

|  |
| --- |
|  |
| Competency Assessment  Excel Technologies Ltd |
| |  |  |  | | --- | --- | --- | | Candidate: Abdullah | Cell: 01917584343 | Position: Software Engineer | |

**Answer to Question No: 1(a)**

**Question:**

1. Consider the following register. Holy Family Red Cross Hospital is using this register to manage doctors’ lists, their contact numbers, and the departments to which the doctors belong. With this register, the hospital is also managing doctor’s service points within the hospital.

1. Apply the normalization rule to normalize this register up to the 3rd normal form.

|  |  |  |  |
| --- | --- | --- | --- |
| Doctor | Contact Number | Service Points | Department |
| Dr. Lissa Mwenda | +260766219936 | Antenatal Care,  Family Planning,  Postnatal Care | Gynecology |
| Dr. Yvonne Sishuwa | +260766219937 | Family Planning,  Postnatal Care | Pediatrics |
| Dr. Machalo Mbale | +260766219938 | Antenatal Care | Radiology and Imaging |

**Answer:**

Normalization is the process of organizing data in a relational database to eliminate redundancy and dependency. Here is the summary of the normalization step:

* In 1NF, columns will have only atomic values.
* in 2NF, there will be no partial dependencies.
* in 3NF, there will be no transitive dependencies.

*So, I’m going to break this table up into 3rd normal form. For a better explanation, I’ll break down the whole process as small as possible which is not required and recommended for larger database normalization scenarios.*

1. **First Normal Form (1NF):** Let’s ensure that each column contains only indivisible values. So, there is no repeating group of columns.

|  |  |  |  |
| --- | --- | --- | --- |
| Doctor | Contact Number | Service Points | Department |
| Dr. Lissa Mwenda | +260766219936 | Antenatal Care | Gynecology |
| Dr. Lissa Mwenda | +260766219936 | Family Planning | Gynecology |
| Dr. Lissa Mwenda | +260766219936 | Postnatal Care | Gynecology |
| Dr. Yvonne Sishuwa | +260766219937 | Family Planning | Pediatrics |
| Dr. Yvonne Sishuwa | +260766219937 | Postnatal Care | Pediatrics |
| Dr. Machalo Mbale | +260766219938 | Antenatal Care | Radiology and Imaging |

Ok, the 1NF target is achieved. But now, I’m seeing repetitive data. Let’s simply break the table.

|  |  |  |
| --- | --- | --- |
| Doctor ID (PK) | Doctor | Contact Number |
| 1 | Dr. Lissa Mwenda | +260766219936 |
| 2 | Dr. Yvonne Sishuwa | +260766219937 |
| 3 | Dr. Machalo Mbale | +260766219938 |

|  |
| --- |
| **Service Point** |
| Antenatal Care |
| Family Planning |
| Postnatal Care |

|  |
| --- |
| **Department** |
| Gynecology |
| Pediatrics |
| Radiology and Imaging |

1. **Second Normal Form (2NF):** I have to eliminate all partial dependencies. That means non-key attributes are fully functionally dependent on the primary key. So, let’s create a separate table.

**Department Table:**

|  |  |
| --- | --- |
| **Department ID (PK)** | **Department Name** |
| 1 | Gynecology |
| 2 | Pediatrics |
| 3 | Radiology and Imaging |

**ServicePoint Table:**

|  |  |
| --- | --- |
| **Service Point ID (PK)** | **Service Point Name** |
| 1 | Antenatal Care |
| 2 | Family Planning |
| 3 | Postnatal Care |

1. **Third Normal Form (3NF):** Let’s remove transitive dependencies. So, I’m ensuring that no column is dependent on another non-key attribute. Currently, our dependency looks like:

* Doctor -> Department
* Doctor -> Service Points
* Department -> Service Points (Transitive Dependency)

**Updated Doctor Table: One-To-One relationship with Department Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **DoctorID (PK)** | **Doctor** | **Department ID (FK)** | **Department Name** |
| 1 | Dr. Lissa Mwenda | 1 | Gynecology |
| 2 | Dr. Yvonne Sishuwa | 2 | Pediatrics |
| 3 | Dr. Machalo Mbale | 3 | Radiology & Imaging |

**DoctorServicePoint (Intermediate Table) \_\_ Many-To-Many relationship**

|  |  |  |  |
| --- | --- | --- | --- |
| **DoctorID (PK)** | **Doctor** | **ServicePointID (PK)** | **Service Points** |
| 1 | Dr. Lissa Mwenda | 1 | Antenatal Care |
| 1 | Dr. Lissa Mwenda | 2 | Family Planning |
| 1 | Dr. Lissa Mwenda | 3 | Postnatal Care |
| 2 | Dr. Yvonne Sishuwa | 2 | Family Planning |
| 2 | Dr. Yvonne Sishuwa | 3 | Postnatal Care |
| 3 | Dr. Machalo Mbale | 1 | Antenatal Care |

**Now 3rd Normalization is Complete!**

By following this structure SQL is returning data exactly like 1NF. Here is the Join Query Code:

