endl;
                    cout << "Visit Date: " << current->visitDate << "  "
                        << "Visit Time: " << current->visitTime << endl;
                    cout << "Disability: " << current->disabilityOption << endl;
                    cout << "Doctor's Name: " << current->doctorName << endl;
                    cout << "Medicine Information: "
                        << current->medicineInformation << endl << endl;
                    flag = 0;
                }
                current = current->nextAddress;
                position++;
            }
            if (flag == 1) {
                cout << "Patient was not found in the waiting list.\n";
            }
            break;
        case 2:
            cout << "Enter First Name:";
            cin >> search;
            cin.ignore();

            //situation 2: list not empty
            current = waitingHead;
            while (current != NULL) {
                if (search == current->firstName) {
                    cout << "==============" << endl;
                    cout << position << " - " << current->patientID << endl;
                    cout << "==============" << endl;
                    cout << "First Name: " << current->firstName << "  "
                        << "Last Name: " << current->lastName << endl;
                    cout << "Age: " << current->age << endl;
                    cout << "Gender: " << current->gender << endl;
                    cout << "Phone Number: " << current->phoneNo << endl;
                    cout << "Address: " << current->address << endl;
                    cout << "Sickness Description: "
                        << current->sicknessDescription << endl;
                    cout << "Visit Date: " << current->visitDate << "  "
                        << "Visit Time: " << current->visitTime << endl;
                    cout << "Disability: " << current->disabilityOption << endl;
                    cout << "Doctor's Name: " << current->doctorName << endl;
                    cout << "Medicine Information: "
                        << current->medicineInformation << endl << endl;
                    flag = 0;
                }
                current = current->nextAddress;
                position++;
            }
            if (flag == 1) {
                cout << "Patient was not found in the waiting list.\n";
            }
            break;
        default:
            cout << "Invalid option!\n";
        }
        system("pause");
        system("cls");
    }
}
```

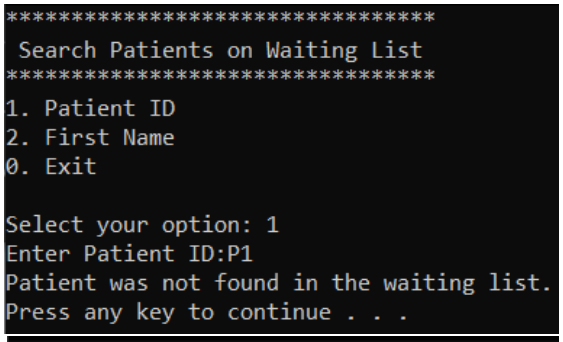
This function is used to search the patient according to their patient ID or first name on the waiting list. Firstly, a new string “search” will be defined at the beginning to store the input from users. The new integer “decision” is used to accept the input from users to decide the type of search they want to perform. The program will check whether the waitingHead is empty or not. If waitingHead is empty, it will display the output “There is no patient on the waiting list” and return to the main menu. Input 1 for decision variable will lead users to search patient based on patient id, input 2 for decision variable will lead users to search patient based on first name, input 0 for decision variable will bring users back to the main menu. After input for the search variable, the program will check the variable is equal to patient ID or first name in the

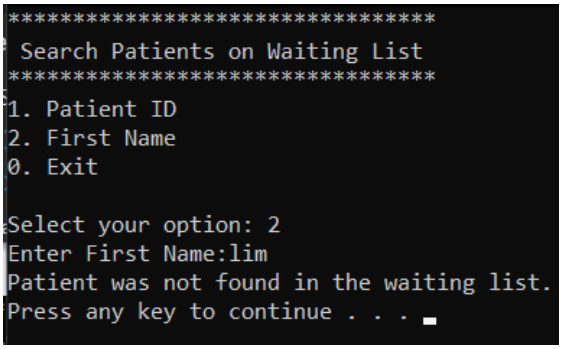
current pointer or not. If the search is matched with them, it will display all the patient details and the position of that patient. Else, it will display “Patient was not found in the waiting list”.

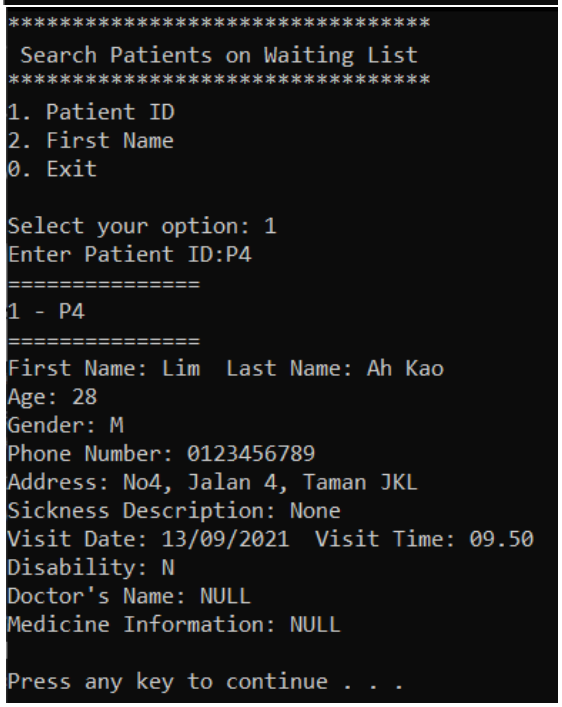
Linear Search:

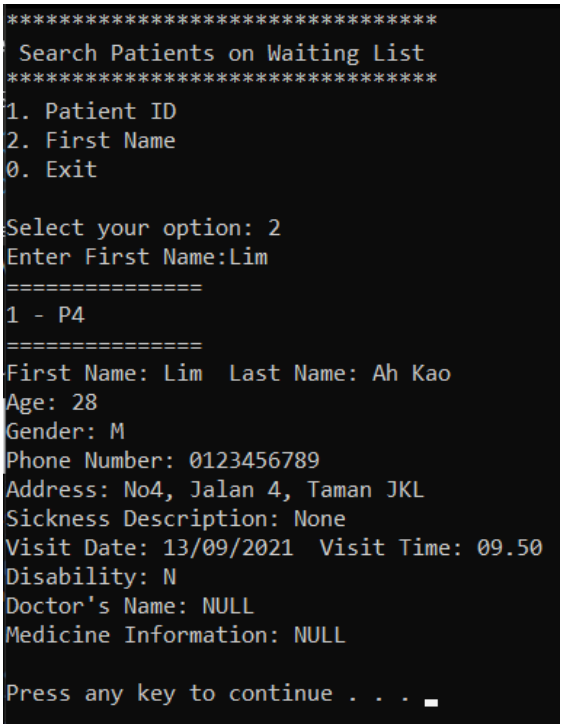


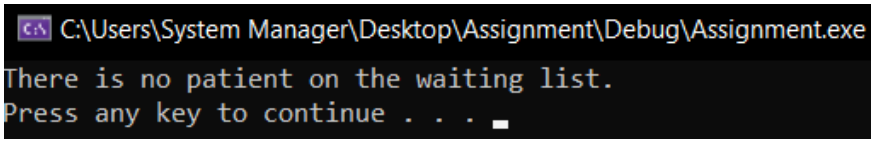












The search function is case-sensitive. If the input is not completely matched with the correct uppercase and lowercase character, it will display the error message instead of the patient information.

Sort Waiting List

Code Section:

