CS-221 Data Structures & Algorithms Lab Semester Project Report

MEMBERS:

- **2021323** (M. Abdullah Rizwan)
- **2021191** (Hammad Qaiser)
- **2021016** (Abdul Moiz Javaid)
- **2021307** (Kumail Hasan Rizvi)

Table of Contents:

Project Description:

The code provided is a C++ program that implements a hospital management system using a priority queue to manage the patient queue while storing the patient's data in a structure.

The program begins by defining a Patient struct to represent a patient and their medical information. The Patient struct includes fields for the patient's name, age, gender, and condition (severity of their medical condition). The struct overloads the < operator to allow patients to be compared based on their condition, with patients who have more severe conditions being given higher priority.

Next, the program defines a Hospital class to implement the hospital management system. The Hospital class has a private member variable patient_queue_ which is a priority queue of Patient objects. The class provides public member functions to add new patients to the queue, remove patients from the queue, and check if the queue is empty.

In the main function, the program creates an instance of the Hospital class and adds some patients to the queue using the AddPatient function. It then processes the patient queue by removing patients from the queue one at a time using the GetNextPatient function and printing their names. The program continues to process the queue until it is empty, as determined by the IsEmpty function.

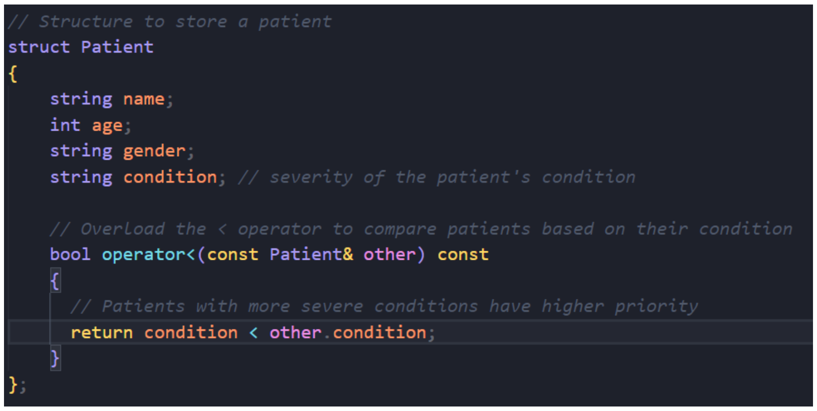
Overall, this C++ program demonstrates how to use a priority queue to implement a hospital management system that manages the order in which patients are seen by doctors based on the severity of their condition.

Data Structures used:

- Structure (used to store patients' data)
- Queue (used to prioritize patients according to their conditions)

Screenshots:

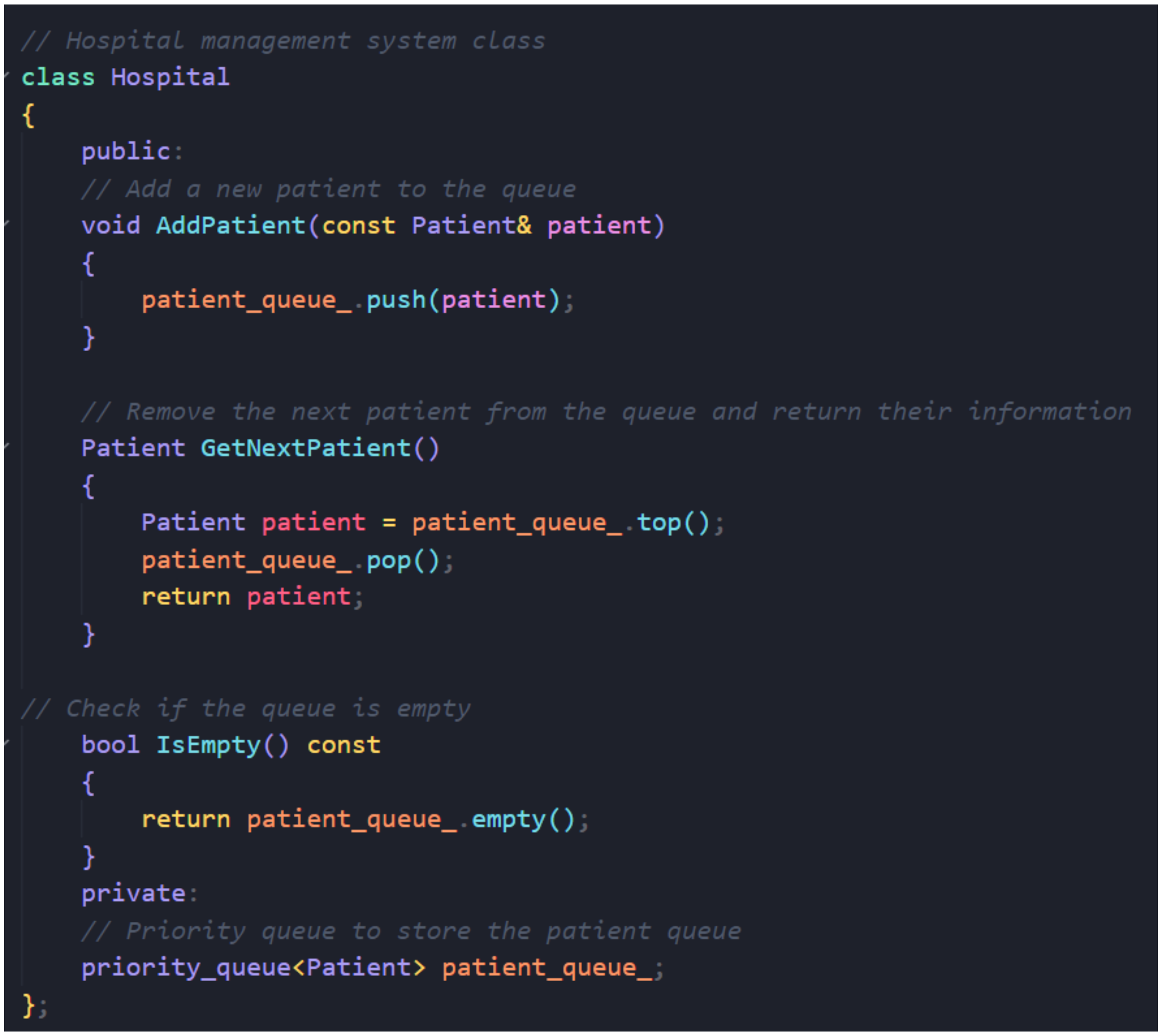
- The Patient struct that stores patient's details:



```cpp
// Structure to store a patient
struct Patient
{
    string name;
    int age;
    string gender;
    string condition; // severity of the patient's condition

    // Overload the < operator to compare patients based on their condition
    bool operator<(const Patient& other) const
    {
      // Patients with more severe conditions have higher priority
      return condition < other.condition;
    }
};
```

- The Hospital class:

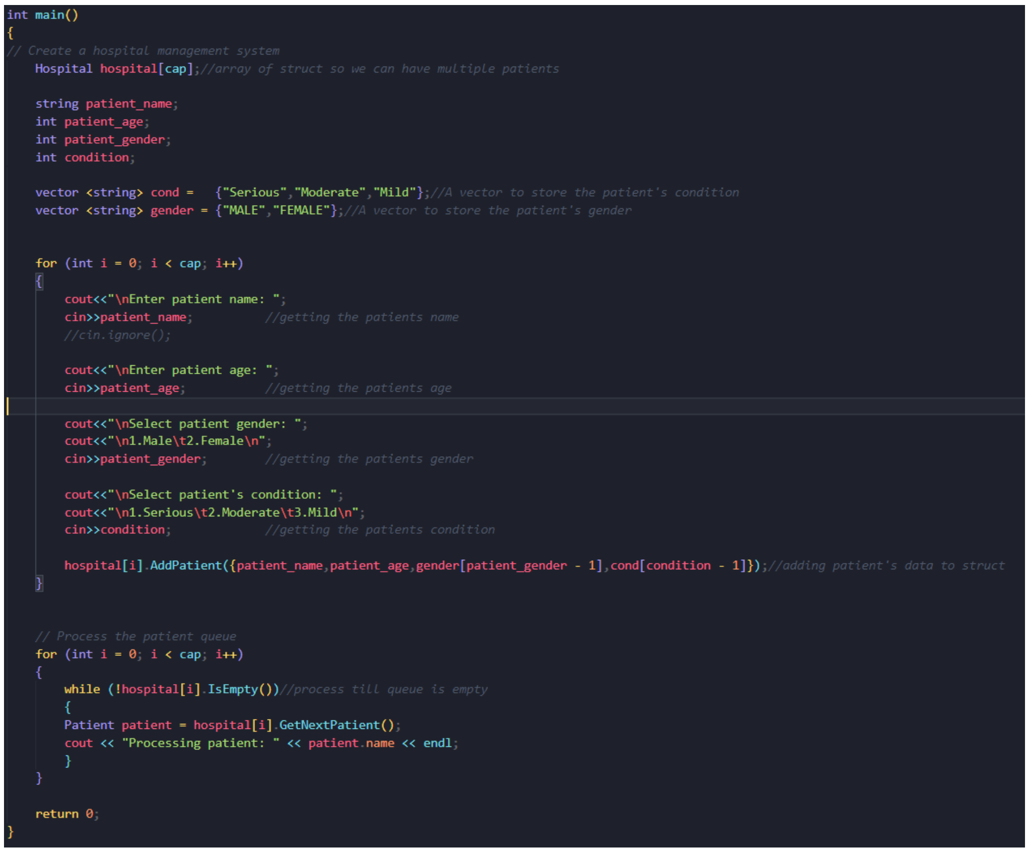


```cpp
// Hospital management system class
class Hospital
{
    public:
    // Add a new patient to the queue
    void AddPatient(const Patient& patient)
    {
        patient_queue_.push(patient);
    }

    // Remove the next patient from the queue and return their information
    Patient GetNextPatient()
    {
        Patient patient = patient_queue_.top();
        patient_queue_.pop();
        return patient;
    }

// Check if the queue is empty
    bool IsEmpty() const
    {
        return patient_queue_.empty();
    }
    private:
    // Priority queue to store the patient queue
    priority_queue<Patient> patient_queue_;
};
```

- The main function:



```cpp
int main()
{
// Create a hospital management system
    Hospital hospital[cap];//array of struct so we can have multiple patients

    string patient_name;
    int patient_age;
    int patient_gender;
    int condition;

    vector <string> cond =   {"Serious","Moderate","Mild"};//A vector to store the patient's condition
    vector <string> gender = {"MALE","FEMALE"};//A vector to store the patient's gender


    for (int i = 0; i < cap; i++)
    {
        cout<<"\nEnter patient name: ";
        cin>>patient_name;          //getting the patients name
        //cin.ignore();

        cout<<"\nEnter patient age: ";
        cin>>patient_age;           //getting the patients age

        cout<<"\nSelect patient gender: ";
        cout<<"\n1.Male\t2.Female\n";
        cin>>patient_gender;        //getting the patients gender

        cout<<"\nSelect patient's condition: ";
        cout<<"\n1.Serious\t2.Moderate\t3.Mild\n";
        cin>>condition;             //getting the patients condition

        hospital[i].AddPatient({patient_name,patient_age,gender[patient_gender - 1],cond[condition - 1]});//adding patient's data to struct
    }


    // Process the patient queue
    for (int i = 0; i < cap; i++)
    {
        while (!hospital[i].IsEmpty())//process till queue is empty
        {
        Patient patient = hospital[i].GetNextPatient();
        cout << "Processing patient: " << patient.name << endl;
        }
    }

    return 0;
}
```