Question 1

a)

Class can create a subclass that will inherit parent's properties and methods but structs doesn't support inheritance.

Ex : 'Person' class can create sub classes such as 'Student' and 'Lecturer'.

In a class all members private by default but in a struct members a public.

Ex : Other objects don't have direct access to the private data in a class.

Classes are reference types and structs are value types.

Class can contain constructors and destructors, but structure only can contain default constructor.

Structure only can handle data while class can handle data and functions.

Ex : 'Student' class have student's attributes such as name, age, mark, faculty and methods such as calculateTotal(), calculateAverage(). But 'Student' structure only can hold attributes of the student.

b)

1.First identify the data in the objects as well as the process or actions that can be performed on that data.
2.Encapsulate their data and the processes that act on those data.

3.Using information hiding method hide private data of an object from other objects.

4.Determine the relationship between objects.

5.Design the algorithms for methods, using structured design.

6.Delevop the program from the algorithm.

c)

i)

Class	**Objects**	**Attributes**
Staff	Dr.Ajith Pieris, Dr.Amal Perera	staffId, name, position, faculty, joined date, idExpireDate
Student	Kamal	name, degree, faculty
Visitor	Mr.Perera	Id, name, visitingDate

ii)

Abstraction – Abstraction is the process that programmer hides all unnecessary data and keep only relevant data about an object in order to reduce complexity and increase efficiency.

Ex : On ‘Student’ class only necessary attributes for program are taken for the class.

Encapsulation - An act of combining properties and methods, related to the same object, is known as Encapsulation.

Ex : Identifying all attributes related to staff and put in to “Staff” class can call as encapsulation.

Information Hiding – Information hiding is hide internal object details such as data members to ensure privacy of data and preventing unintended or intended changes.

Ex : On “Visitor” class, divide attributes as private and restrict direct access of those data from other objects.

d)

i)

```
class Item{
        private:
                int itemID;
                char name[];
                double price;
        public:
                Item();
                Item( int pid, char pname[] );
                void setPrice( double pprice );
                double getPrice();
                void display();
};
```

ii)

```
Item( int pid, char pname[] ){
	itemID = pid;
	name[] = pname[];
}
```

iii)

```
void Item :: setPrice( double pprice ){
	price = pprice;
}
double Item :: getPrice(){
	return price;
}
```

e)

i)

```
void A :: add( B b ){
	num1 = num1 + b;
}
```

ii)

```
Int main(){
	B b1;
	A a1;
	a1.add(b1);
	a1.display();

	return 0;
}
```

iii)

```
Int main(){
	B b1(20);
	A  *a2(10);

	a1.add(b1);
	a1.display();

	return 0;
}
```

f)

i)

Aggregation

ii)

```
class Order{
private:
	date date;
	string status;
	double price;
	OrderDetails *od;
public:
	void calcSubTotal();
	float calcTax();
	double calcTotal();
}
```

```
Class OrderDetail{

Private:

        Int quantity;

        String taxStatus;

Public:

        Void calcSubTotal();

        float calcTax();

}
```

Question 2

a)

Program	Class
Rental	Redundant(Rent)
Landlord	Class
Tenant	Class
Apartment	Outside the scope
Building	Outside the scope
Rent	Class
Expense	Class
Electricity	Redundant(Expense)
Water	Redundant(Expense)
Rent record	Redundant(Rent)
Expense record	Redundant(Expense)
Date	An attribute
Payee	An attribute
Amount	An attribute
Budget category	An attribute
Annual summary	Redundant(Record)
Basic information	An attribute
Name	An attribute
Nic no	An attribute
Contact details	An attribute
Room number	An attribute
Rental contract	An attribute
Joined date	An attribute
Period of stay	An attribute

Details	An attribute
User	Redundant(Tenant)

b)

Program	
Responsibilities	Collaborations
Display annual summary	Rent, Expense

Landlord	
Responsibilities	Collaborations
Input the rents	Rent
Input expenses	Expense
Store tenants’ details	Tenant

