Open-Ended Problem (OEP)

- **Enrollment no :IU2341231571(Anurag Barkhade)**
- **Sub : Data Structure and Algorithms**
- **Sub-Code : CE0417**
- **Name of Faculty : Ms.Zalak Vyas**

Open-Ended Problem Statement: Hospital Patient Management System

Scenario: You are tasked with designing a Hospital Patient Management System to efficiently manage patient records, appointments, and medical histories for a hospital. The hospital serves a large number of patients daily and requires a robust system to handle the influx of data and provide quick access to relevant information for healthcare professionals.

Requirements:

Patient Records Management:

The system should be able to store and manage patient records efficiently. Each record should include details such as patient name, age, gender, contact information, medical history, etc.
The data structure chosen for this task should allow for quick insertion, retrieval, and modification of patient records.

Appointment Scheduling:

Healthcare professionals should be able to schedule appointments for patients based on their availability and the patient's needs.
The system should ensure that appointments do not overlap and can handle rescheduling or cancellations efficiently.
The chosen data structure should support scheduling algorithms to optimize appointment slots and minimize conflicts.

Medical History Tracking:

The system should maintain a comprehensive medical history for each patient, including past diagnoses, treatments, prescriptions, etc.
Healthcare professionals should be able to access a patient's medical history quickly during consultations or treatments.
The data structure chosen for this task should allow for efficient storage and retrieval of medical records, possibly organizing them chronologically or categorically.

Emergency Handling:

In case of emergencies, healthcare professionals should be able to quickly access critical information about patients, such as allergies, ongoing treatments, etc.The system should prioritize emergency cases and provide instant access to relevant data.The data structure selected should support fast retrieval of critical information, possibly through indexing or prioritization mechanisms.

Scalability and Performance:
The system should be designed to handle a large volume of patient data and remain responsive even during peak hours.The chosen data structure(s) should be scalable and efficient, ensuring optimal performance under varying loads.

- **Following are Solution Approaches :**

Solution Approach: To address the requirements outlined above, we can consider using various data structures such as:

Hash Tables: For fast retrieval of patient records and medical history based on unique identifiers like patient ID.

Priority Queues: For managing appointment scheduling, with priorities based on appointment times.

Linked Lists or Trees: For organizing and traversing patient records and medical histories efficiently.

Graphs: For modeling relationships between patients, healthcare professionals, and medical facilities, facilitating emergency handling and resource allocation.

- **What We Do :**

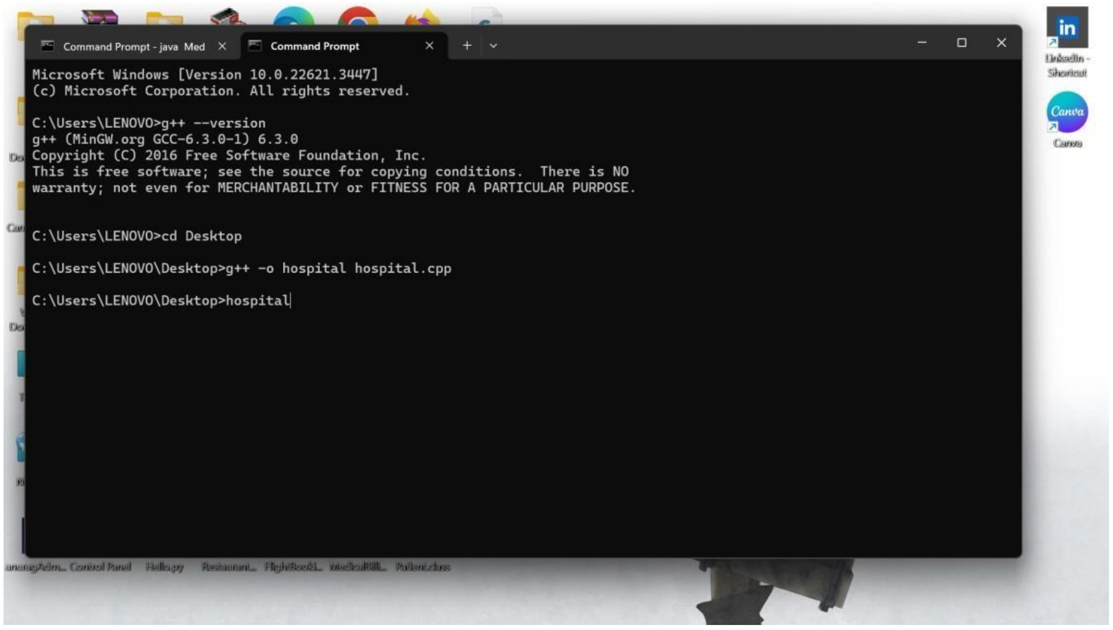
By working together, the team can design a Hospital Patient Management System that leverages appropriate data structures to effectively manage patient records, appointments, and medical histories, ultimately improving healthcare delivery and patient outcomes.