```
void sortWaitingList() {
    Patient* index = NULL;
    int swapping;
    //situation 1: list is empty
    if (waitingHead == NULL || waitingHead->nextAddress == NULL) {
        cout << "There is no patient on the waiting list.\n";
        cout << "Cannot sort!" << endl;
        system("pause");
        return;
    }

    //situation 2: list not empty
    //bubble sort
    do {
        swapping = 0;
        for (current = waitingHead; current->nextAddress != NULL; current = current->nextAddress) {
            index = current->nextAddress;
            if (stod(current->visitTime) > stod(index->visitTime)) {
                swap(current->patientID, index->patientID);
                swap(current->firstName, index->firstName);
                swap(current->lastName, index->lastName);
                swap(current->age, index->age);
                swap(current->gender, index->gender);
                swap(current->phoneNo, index->phoneNo);
                swap(current->address, index->address);
                swap(current->sicknessDescription, index->sicknessDescription);
                swap(current->visitDate, index->visitDate);
                swap(current->visitTime, index->visitTime);
                swap(current->disabilityOption, index->disabilityOption);
                swapping = 1;
            }
        }
    } while (swapping);
    viewPatientsPageByPage(waitingHead);
}
```

This function is used to sort the patients on the waiting list by their visit time in ascending order. First of all, a new pointer “index” will be defined as null value. The new integer “swapping” is used to decide whether there is a swapping happens between two different nodes. Check the waitingHead and its next address is equal to null value or not. If one of them is null value, it will directly display “There is no patient on the waiting list” and “Cannot sort!” and return to the main menu. On contrast, the program will go through a loop which is starting from referring the current pointer to the waitingHead until current’s next address is equal to null value. Each increment will refer the current to the current’s next address. In each loop, refer the index pointer to the current’s next address. Its purpose is to compare the visit time between the current and the index pointer. The visit time is transferred from string data type to double data type in order to carry out the comparison process. If the current’s visit time is larger than the index’s visit time, every patient’s detail stored in current pointer will switch with the detail

stored in index's pointer and the swapping will be referred to 1. This loop process will continue running until the swapping remains 0 at the end of the loop.

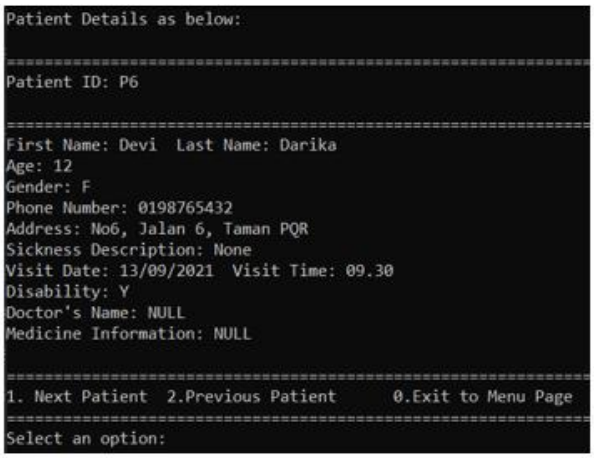
After that, the sorted waiting list will be brought to viewPatientsPageByPage function.

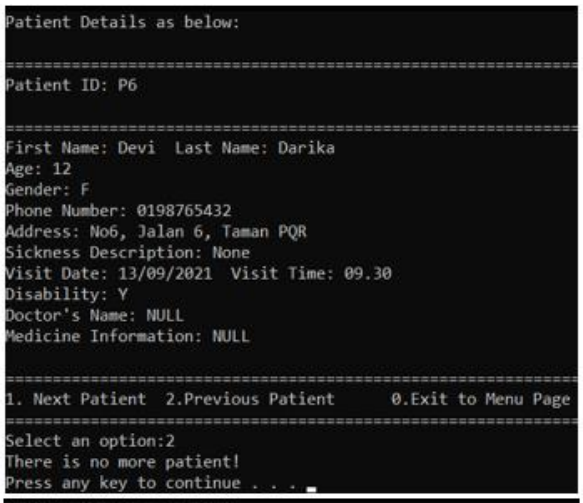
```
void viewPatientsPageByPage(Patient* listHead) {

