CA 3: Experiential Learning

Group Members:

Sr. No.	PRN	Name of Student	Mail id
1	22070122056	Divyansh Kumar	divyansh.btech2022@sitpune.edu.in
2	22070122056	Dhruva Kashyap	dhruva.kashyap.btech2022@sitpune.edu.in

Problem Statement:

Hospital Management System

The Hospital Management System is a comprehensive solution designed to efficiently manage the operations of a healthcare facility. It encompasses the management of patients, medical professionals, and medical records. This report outlines the key components and features of the system, highlighting the effective use of multi-level inheritance and polymorphism to enhance functionality.

Explanation:

This C++ program simulates a basic hospital management system. Here's a brief explanation of the code:

1.Class `paitentRecords`: This class represents a patient's records.It contains protected member variables like `name`, `age`, `admissiondDate`, `sex`, and arrays for `Symptoms`, `vaccines`, and `notes`. It has member functions to get details, record symptoms, medical history, medicines, and add notes.

2. Class `Hospital`:This class represents the hospital itself. It contains protected member variables like `num_beds`, `num_patients`, `num_vacant`, `index`, `pointer`, and arrays for `patients`.It has functions like `admitPatient` to admit a patient, and `displayBill` to generate a bill for a patient's stay.

3. Class `Orthopedics`, `Cardiology`, `Pediatrics`:These classes inherit from `Hospital` and represent specialized departments. Each has a specific fee associated with it (`orthopedicsFee`, `cardiologyFee`, `pediatricsFee`).

4. Class `Doctors`: - This class inherits from both `paitentRecords` and `Hospital`.It represents a doctor and contains member variables `specialization` and `doctorID`.It has a function `prescribeMedication` to prescribe medication.

5. `main` Function:The `main` function is where the program starts executing. It displays a menu for the user to choose between different wards or to exit.Depending on the user's choice, it allows admitting patients or generating bills.

6. User Interaction: The program provides a menu with options for different wards (General, Pediatrics, Cardiology, Orthopedics) and an option to exit. Within each ward, the user can choose to admit a patient or generate a bill.

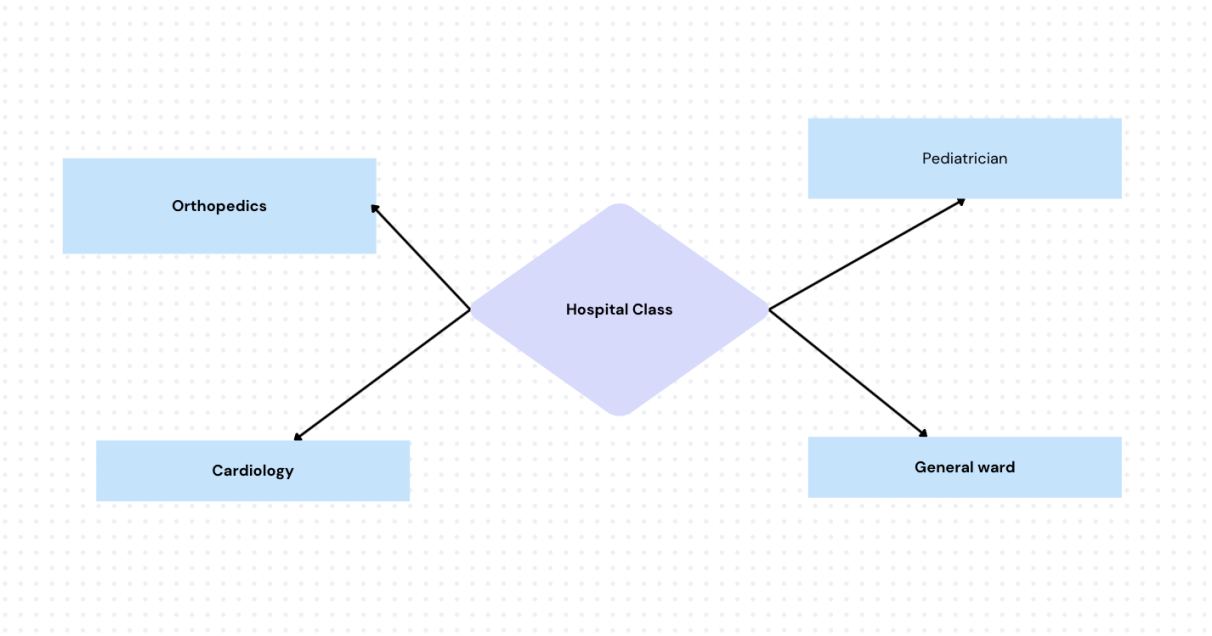
7. Looping Structure:The program uses a `do-while` loop to repeatedly display the menu and allow the user to interact until they choose to exit.