Code :

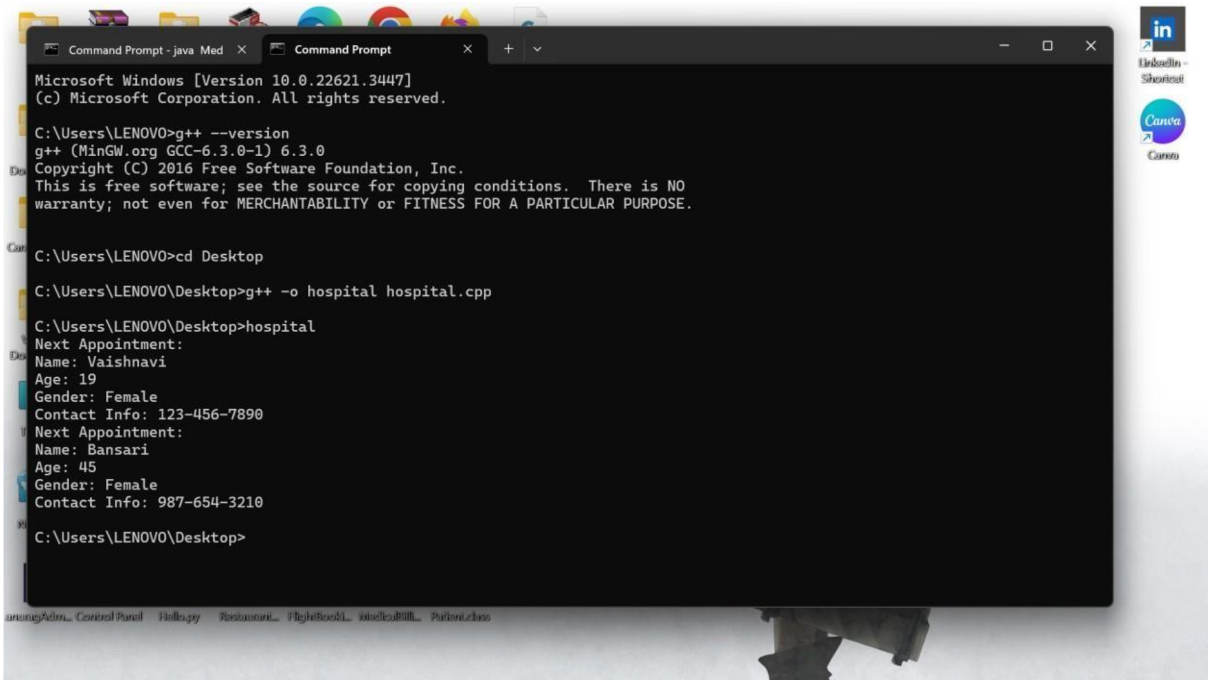
```
#include <iostream>
#include <unordered_map>
#include <string>  #include
<queue>

class Patient {
private:
std::string name;     int
age;     std::string
gender;
   std::string contactInfo;

public:
   Patient(const std::string& name, int age, const std::string& gender, const std::string&
contactInfo)
       : name(name), age(age), gender(gender), contactInfo(contactInfo) {}

   void display() const {         std::cout << "Name: " << name <<
std::endl;         std::cout << "Age: " << age << std::endl;
std::cout << "Gender: " << gender << std::endl;         std::cout
<< "Contact Info: " << contactInfo << std::endl;
   }
};

class Hospital { private:
   std::unordered_map<int, Patient> patientRecords;
std::queue<int> appointmentQueue;

public:     void addPatientRecord(int id, const Patient&
patient) {         patientRecords.emplace(id, patient);
   }

   void scheduleAppointment(int id) {
appointmentQueue.push(id);
   }

   void displayNextAppointment() {         if
(!appointmentQueue.empty()) {             int
id = appointmentQueue.front();
appointmentQueue.pop();             auto it =
patientRecords.find(id);             if (it !=
patientRecords.end()) {
             std::cout << "Next Appointment:" << std::endl;
it->second.display();
```

```
            } else {
                std::cout << "Patient ID " << id << " not found." << std::endl;
            }
        } else {
            std::cout << "No appointments scheduled." << std::endl;
        }
    }
};

int main() {

    Hospital hospital;

    Patient patient1("Vaishnavi", 19, "Female", "123-456-7890");
    Patient patient2("Bansari", 45, "Female", "987-654-3210");

    hospital.addPatientRecord(1, patient1);
hospital.addPatientRecord(2, patient2);

    hospital.scheduleAppointment(1);
hospital.scheduleAppointment(2);

    hospital.displayNextAppointment(); // Should display John Doe's appointment
hospital.displayNextAppointment(); // Should display Jane Smith's appointment

    return 0;
}
```

Output :



```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>g++ --version
g++ (MinGW.org GCC-6.3.0-1) 6.3.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


C:\Users\LENOVO>cd Desktop

C:\Users\LENOVO\Desktop>g++ -o hospital hospital.cpp

C:\Users\LENOVO\Desktop>hospital
```



```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>g++ --version
g++ (MinGW.org GCC-6.3.0-1) 6.3.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


C:\Users\LENOVO>cd Desktop

C:\Users\LENOVO\Desktop>g++ -o hospital hospital.cpp

C:\Users\LENOVO\Desktop>hospital
Next Appointment:
Name: Vaishnavi
Age: 19
Gender: Female
Contact Info: 123-456-7890
Next Appointment:
Name: Bansari
Age: 45
Gender: Female
Contact Info: 987-654-3210

C:\Users\LENOVO\Desktop>
```

Conclusion : Thus,In this way we have explained the open ended problem statement of Hospital Management System and solve it.