Program Development Process

Step 1: Requirements specification

Read the problem to be solved.
State the problem clearly / Understand the problem.
Describe the problem to be solved in your own words here:

The purpose of the program is to create a program that allows to input a patient's name with the doctor's name and data to get the total amount of money owed.

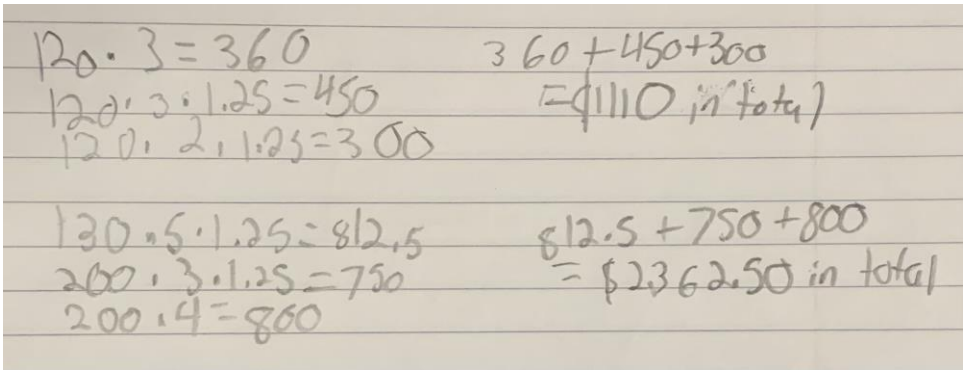
Step 2: Analyze

Describe the data flow and to identify the inputs, outputs, and constants.
Identify what the output is first, and then figure out what input data you need.
This list will eventually lead to the list of variables and constants to be defined.

Inputs	Outputs	Constants
First Name	Amount of money owed	None
Last Name	Full Name	
Pay Rate	Total Pay	
Primary Physician		
Amount of Pay		

Step 3: Program design

Describe the process for obtaining the output from the input.
Note the steps you are performing.
This will lead to the C++ statements you write.
Your algorithm design is described here.
Show your manual calculations (hand work) here.
To create this program, I would have to first make a base class called Person and create two derived classes called Doctor and Patient. I would then use the classes to store the data needed to calculate the amount of money. The total would be the amount due multiplied by the doctor's hourly rate. If the doctor is not the patient's primary physician, then there would be an additional 25% increase in the price.