8. Input Validation:The program prompts the user for input and expects specific types of data (e.g., integers, strings).It doesn't handle cases where invalid input is provided, so it could be enhanced with error handling.

9. Output Formatting: - The program uses `setw`, `left`, `right`, and `fixed` from `<iomanip>` for formatting the output.

10. Exiting the Program - When the user chooses to exit, the program calls `exit(0)` to terminate.

Class Diagram:



Code snippets:

```
class paitentRecords
{
protected:
    string name;
    int age, admissiondDate;
    char sex;
    int x, y;

    string Symptoms[30];
    string vaccines[10];
    string notes;

public:
    paitentRecords()
    {
        name = "";
        age = 0;
        sex = '\0';
    }
    void getDetails()
    {
        cout << "Enter paitent name: ";
        cin >> name;
        cout << "Enter paintent age: ";
        cin >> age;
        cout << "Enter date of admission(in DDMMYYYY): ";
        cin >> admissiondDate;
    };
    void symptoms()
    {
        int x;
        cout << "Enter number of symptoms: ";
        cin >> x;
        for (int i = 0; i < x; i++)
        {
            cout << "Enter Symptom: ";
            cin >> Symptoms[i];
        }
    };
    void medicalHistory()
    {
        cout << "Enter vaccine history of Patient: ";
        cin >> y;
        for (int i = 0; i < y; i++)
        {
            cin >> vaccines[i];
        }
    };
    void medicines(){
```

```
    void medicines(){

    };

    friend class Hospital;
};

class Hospital
{
protected:
    int num_beds, num_patients, num_vacant;
    int index[50];
    int pointer = 0;
    double total;
    int days;
    paitentRecords patients[50];

public:
    Hospital()
    {
        num_beds = 50;
        num_patients = 0;
        num_vacant = 50;
    }

    void admitPatient()
    {
        patients[pointer].getDetails();
        index[pointer] = 1;
        cout << "Patient admitted to bed number " << pointer << endl;
        pointer++;
    }

    virtual void displayBill()
    {
        int bednum;
        cout << "Enter patient's bed num: ";
        cin >> bednum;
        cout << "Enter number of days: ";
        cin >> days;
        cout << fixed << setprecision(2);
        cout << "------------------------ Hospital Bill ------------------------" << endl;
        cout << setw(20) << left << "Patient Name: " << setw(30) << right << patients[bednum].name << endl;
        cout << setw(20) << left << "Patient ID: " << setw(30) << right << bednum << endl;
        cout << "---------------------------------------------------------------" << endl;
        cout << setw(20) << left << "Base Charger: " << setw(30) << right << "Rs" << days * 200 << endl;
        cout << "---------------------------------------------------------------" << endl;
    }
    friend class paitentRecords;
};

class Orthopedics : public Hospital
```

```
    friend class paitentRecords;
};

class Orthopedics : public Hospital
{
private:
    double orthopedicsFee;

public:
    Orthopedics()
    {
        orthopedicsFee = 450;
    }

    void displayBill() override
    {
        Hospital::displayBill();

        std::cout << std::setw(20) << std::left << "Orthopedics Fee: " << std::setw(30) << std::right << "Rs" << orthopedicsFee * days << std::endl;
        std::cout << std::setw(20) << std::left << "Total Fees: " << std::setw(30) << std::right << "Rs" << orthopedicsFee*days + days * 200 << std::endl;
        std::cout << "-----------------------------------------------------------------" << std::endl;
    }
};

class Cardiology : public Hospital
{
private:
    double cardiologyFee;

public:
    Cardiology()
    {
        cardiologyFee = 6000;
    }

    void displayBill() override
    {
        Hospital::displayBill();

        std::cout << std::setw(20) << std::left << "Cardiology Fee: " << std::setw(30) << std::right << "Rs" << cardiologyFee*days << std::endl;
        std::cout << std::setw(20) << std::left << "Total Fees: " << std::setw(30) << std::right << "Rs" << cardiologyFee * days + days * 200 << std::endl;
        std::cout << "-----------------------------------------------------------------" << std::endl;
    }
};

class Pediatrics : public Hospital
{
private:
    double pediatricsFee;

public:
    Pediatrics()
```