SELECT

d.DoctorID,

d.Name AS DoctorName,

d.ContactNumber,

dp.Name AS Department,

sp.Name AS ServicePoint

FROM

Doctor d

JOIN

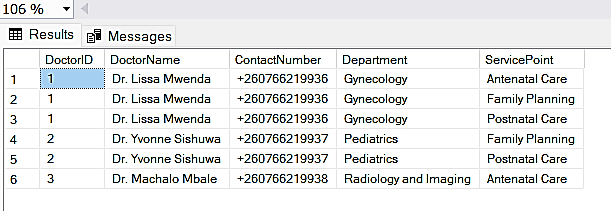
DoctorServicePoint dsp ON d.DoctorID = dsp.DoctorID

JOIN

ServicePoints sp ON dsp.ServicePointID = sp.ServicePointID

JOIN

Department dp ON d.DepartmentID = dp.DepartmentID;



Please Find the SQL script file in the Folder.

**Answer to Question No: 1(b)**

1. After normalization, draw Entity Relationship Diagram and show the degree of cardinality among entities using crow’s foot notation.

**Answer:**

Doctor

Doctor

DoctorID

Name

ContactNumber

DepartmentID

**Department**

 DepartmentID

 Name

**DoctorServicePoint**

 DoctorID

 ServicePointID



**ServicePoints**

 ServicePointID

Name

**Figure:** Entity Relationship Diagram using crow’s foot notation

**Answer to Question No: 2**

1. Consider the following loop. Trace the value of “n” in every iteration of the loop.

*int n = 30;*

*for (int i = 0; i <= 5; i++)*

*{*

*n += i;*

*}*

*print(n);*

**Answer:**

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration No | Value of ‘I’ | Increment | Value of ‘n’ |
| 1 | 0 | 30+0 | 30 |
| 2 | 1 | 30+1 | 31 |
| 3 | 2 | 31+2 | 33 |
| 4 | 3 | 33+3 | 36 |
| 5 | 4 | 36+4 | 40 |
| 6 | 5 | 40+5 | 45 |

**Answer to Question No: 4**

1. **Explain method overloading and method overriding with example. Write your code in C# programming language.**

**Answer:**

* Both method overloading and method overriding are forms of polymorphism in C#, providing flexibility in handling different method implementations based on the context of usage.
* **Method Overloading:**
* Allows a class to have multiple methods with the same name but different parameter lists.
* The compiler distinguishes between overloaded methods based on the number or types of parameters during compile time.

Here is the Example using C#:

public class MethodOverloading

{

    public int Add(int a, int b)

    {

        return a \* b;

    }

    public int Add(int a, int b, int c)

    {

        return a + b + c;

    }

    public double Add(double a, double b, int c)

    {

        return a + b + c;

    }

    public double Add(double a, int c, double b)

    {

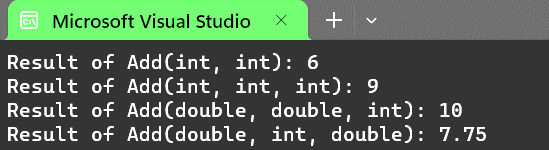
        return a \* b + c;

    }

}

**Number Of Parameters are different**

**Let’s Run this Program!**



using Answer\_4;

class Program

{

    static void Main()

    {

        MethodOverloading calculator = new MethodOverloading();

        int result1 = calculator.Add(2, 3);

        Console.WriteLine("Result of Add(int, int): " + result1);

        int result2 = calculator.Add(2, 3, 4);

        Console.WriteLine("Result of Add(int, int, int): " + result2);

        double result3 = calculator.Add(2.5, 3.5, 4);

        Console.WriteLine("Result of Add(double, double, int): " + result3);

        double result4 = calculator.Add(2.5, 4, 1.5);

        Console.WriteLine("Result of Add(double, int, double): " + result4);

    }

}

**Order Of Parameters are different**

**Type Of Parameters are different**

* **Method overriding:**
* Method overriding is a form of type polymorphism, allowing multiple methods with the same name and signature in different classes.
* It relies on Inheritance.
* The compiler resolves overridden methods based on the actual type of the object during run time, considering the method in the derived class instead of the base class.

**Here is the Example using C#:**

using System;

namespace Answer\_4

{

    public class BaseClass

    {

        public virtual void Greetings()

        {

            Console.WriteLine

("Muslim: As-Salamu-Alaikum");

        }

    }

    public class SubClass : BaseClass

    {

        public override void Greetings()

        {

            Console.WriteLine("Hindu: Namaskar");

        }

    }

}

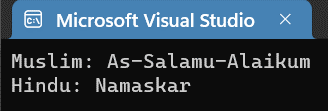
Base/Parent class use virtual keyword

Method overriding Cannot be achieved without inheritance

Derived/Child class use override keyword

**Let’s Run This Program!**

**Output:**



using Answer\_4;

class Program

{

    static void Main()

    {

        BaseClass baseObj = new BaseClass();

        SubClass subObj = new SubClass();

        baseObj.Greetings();