    system("cls");
    current = listHead;

    int decision = 1;
    while (decision != 0)
    {
        cout << "Patient Details as below:" << endl;
        cout << "\n=============================================================================================================\n";
        cout << "Patient ID: " << current->patientID << endl;
        cout << "\n=============================================================================================================\n";
        cout << "First Name: " << current->firstName << "  " << "Last Name: " << current->lastName << endl;
        cout << "Age: " << current->age << endl;
        cout << "Gender: " << current->gender << endl;
        cout << "Phone Number: " << current->phoneNo << endl;
        cout << "Address: " << current->address << endl;
        cout << "Sickness Description: " << current->sicknessDescription << endl;
        cout << "Visit Date: " << current->visitDate << "  " << "Visit Time: " << current->visitTime << endl;
        cout << "Disability: " << current->disabilityOption << endl;
        cout << "Doctor's Name: " << current->doctorName << endl;
        cout << "Medicine Information: " << current->medicineInformation << endl;
        cout << "\n=============================================================================================================\n";
        cout << "1. Next Patient\t 2.Previous Patient\t 0.Exit to Menu Page \t";
        cout << "\n=============================================================================================================\n";
        cout << "Select an option:";
        cin >> decision;
        cin.ignore();
        if (decision == 1 && current->nextAddress != NULL)
        {
            current = current->nextAddress;
        }
        else if (decision == 2 && current->prevAddress != NULL)
        {
            current = current->prevAddress;
        }
        else if (decision == 0)
        {
            return;
        }
        else if (decision != 1 && decision != 2 && decision != 0)
        {
            cout << "Invalid option!\n";
            system("pause");
        }
        else
        {
            cout << "There is no more patient!" << endl;
            system("pause");
        }