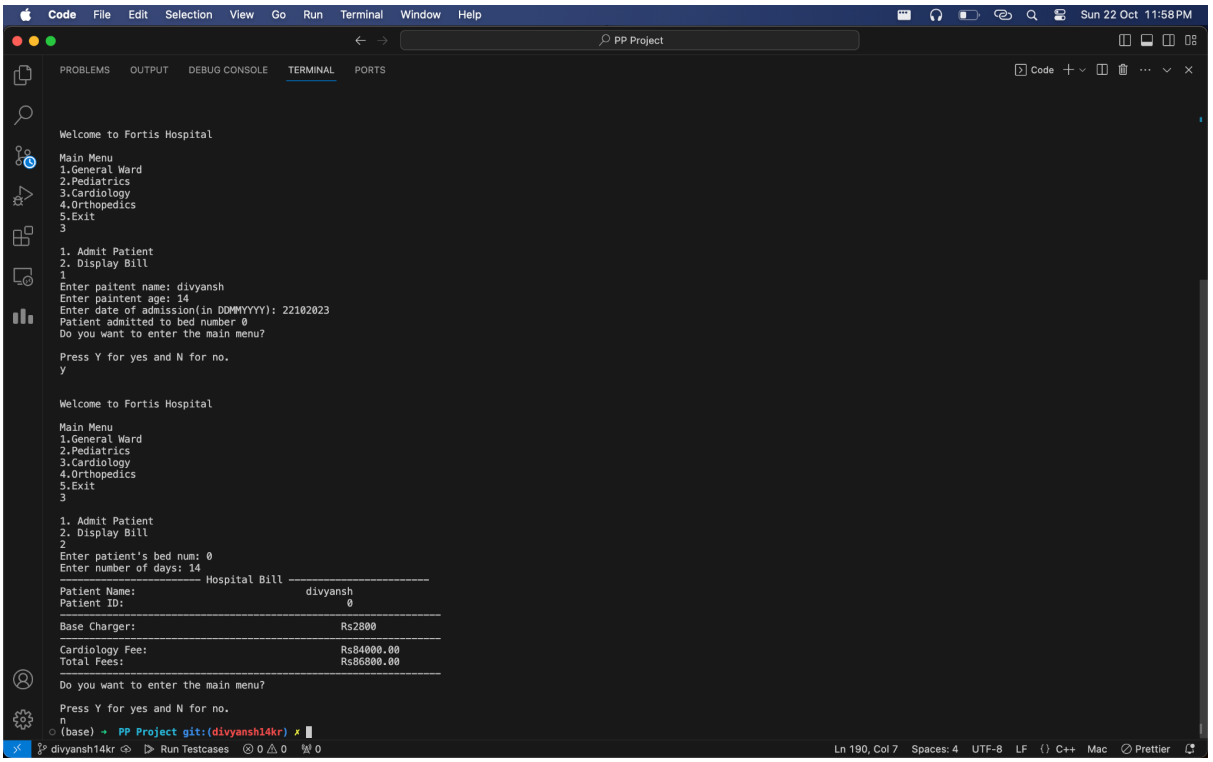
```
            case 2:
                cout << "\n1. Admit Patient\n";
                cout << "2. Display Bill\n";
                cin >> choice2;
                switch (choice2)
                {
                case 1:
                    pedo.admitPatient();
                    break;
                case 2:
                    pedo.displayBill();
                    break;
                }
                break;
            case 3:
                cout << "\n1. Admit Patient\n";
                cout << "2. Display Bill\n";
                cin >> choice2;
                switch (choice2)
                {
                case 1:
                    cardiologyPatient.admitPatient();
                    break;
                case 2:
                    cardiologyPatient.displayBill();
                    break;
                }
                break;
            case 4:
                cout << "\n1. Admit Patient\n";
                cout << "2. Display Bill\n";
                cin >> choice2;
                switch (choice2)
                {
                case 1:
                    orthopedicsPatient.admitPatient();
                    break;
                case 2:
                    orthopedicsPatient.displayBill();
```

```
        std::cout << "-------------------------------------------------------------------------" << std::endl;
    }
};
int main()
{
    int choice, choice2;
    char input = 'Y';
    Cardiology cardiologyPatient;
    Orthopedics orthopedicsPatient;
    Hospital paitent;
    Pediatrics pedo;
    do
    {
        cout << "\n\nWelcome to Fortis Hospital\n";
        cout << "\nMain Menu\n";
        cout << "1.General Ward\n";
        cout << "2.Pediatrics\n";
        cout << "3.Cardiology\n";
        cout << "4.Orthopedics\n";
        cout << "5.Exit\n";

        cin >> choice;

        switch (choice)
        {
        case 1:
            cout << "\n1. Admit Patient\n";
            cout << "2. Display Bill\n";
            cin >> choice2;
            switch (choice2)
            {
            case 1:
                paitent.admitPatient();
                break;
            case 2:
                paitent.displayBill();
                break;
            }
            break;
```

Input/Output:



Github repository link:

https://github.com/divyansh14kr/hospitalManagementSystem