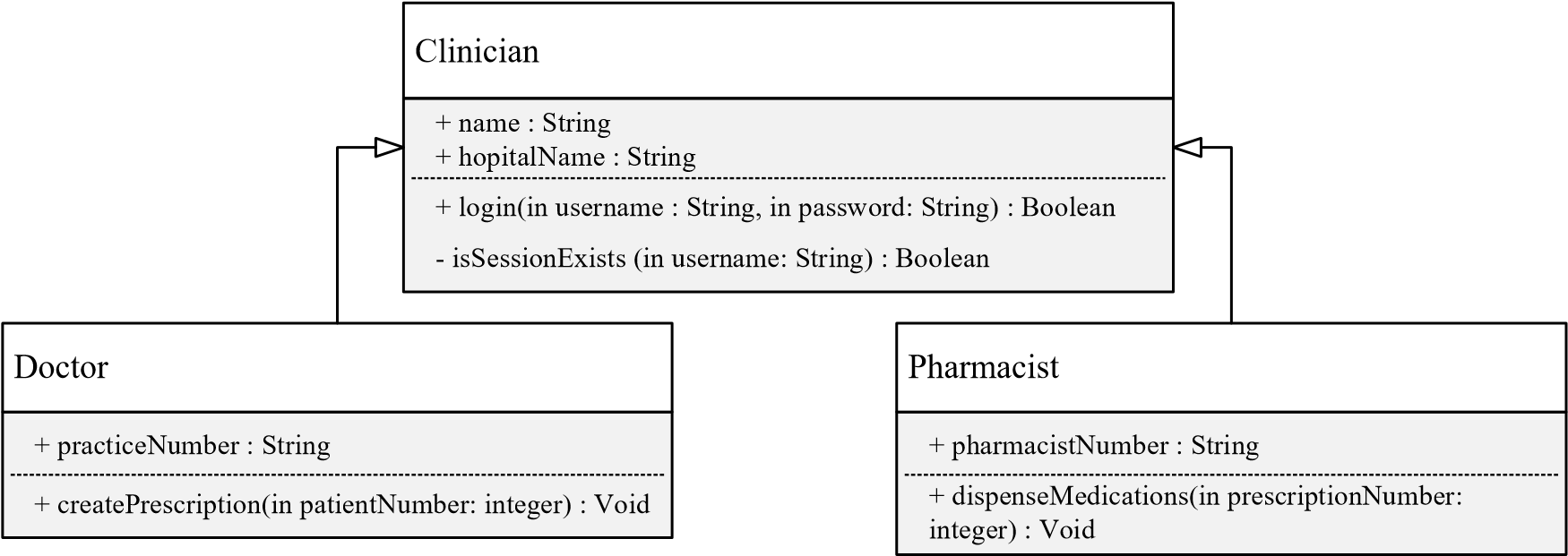
        subObj.Greetings();

    }

}

**Answer to Question No: 5**

1. **Translate the following UML Class Diagram into program code. Write your code in C# programming language.**



**Answer:**

* **Clinician Class:**

public class Clinician

    {

        public string Name { get; set; }

        public string HospitalName { get; set; }

        public bool Login(string username, string password)

        {

            return true;

        }

        private bool IsSessionExists(string username)

        {

            return true;

        }

    }

* **Doctor Class:**

public class Doctor : Clinician

    {

        public string PracticeNumber { get; set; }

        public void CreatePrescription(int patientNumber)

        {

            Console.WriteLine($"Prescription created for patient {patientNumber} by Doctor {Name}");

        }

    }

* **Pharmacist Class:**

public class Pharmacist : Clinician

    {

        public string PharmacistNumber { get; set; }

        public void DispenseMedications(int prescriptionNumber)

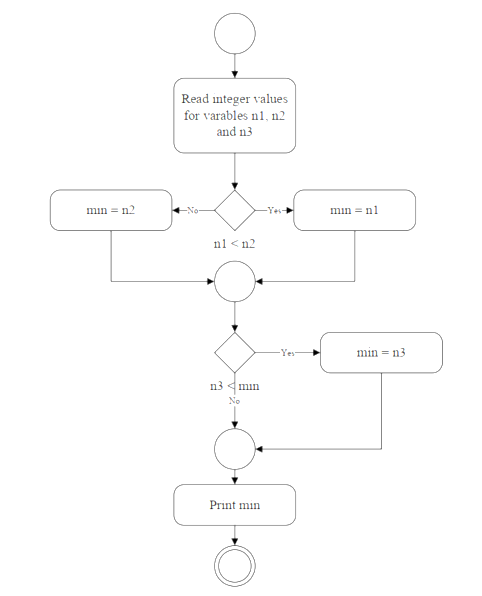
        {

            Console.WriteLine($"Medications dispensed for prescription {prescriptionNumber} by Pharmacist {Name}");

        }

    }

**Answer to Question No: 6**

1. **Translate the UML Activity diagram into program code. Write your code either in C#**

**Answer:**

public class Flowchart

    {

        public void PrintMinNum(int n1, int n2, int n3)

        {

            int min = 0;

            if (n1 < n2)

            {

                min = n1;

            }

            else

            {

                min = n2;

            }

            if (n3 < min)

            {

                min = n3;

            }

            Console.WriteLine($"Minimum Number : {min}");

        }

    }