

软件设计师（中级）

2022 年 05 月下午案例分析真题及答案解析

富国老师（徐富国）

试题一（共 15 分）（202205）

阅读下列说明和图，回答问题 1 至问题 4,将解答填入答题纸的对应栏内。

【说明】

某公司欲开发一款外卖订餐系统，集多家外卖平台和商户为一体,为用户提供在线浏览餐品、订餐和配送等服务。该系统的主要功能是：

1. 入驻管理。用户注册：商户申请入驻，设置按时间段接单数量阈值等。系统存储商户/用户信息。
2. 餐品管理。商户对餐品的基本信息和优惠信息进行发布、修改、删除。系统存储相关信息。
3. 订餐。用户浏览商户餐单，选择餐品及数量后提交订餐请求。系统存储订餐订单。
4. 订单处理。收到订餐请求后，向外卖平台请求配送。外卖平台接到请求后发布配送单，由平台骑手接单，外卖平台根据是否有骑手接单返回接单状态。若外卖平台接单成功，系统给支付系统发送支付请求，接收支付状态。支付成功，更新订单状态为已接单，向商户发送订餐请求并由商户打印订单，给用户发送订单状态；若支付失败，更新订单状态为下单失败，向外卖平台请求取消配送，向用户发送下单失败。若系统接到外卖平台返回接单失败或超时未返回接单状态，则更新订单状态为下单失败，向用户发送下单失败。
5. 配送。商户备餐后，由骑手取餐配送给用户。送达后由用户扫描骑手出示的订单上的配送码后确认送达，订单状态更改为已送达，并发送给商户。
6. 订单评价。用户可以对订单餐品、骑手配送服务进行评价，推送给对应的商户、所在外卖平台，商户和外卖平台对用户的评价进行回复。系统存储评价。现采用结构化方法对外卖订餐系统进行分析与设计，获得如图 1-1 所示的上下文数据流图和图 1-2 所示的 0 层数据流图。