        system("cls");
    }
}
```

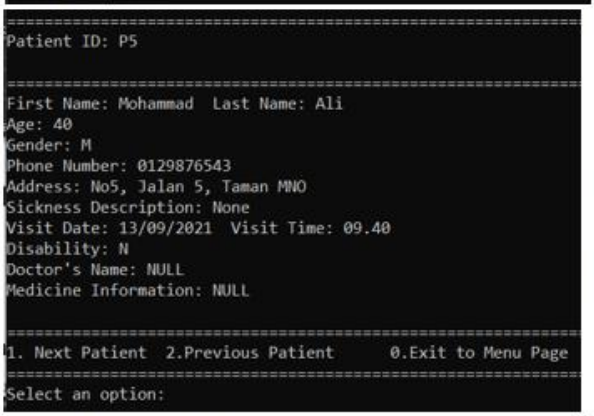
This function is used to display the patient details with page by page browsing. One page represents one patient. A parameter which is the head of list (waitingHead or visitHistoryHead) will be brought to this function. The current pointer will refer to the head pointer. The integer decision will be initialized as 1 which is used to accept the input of users to decide whether they want to go to the next patient, previous patient, or exit to the main menu. The program will continue looping until the decision is equal to 0. If the users choose 1 and current's next address is not equal to null value, the current will refer to current's next pointer and display the patient's detail of that address. If the users choose 2 and current's previous address is not equal

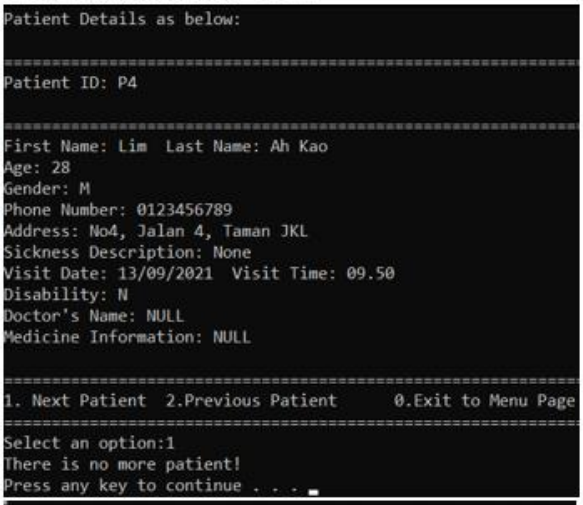
to null value, the current will refer to current's previous pointer and display the patient's detail of that address. If the input is out of the range of 0-2, it will display "Invalid option!". If all the requirement stated above is not matched, it will display "There is no more patient!".

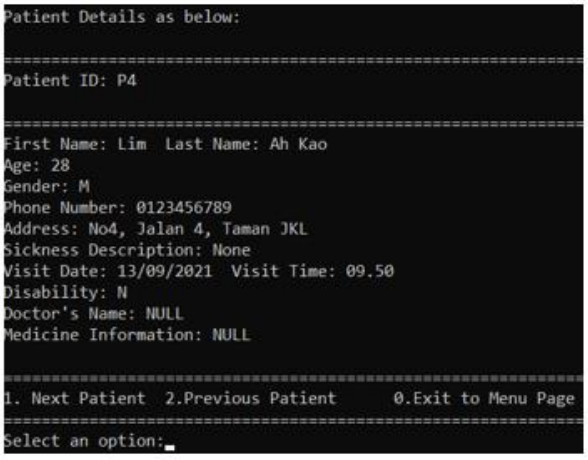
Bubble Sort:

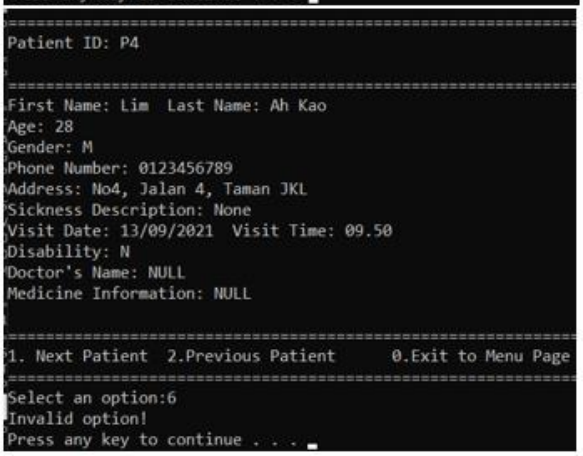


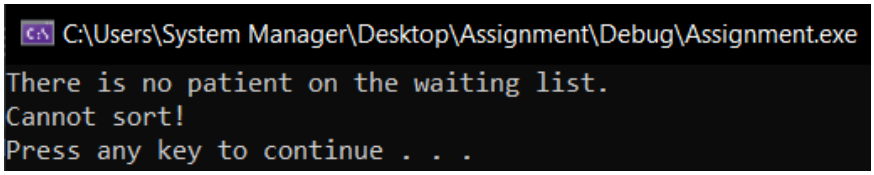












The left figures show the sequences of patients from ID “P6”, “P5”, to “P4”, while the right figures show the possible error messages. As patient ID P6 is the first node in the waiting list,

it will display “There is no more patient” without changing the page. As patient ID P4 is the last node in the waiting list, it will display “There is no more patient” without changing the page. If the users enter the input out of the range from 0-2, it will display “Invalid option!”. If the waiting list is empty, it will show the error messages and return to the main menu.

Doctor

Search and Modify Patient Record from Visit History

```
void searchAndModifyVisitHistory() {
    bool r = true;
    while (r == true) {
        bool found = false;
        while (found == false) {
            string patientID;
            cout << "Enter Patient ID: ";
            getline(cin, patientID);
            // Start searching for matching ID
            if (visitHistoryHead == NULL) {
                cout << "The patient visit history is empty!";
                return;
            }

            current = visitHistoryHead;

            while (current != NULL) {
                if (patientID == current->patientID) {
                    found = true;
                    // display
                    cout << "==============" << endl;
                    cout << "Patient ID: " << current->patientID << endl;
                    cout << "==============" << endl;
                    cout << "First Name: " << current->firstName << " " << "Last Name: " << current->lastName << endl;
                    cout << "Age: " << current->age << endl;
                    cout << "Gender: " << current->gender << endl;
                    cout << "Phone Number: " << current->phoneNo << endl;
                    cout << "Address: " << current->address << endl;
                    cout << "Sickness Description: " << current->sicknessDescription << endl;
                    cout << "Visit Date: " << current->visitDate << " " << "Visit Time: " << current->visitTime << endl;
                    cout << "Disability: " << current->disabilityOption << endl;
                    cout << "Doctor's Name: " << current->doctorName << endl;
                    cout << "Medicine Information: " << current->medicineInformation << endl << endl;
                    break;
                }
                else {
                    current = current->nextAddress;
                }
            }
            if (!found) {
                cout << "Patient ID not found!" << endl;
            }
        }
        string firstName;
        string lastName;
        int age;
        char gender;
        string phoneNo;
        string address;
        string sicknessDescription;
        string visitDate;
        string visitTime;
        char disabilityOption;
        string doctorName;
        string medicineInformation;
        char option;
        regex n("[A-Za-z]+");
        regex p("^(\\d{3}[- .]?){2}\\d{4}$");
        regex d("([0-9]{2})/([0-9]{2})/([0-9]{4})");
        regex t("([0-9]{2}).([0-9]{2})");
        regex k("[A-Za-z1-9.,]+");
        do {
            cout << "First Name: ";
            getline(cin, firstName);
            if (!(regex_match(firstName, n) || firstName.empty())) {
                cout << "Invalid first name!!!" << endl;
            }
        } while (!(regex_match(firstName, n) || firstName.empty()));
        current->firstName = firstName;
```

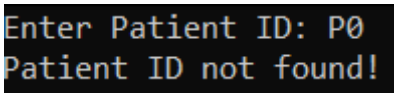
```
    do {
        cout << "Last Name: ";
        getline(cin, lastName);
        if (!(regex_match(lastName, n) || lastName.empty())) {
            cout << "Invalid last name!!!" << endl;
        }
    } while (!(regex_match(lastName, n) || lastName.empty()));
    current->lastName = lastName;
    bool valid = false;
    do {
        cout << "Age: ";
        cin >> age;
        if (cin.fail()) {
            cout << "Invalid age!!!" << endl;
            cin.clear();
            cin.ignore();
        }
        else {
            valid = true;
        }
    } while (!valid);
    current->age = age;