Tenant	
Responsibilities	Collaborations
Pay rent	Rent
Search the details	

Rent	
Responsibilities	Collaborations
Display rent record	Tenant

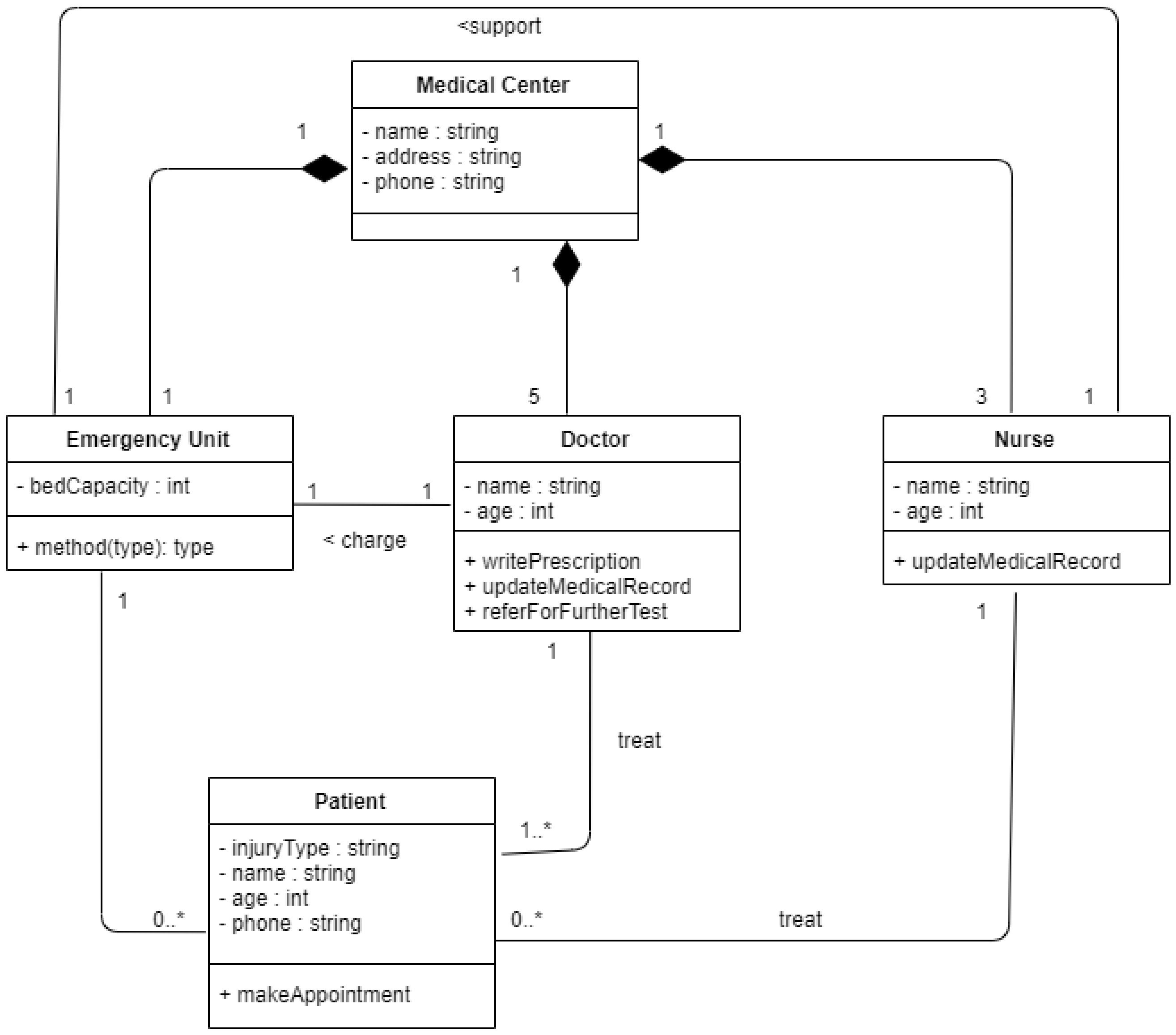
Expense	
Responsibilities	Collaborations
Display expense record	

Question 3

Five doctors with three trainee nurses run the "Health First" Medical Center. When a patient calls for an appointment, he or she usually sees the same doctor, but at busy times, patients may see any of the doctors or nurses. Once a patient has been seen by the doctor or nurse, the medical records are updated, and the doctor may also write or a prescription for the patient. Sometimes the doctor considers that the patient needs further tests. These may be routine or intensive. They are carried out at one of the local hospitals.

The Medical also has an emergency unit with four beds that can be used by patients with emergency cases or minor injuries. One doctor is in charge of the emergency' unit with the support of a nurse. A patient can be referred to the local hospital in case of further treatment or for consultation of a surgeon.

Doctor, Nurse, Patient, Medical Center, Emergency Unit



Question 4

```
Class Hospital {
        Private:
                String name;
                String address;
                String phone;
                Ward *ward[SIZE];
                Team *team[SIZE];
}

Class Team {
        Public:
                String name;
                Doctor *doc[SIZE];
                Patient *patient;
                ConsultantDoctor *cDoctor;
}


Class Ward {
        Private:
                String name;
                String gender;
                Int capacity;
                Patient *pat;
}
```

```
Class Doctor {
        Protected:
                String speciality[];
                String location[];
        Private:
                Patient *ppatient;
}

Class Patient {
        Private:
                String id;
                String gender;
                Int age;
                Date accepted;
                String sickness[];
                String prescription[];
                String allergies[];
                String specialReq[];
                Doctor *doc;
                Ward *ward;
                Team *team;
                ConsultantDoctor *conDoc;
}

Class ConsultantDoctor : public Doctor{
        Private:
                Double fee;
                Patient *ppp;
                Team *tee;
}
```

```
Class JuniorDoctor : public Doctor {
        Private:
                Int hours;
}
```