Step 4: Implementation in C++

Also known as coding
Develop a C++ solution using your work from step 3.
Write the declarations first and then write the C++ statements.
Enter and debug program on the computer.
Show your source code here:

```
#include <iostream>
#include <iomanip>
using namespace std;

class Person //Base Class
{
protected:
string firstname; //First Name
string lastname; //Last Name
public:
//Constructors
Person();
Person(string first, string last);

//Mutators
void setFirst(string first);
void setLast(string last);

//Accesors
string getFirst();
string getLast();

//Overloading Operators
bool operator == (Person& p);
Person& operator = (Person& p);
friend ostream& operator <<(ostream& output, Person& p);
friend istream& operator >>(istream& input, Person& p);
};

class Doctor : public Person //Derived Class
{
private:
double rate; //Pay Rate
public:
//Constructors
Doctor();
Doctor(double r, Person p);

//Mutator
void setRate(double r);

//Accesor
double getRate();

//Overloading Operator
Person& operator = (Doctor& p);
};

class Patient : public Person //Derived Class
{
private:
Doctor prime; //Primary Physician
public:
//Constructors
Patient();
Patient(Doctor d, Person p);

//Mutator
void setPrime(Doctor p);

//Accesor
Doctor getPrime();
};

class Billing
{
private:
Doctor doctor; //Doctor
```

```
Patient patient; //Patient
double amount; //Amount of pay
public:
//Constructors
Billing();
Billing(Doctor d, Patient p, int hours);

//Mutator
void setAmount(double a);

//Accesor
double getAmount();
};

int main()
{
Person patient, physician, doctor;
double rate;
double total = 0;
int hours;

//Patient 1
cout << "Enter the patient's name: ";
cin >> patient;
cout << "Enter primary physician's name and their rate: ";
cin >> physician >> rate;
Patient person1(Doctor(rate, physician), patient); //Stores patient data
cout << "Enter a doctor's name and their hourly rate: ";
cin >> doctor >> rate;
cout << "Enter amount of hours: ";
cin >> hours;
Billing bill1(Doctor(rate, doctor), person1, hours); //Stores billing data
total = total + bill1.getAmount(); //Calcualtes total

//Patient 2
cout << "\nEnter the patient's name: ";
cin >> patient;
cout << "Enter primary physician's name and their rate: ";
cin >> physician >> rate;
Patient person2(Doctor(rate, physician), patient); //Stores patient data
cout << "Enter a doctor's name and their hourly rate: ";
cin >> doctor >> rate;
cout << "Enter amount of hours: ";
cin >> hours;
Billing bill2(Doctor(rate, doctor), person2, hours); //Stores billing data
total = total + bill2.getAmount(); //Calcualtes total

//Patient 3
cout << "\nEnter the patient's name: ";
cin >> patient;
cout << "Enter primary physician's name and their rate: ";
cin >> physician >> rate;
Patient person3(Doctor(rate, physician), patient); //Stores patient data
cout << "Enter a doctor's name and their hourly rate: ";
cin >> doctor >> rate;
cout << "Enter amount of hours: ";
cin >> hours;
Billing bill3(Doctor(rate, doctor), person3, hours); //Stores billing data
total = total + bill3.getAmount(); //Calcualtes total

//Totals
cout << "\n" << person1 << " owes: " << bill1.getAmount() << " dollars" << endl;
cout << person2 << " owes: " << bill2.getAmount() << " dollars" << endl;
cout << person3 << " owes: " << bill3.getAmount() << " dollars" << endl;

cout << "\nThe total income from the billing records: " << fixed << setprecision(2) << total << " dollars";
}

//Person Class
Person::Person() //Default Constructor
{
firstname = "";
lastname = "";
}

Person::Person(string first, string last) //Constructor that takes 2 strings
```

```
{
firstname = first;
lastname = last;
}

void Person::setFirst(string first) //Sets first name
{
firstname = first;
}

void Person::setLast(string last) //Sets last name
{
lastname = last;
}

string Person::getFirst() //Returns first name
{
return firstname;
}

string Person::getLast() //Returns last name
{
return lastname;
}

bool Person::operator==(Person& p) //Overloads == operator
{
if (firstname == p.firstname && lastname == p.lastname)
return true;
return false;
}

Person& Person::operator=(Person& p) //Overloads = operator
{
firstname = p.firstname;
lastname = p.lastname;
return *this;
}

ostream& operator<<(ostream& output, Person& p) //Overloads << operator
{
output << p.firstname << " " << p.lastname;
return output;
}

istream& operator>>(istream& input, Person& p) //Overloads >> operator
{
input >> p.firstname >> p.lastname;
return input;
}

//Doctor Class
Doctor::Doctor():Person() //Default Constuctor
{
rate = 0;
}

Doctor::Doctor(double r, Person p) //Takes pay rate and person
{
firstname = p.getFirst();
lastname = p.getLast();
rate = r;
}

void Doctor::setRate(double r) //Sets pay rate
{
rate = r;
}

double Doctor::getRate() //returns pay rate
{
return rate;
}

Person& Doctor::operator=(Doctor& p) //Overloads = operator
{
```

```
firstname = p.firstname;
lastname = p.lastname;
rate = p.rate;
return *this;
}

//Patient Class
Patient::Patient():Person() //Default Constuctor
{
Doctor();
}

Patient::Patient(Doctor d, Person p) //Takes in doctor and person object
{
firstname = p.getFirst();
lastname = p.getLast();
prime = d;
}

void Patient::setPrime(Doctor p) //Sets primary physician
{
prime = p;
}

Doctor Patient::getPrime() //Returns primary physician
{
return prime;
}

//Billing Class
Billing::Billing() //Default Constructor
{
Doctor();
Patient();
amount = 0.0;
}

Billing::Billing(Doctor d, Patient p, int hours) //Takes doctor, patient, and the amount of hours
{
doctor = d;
patient = p;
if (p.getPrime().getFirst() == d.getFirst() && p.getPrime().getLast() == d.getLast()) //if the doctor involved in the bill is the patient's primary physician
amount = hours * d.getRate();
else //if the doctor involved is not the patient's primary physician
amount = hours * d.getRate() * 1.25;
}

void Billing::setAmount(double a) //Sets amount of pay
{
amount = a;
}

double Billing::getAmount() //Returns amount of pay
{
return amount;
}
```

Step 5: Testing

Test your program with sample data set to make sure the output is correct.
Should test multiple data sets including the boundary cases.
Summary and analyze your result.
Show the output screen shots here.

Microsoft Visual Studio Debug Console

```
Enter the patient's name: Michael Johnson
Enter primary physician's name and their rate: Jessica White 120.0
Enter a doctor's name and their hourly rate: Jessica White 120.0
Enter amount of hours: 3

Enter the patient's name: John Smith
Enter primary physician's name and their rate: Jessica White 120.0
Enter a doctor's name and their hourly rate: Eric Wagner 120.0
Enter amount of hours: 3

Enter the patient's name: Megan Miller
Enter primary physician's name and their rate: Peter Kipp 150.0
Enter a doctor's name and their hourly rate: Sean Anderson 120.0
Enter amount of hours: 2

Michael Johnson owes: 360 dollars
John Smith owes: 450 dollars
Megan Miller owes: 300 dollars

The total income from the billing records: 1110.00 dollars
C:\CIS\CIS212\ASN4\x64\Debug\ASN4.exe (process 2812) exited with code 0.
Press any key to close this window . . .
```

Microsoft Visual Studio Debug Console

```
Enter the patient's name: Aidan Sullivan
Enter primary physician's name and their rate: Walter White 200
Enter a doctor's name and their hourly rate: John Doe 130
Enter amount of hours: 5

Enter the patient's name: Bob Dylan
Enter primary physician's name and their rate: John Doe 130
Enter a doctor's name and their hourly rate: Walter White 200
Enter amount of hours: 3

Enter the patient's name: Andrew Dolan
Enter primary physician's name and their rate: Walter White 200
Enter a doctor's name and their hourly rate: Walter White 200
Enter amount of hours: 4

Aidan Sullivan owes: 812.5 dollars
Bob Dylan owes: 750 dollars
Andrew Dolan owes: 800 dollars

The total income from the billing records: 2362.50 dollars
C:\CIS\CIS212\ASN4\x64\Debug\ASN4.exe (process 3184) exited with code 0.
Press any key to close this window . . .
```