    do {
        cout << "Gender ('M' for Male, 'F' for Female): ";
        cin >> gender;
        if (!((gender == 'M') || (gender == 'F') || (gender == 'm') || (gender == 'f'))) {
            cout << endl << "Invalid Gender !!!" << endl;
            cout << "Gender ('M' for Male, 'F' for Female): ";
            cin >> gender;
        }
    } while (!((gender == 'M') || (gender == 'F') || (gender == 'm') || (gender == 'f')));
    current->gender = gender;
```

```
    cin.ignore();

    do {
        cout << "Phone Number: ";
        getline(cin, phoneNo);
        if (!(regex_match(phoneNo, p) || phoneNo.empty())) {
            cout << "Invalid phone number!!!" << endl;
        }
    } while (!regex_match(phoneNo, p) || phoneNo.empty());
    current->phoneNo = phoneNo;

    do {
        cout << "Address: ";
        getline(cin, address);
        if (address.empty() || !regex_match(address, k)) {
            cout << "Invalid address!!!" << endl;
        }
    } while (address.empty() || !regex_match(address, k));
    current->address = address;

    do {
        cout << "Sickness Description: ";
        getline(cin, sicknessDescription);
        if (sicknessDescription.empty() || !regex_match(sicknessDescription, k)) {
            cout << "Invalid sickness description!!!" << endl;
        }
    } while (sicknessDescription.empty() || !regex_match(sicknessDescription, k));
    current->sicknessDescription = sicknessDescription;

    do {
        cout << "Visit Date (e.g. 01/01/2020): ";
        getline(cin, visitDate);
        if (visitDate.empty() || !regex_match(visitDate, d)) {
            cout << "Invalid visit date!!!" << endl;
```

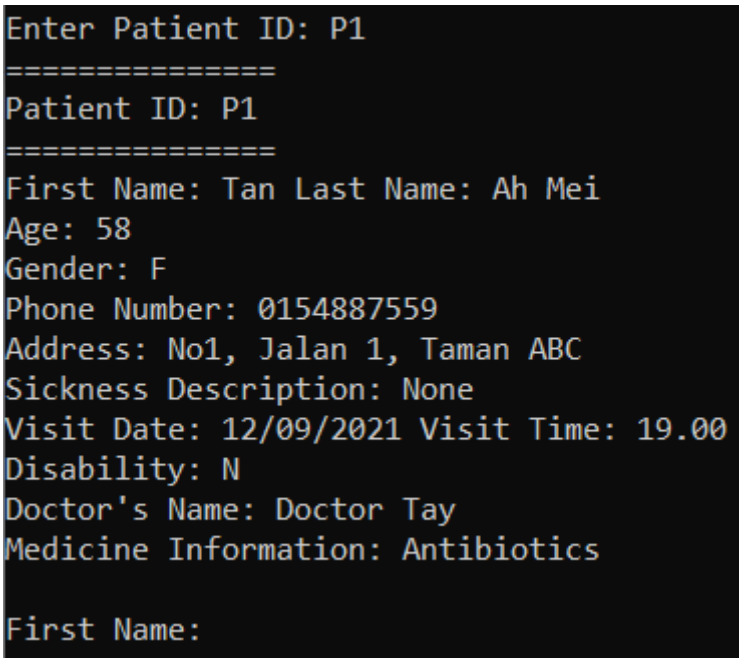
```
        }
    } while (!regex_match(visitDate, d) || visitDate.empty());
    current->visitDate = visitDate;

    do {
        cout << "Visit Time (e.g. 13.00): ";
        getline(cin, visitTime);
        if (!regex_match(visitTime, t) || visitTime.empty()) {
            cout << "Invalid visit date!!!" << endl;
        }
    } while (!regex_match(visitTime, t) || visitTime.empty());
    current->visitTime = visitTime;

    do {
        cout << "Disability (Y/N): ";
        cin >> disabilityOption;
        if (!((disabilityOption == 'Y') || (disabilityOption == 'y') || (disabilityOption == 'N') || (disabilityOption == 'n'))) {
            cout << "Invalid disability option!!!" << endl;
        }
    } while (!((disabilityOption == 'Y') || (disabilityOption == 'y') || (disabilityOption == 'N') || (disabilityOption == 'n')));
    current->disabilityOption = disabilityOption;

    cin.ignore();
    do {
        cout << "Doctor's Name: ";
        getline(cin, doctorName);
        if (!regex_match(doctorName, n) || doctorName.empty()) {
            cout << "Invalid doctor name!!!" << endl;
        }
    } while (!regex_match(doctorName, n) || doctorName.empty());
    current->doctorName = doctorName;

    do {
        cout << "Medicine Information: ";
        getline(cin, medicineInformation);
        if (medicineInformation.empty() || !regex_match(medicineInformation, k)) {
            cout << "Invalid doctor name!!!" << endl;
        }
    } while (medicineInformation.empty() || !regex_match(medicineInformation, k));
    current->medicineInformation = medicineInformation;

    cout << "==============" << endl;
    cout << "Patient ID: " << current->patientID << endl;
    cout << "==============" << endl;
    cout << "First Name: " << current->firstName << " " << "Last Name: " << current->lastName << endl;
    cout << "Age: " << current->age << endl;
    cout << "Gender: " << current->gender << endl;
    cout << "Phone Number: " << current->phoneNo << endl;
    cout << "Address: " << current->address << endl;
    cout << "Sickness Description: " << current->sicknessDescription << endl;
    cout << "Visit Date: " << current->visitDate << " " << "Visit Time: " << current->visitTime << endl;
    cout << "Disability: " << current->disabilityOption << endl;
    cout << "Doctor's Name: " << current->doctorName << endl;
    cout << "Medicine Information: " << current->medicineInformation << endl << endl;

    do {
        cout << "Continue searching? (Y/N): ";
        cin >> option;
        if (!((option == 'Y') || (option == 'y') || (option == 'N') || (option == 'n'))) {
            cout << "Invalid option!!!" << endl;
        }
    } while (!((option == 'Y') || (option == 'y') || (option == 'N') || (option == 'n')));

    if (option == 'N' || option == 'n') {
        r = false;
    }
}
system("cls");
}
```

The above snippets are the underlying codes for the doctors' function to search and modify patient records in the visit history list. This function enables doctor users to search for patient record in the visit history list by inputting patient ID, then modify said patient record. Firstly, the system prompts the user to enter a patient ID. Then, the "current" node is used to

traverse through the linked list to look for patient record with patient ID that match the input. If no patient record is found with said input patient ID, the system prompts the user to enter a patient ID again. When the patient record is found, the system displays the entire patient record (first name, last name, address, etc.). After that, the system prompts the user to enter new data into each field to modify the searched patient record. Right after entering new values into the fields, the node changes the value of its variables into corresponding new value. The entire patient record with modified values will be displayed again.



The figure above shows what happens if a non-existent ID is entered.



Patient P1 is searched and modified.

```
First Name: Teo
Last Name: Zicheng
Age: 20
Gender ('M' for Male, 'F' for Female): M
Phone Number: 1234567890
Address: abcdef
Sickness Description: Fever
Visit Date (e.g. 01/01/2020): 01/01/2020
Visit Time (e.g. 13.00): 13.45
Disability (Y/N): N
Doctor's Name: Lee
Medicine Information: Painkiller
==============
Patient ID: P1
==============
First Name: Teo Last Name: Zicheng
Age: 20
Gender: M
Phone Number: 1234567890
Address: abcdef
Sickness Description: Fever
Visit Date: 01/01/2020 Visit Time: 13.45
Disability: N
Doctor's Name: Lee
Medicine Information: Painkiller

Continue searching? (Y/N):
```

```
***************
 View Patients
***************


==============
Patient ID: P1
==============
First Name: Teo  Last Name: Zicheng
Age: 20
Gender: M
Phone Number: 1234567890
Address: abcdef
Sickness Description: Fever
Visit Date: 01/01/2020  Visit Time: 13.45
Disability: N
Doctor's Name: Lee
Medicine Information: Painkiller
```

Sort and Display All Records from Visit History List in Descending Order

```
void sortVisitHistory() {
    Patient* index = NULL;
    int swapping;
    //situation 1: list is empty
    if (visitHistoryHead == NULL) {
        cout << "There is no patient on the visit history list.\n";
        cout << "Cannot sort!" << endl;
        system("pause");
        return;
    }

    int currentDate[3];
    int indexDate[3];

    //situation 2: list not empty
    do {
        swapping = 0;
        current = visitHistoryHead;

        for (current = visitHistoryHead; current->nextAddress != NULL; current = current->nextAddress) {
            currentDate[0] = stoi(current->visitDate.substr(0, 2).erase(0, current->visitDate.substr(0, 2).find_first_not_of('0')));
            currentDate[1] = stoi(current->visitDate.substr(3, 2).erase(0, current->visitDate.substr(3, 2).find_first_not_of('0')));
            currentDate[2] = stoi(current->visitDate.substr(6, 4).erase(0, current->visitDate.substr(6, 4).find_first_not_of('0')));
            index = current->nextAddress;
            indexDate[0] = stoi(index->visitDate.substr(0, 2).erase(0, index->visitDate.substr(0, 2).find_first_not_of('0')));
            indexDate[1] = stoi(index->visitDate.substr(3, 2).erase(0, index->visitDate.substr(3, 2).find_first_not_of('0')));
            indexDate[2] = stoi(index->visitDate.substr(6, 4).erase(0, index->visitDate.substr(6, 4).find_first_not_of('0')));
            if ((currentDate[2] < indexDate[2]) || ((currentDate[2] == indexDate[2]) &&
                (currentDate[1] < indexDate[1])) || (((currentDate[2] == indexDate[2]) &&
                    (currentDate[1] == indexDate[1])) && (currentDate[0] < indexDate[0])) ||
                    ((((currentDate[2] == indexDate[2]) && (currentDate[1] == indexDate[1])) &&
                        (currentDate[0] == indexDate[0])) && (stod(current->visitTime) <= stod(index->visitTime)))) {
                swap(current->patientID, index->patientID);
                swap(current->firstName, index->firstName);
                swap(current->lastName, index->lastName);
                swap(current->age, index->age);
                swap(current->gender, index->gender);
                swap(current->phoneNo, index->phoneNo);
                swap(current->address, index->address);
                swap(current->sicknessDescription, index->sicknessDescription);
                swap(current->visitDate, index->visitDate);
                swap(current->visitTime, index->visitTime);
                swap(current->disabilityOption, index->disabilityOption);
                swap(current->doctorName, index->doctorName);
                swap(current->medicineInformation, index->medicineInformation);
                swapping = 1;
            }
        }
    } while (swapping);
    viewPatientsPageByPage(visitHistoryHead);
    system("cls");
}
```

The sortVisitHistory() function will display all the patient records in the visit history list, page-by-page, in descending order of visit datetime. Basically, the most recently visited patient record will be displayed first, while the oldest patient record will be displayed last. For the sorting algorithm, we choose bubble sorting. First, the system checks whether the visit history list is empty. If true, nothing will be displayed. Else, the system starts sorting the linked list with bubble sort. The system compares the first node with the next, then the next with its next, and so on. The loop breaks whenever in one of its iteration, no swapping happened.

To compare whether one date is older than the other, the algorithm first compares the year. The date with the bigger year is the more recent one. If both years are equal, the algorithm compares the month. The month with the bigger month is the more recent one. The process

repeats for day and time. “substr()” function is used to extract day, month and year into integer array elements, then individually compared.

```
Patient Details as below:

================================================================================================================
Patient ID: P3

================================================================================================================
First Name: Suresh  Last Name: Sarda
Age: 62
Gender: M
Phone Number: 0128549678
Address: No3, Jalan 3, Taman GHI
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 15.19
Disability: N
Doctor's Name: Doctor Tay
Medicine Information: Antibiotics

================================================================================================================
1. Next Patient  2.Previous Patient      0.Exit to Menu Page
================================================================================================================
Select an option:1
```

```
Patient Details as below:

================================================================================================================
Patient ID: P2

================================================================================================================
First Name: Ahmad  Last Name: Faliq
Age: 2
Gender: M
Phone Number: 0167844877
Address: No2, Jalan 2, Taman DEF
Sickness Description: None
Visit Date: 12/09/2021  Visit Time: 09.14
Disability: N
Doctor's Name: Doctor Mutu
Medicine Information: Antibiotics

================================================================================================================
1. Next Patient  2.Previous Patient      0.Exit to Menu Page
================================================================================================================
Select an option:1
```

```
Patient Details as below:

================================================================================================================
Patient ID: P1

================================================================================================================
First Name: Teo  Last Name: Zicheng
Age: 20
Gender: M
Phone Number: 1234567890
Address: abcdef
Sickness Description: Fever
Visit Date: 01/01/2020  Visit Time: 13.45
Disability: N
Doctor's Name: Lee
Medicine Information: Painkiller

================================================================================================================
1. Next Patient  2.Previous Patient      0.Exit to Menu Page
================================================================================================================
Select an option:1
There is no more patient!
Press any key to continue . . .
```

Notice how the records are arranged in descending order in terms of visit date and time.

Search Patients from Visit History List

```
void searchVisitHistory() {
    bool r = true;
    char input;
    while (r == true) {
        bool found = false;
        bool contain = false;
        string keyWord;
        int option;
        bool valid = false;
        do {
            cout << "Search by: (0 for Sickness Description, 1 for First Name)";
            cin >> option;
            if (cin.fail()) {
                cin.clear();
                cin.ignore();
                cout << "Invalid input. Enter only 0 or 1." << endl << endl;
                continue;
            }
            else {
                valid = true;
            }
            switch (option) {
            case 0:
                cout << "Enter Sickness Description: ";
                cin.ignore();
                getline(cin, keyWord);
                break;
            case 1:
                cout << "Enter first name: ";
                cin.ignore();
                getline(cin, keyWord);
                break;
            default:
                cout << "Invalid input. Enter only 0 or 1." << endl << endl;
                valid = false;
            }
        } while (!valid || (option != 0 && option != 1));

        // Start searching for matching keyword
        if (visitHistoryHead == NULL) {
            cout << "The patient visit history is empty!";
            return;
        }

        current = visitHistoryHead;

        while (current != NULL) {
            switch (option) {
            case 0:
                if (current->sicknessDescription.find(keyWord) != string::npos) {
                    found = true;
                    contain = true;
                }
                break;
            case 1:
                if (current->firstName.find(keyWord) != string::npos) {
                    found = true;
                    contain = true;
                }
            }

            if (contain == true) {
                cout << "===============" << endl;
                cout << "Patient ID: " << current->patientID << endl;
                cout << "===============" << endl;
                cout << "First Name: " << current->firstName << " " << "Last Name: " << current->lastName << endl;
                cout << "Age: " << current->age << endl;
                cout << "Gender: " << current->gender << endl;
```

```
                cout << "Phone Number: " << current->phoneNo << endl;
                cout << "Address: " << current->address << endl;
                cout << "Sickness Description: " << current->sicknessDescription << endl;
                cout << "Visit Date: " << current->visitDate << " " << "Visit Time: " << current->visitTime << endl;
                cout << "Disability: " << current->disabilityOption << endl;
                cout << "Doctor's Name: " << current->doctorName << endl;
                cout << "Medicine Information: " << current->medicineInformation << endl << endl;
                contain = false;
            }
            current = current->nextAddress;
        }

        if (!found) {
            cout << "No results found" << endl;
        }

        do {
            cout << "Continue searching? (Y/N): " << endl;
            cin >> input;
            cin.ignore();
            if (!(input == 'Y' || input == 'y' || input == 'N' || input == 'n')) {
                cout << "Invalid input. Enter only Y/N.";
            }
        } while (!(input == 'Y' || input == 'y' || input == 'N' || input == 'n'));
        if (input == 'N' || input == 'n') {
            r = false;
        }
    }
    system("cls");
}
```

The searchVisitHistory() function lets doctor users search for patient record in the visit history list by either sickness description or first name. First, the system prompts the user to enter either 0 to search by sickness description, or 1 to search by first name. Then, the user will be prompted to enter a keyword to search for. The system will then first check whether the visit history list is empty. If true, the function returns. Else, the “current” node will be used to traverse through the visit history list to look for nodes that satisfy the condition. The system will look for the keyword in the patient record’s sickness description or first name with the find() function. If said field value has the keyword in it, the patient record will be displayed. This way, multiple patient records that meet the condition will be displayed. If no results were found, the system simply notifies the user that there is no record that match the condition.

Below shows the output window of a few search attempt.

```
Search by: (0 for Sickness Description, 1 for First Name)0
Enter Sickness Description: Fever
===============
Patient ID: P1
===============
First Name: Teo Last Name: Zicheng
Age: 20
Gender: M
Phone Number: 1234567890
Address: abcdef
Sickness Description: Fever
Visit Date: 01/01/2020 Visit Time: 13.45
Disability: N
Doctor's Name: Lee
Medicine Information: Painkiller

Continue searching? (Y/N):
Y
Search by: (0 for Sickness Description, 1 for First Name)0
Enter Sickness Description: None
===============
Patient ID: P3
===============
First Name: Suresh Last Name: Sarda
Age: 62
Gender: M
Phone Number: 0128549678
Address: No3, Jalan 3, Taman GHI
Sickness Description: None
Visit Date: 12/09/2021 Visit Time: 15.19
```

```
Disability: N
Doctor's Name: Doctor Tay
Medicine Information: Antibiotics

===============
Patient ID: P2
===============
First Name: Ahmad Last Name: Faliq
Age: 2
Gender: M
Phone Number: 0167844877
Address: No2, Jalan 2, Taman DEF
Sickness Description: None
Visit Date: 12/09/2021 Visit Time: 09.14
Disability: N
Doctor's Name: Doctor Mutu
Medicine Information: Antibiotics

Continue searching? (Y/N):
```

```
Search by: (0 for Sickness Description, 1 for First Name)1
Enter first name: T
===============
Patient ID: P1
===============
First Name: Teo Last Name: Zicheng
Age: 20
Gender: M
Phone Number: 1234567890
Address: abcdef
Sickness Description: Fever
Visit Date: 01/01/2020 Visit Time: 13.45
Disability: N
Doctor's Name: Lee
Medicine Information: Painkiller

Continue searching? (Y/N):
```

```
Search by: (0 for Sickness Description, 1 for First Name)0
Enter Sickness Description: abcdef
No results found
Continue searching? (Y/N):
```

4.0 Conclusion

As a conclusion, the creation of the patient queue-based management system has taught us the power of linked list. It showed us that there is no need to use an external source to store data. A linked list is able to store multiple data without wasting any memory, making it efficient to store meaningful information. Linked list is also easy to be implemented compared to other data structures

Patients' Queue based Management System helps Klinik Sulaiman to solve the patient's information management issue, which includes calling patients to be treat, arranging patients on the waiting list and visit history list, sorting patients according to the prioritization, and others.

Through this module, we have learnt teamwork and successfully divide the jobs equally for each member.