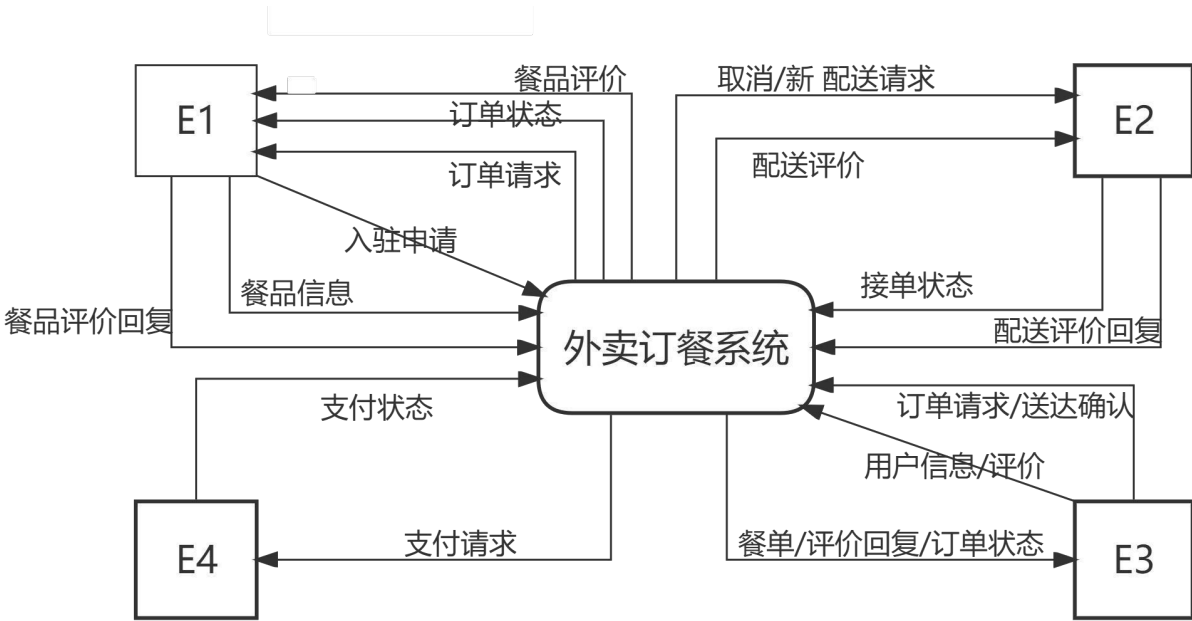


图1-1 上下文数据流图

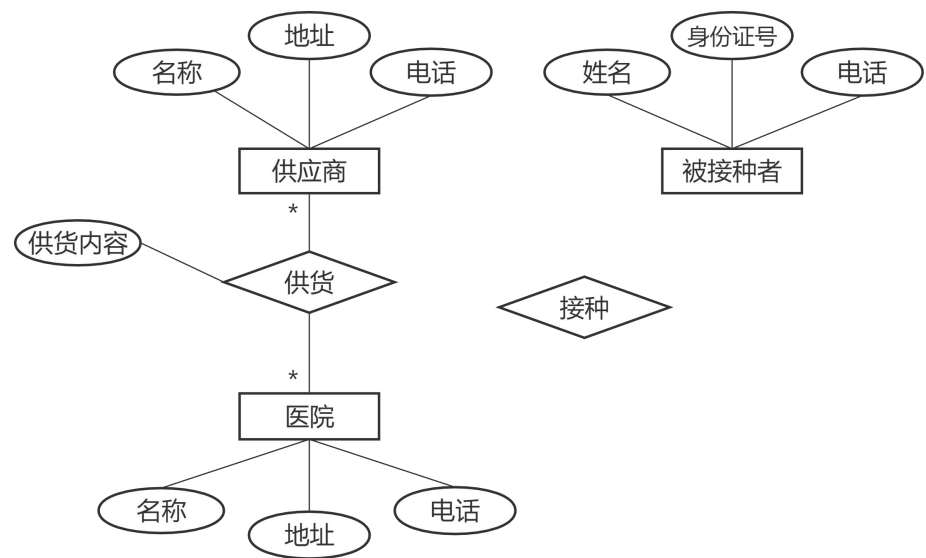


图2-1 E-R图

【逻辑结构设计】

根据概念模型设计阶段完成的实体联系图，得出如下关系模式（不完整）：

供应商（供应商名称、地址、电话）

医院（医院名称、地址、电话）

供货（供应商名称，（a），供货内容）

被接种者（姓名、身份证号、电话）

接种（接种者身份证号，（b），医院名称、供应商名称）

【问题 1】（4 分）

根据问题描述，补充图 2-1 的实体联系图（不增加新的实体）。

【问题 2】（4 分）

补充逻辑结构设计结果中的（a）（b）两处空缺，并标注主键和外键完整性约束。

【问题 3】（7 分）

若医院还兼有核酸检测的业务，检测时可能需要进行多次植酸检测（每天最多检测一次），但每次都可以在全市任意一家医院进行检测。

请在图 2-1 中增加“被检测者”、实体及相应的属性。医院与被检测者之间的“检测”联系及必要的属性，并给出新增加的关系模式。“被检测者”实体包括姓名、身份证号、地址和一个电话。“检测”联系需要包括检测日期和检测结果等。

试题三（共 15 分）（202205）

阅读下列说明和图，回答问题 1 至问题 3，将解答填入答题纸的对应栏内。

【说明】

某公司的人事能门拥有一个地址簿（AddressBook）管理系统（AddressBookSystem），用于管理公司所有员工的地址记录（PersonAddress）。员工的地址记录包括：姓名、住址、城市、省份、邮政编码以及联系电话等信息。管理员可以完成对地址簿中地址记录的管理操作，包括：

- （1）维护地址记录。根据司的人员变动情况，对地址记录进行添加、修改、删除等操作；
- （2）排序。按照员工姓氏的字典顺序或邮政编码对址领中的所有记录。
- （3）打印地址记录。以邮件标签的格式打印一个地址单独的地址簿。系统会记录管理，为便于管理，管理员在系统中为公司的不同部门建立员工对每个地址簿的修改操作，包括：
 - （1）创建地址簿。新建个地址簿并保存。
 - （2）打开地址簿。打开一个已有的地址簿。
 - （3）修改地址簿。对打开的地址簿进行修改并保存系统将提供一个 GUI（图形用户界面）实现对地址簿的各种操作。现采用面向对象方法分析并设计该地址簿管理系统，得到如图 3-1 所示的用例图和图 3-2 所示的类图。

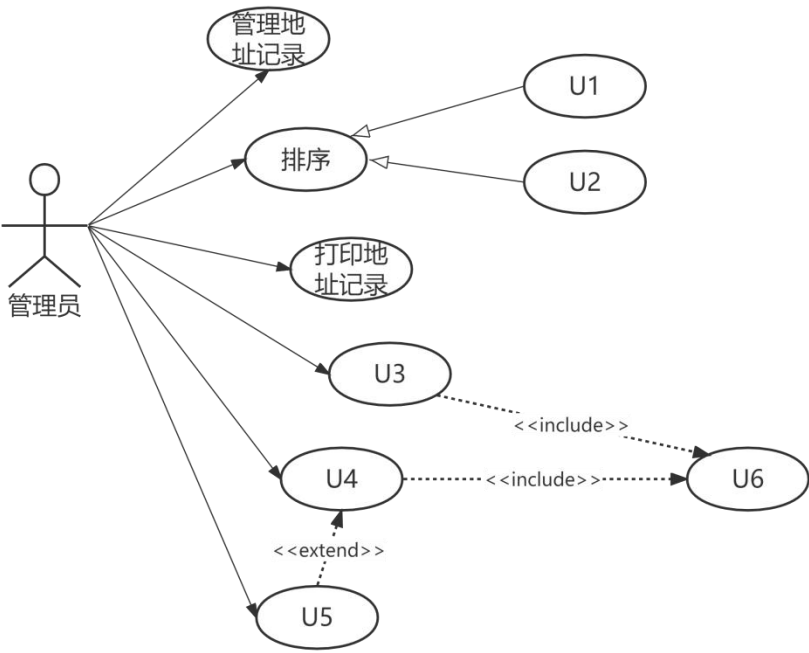


图3-1 用例图

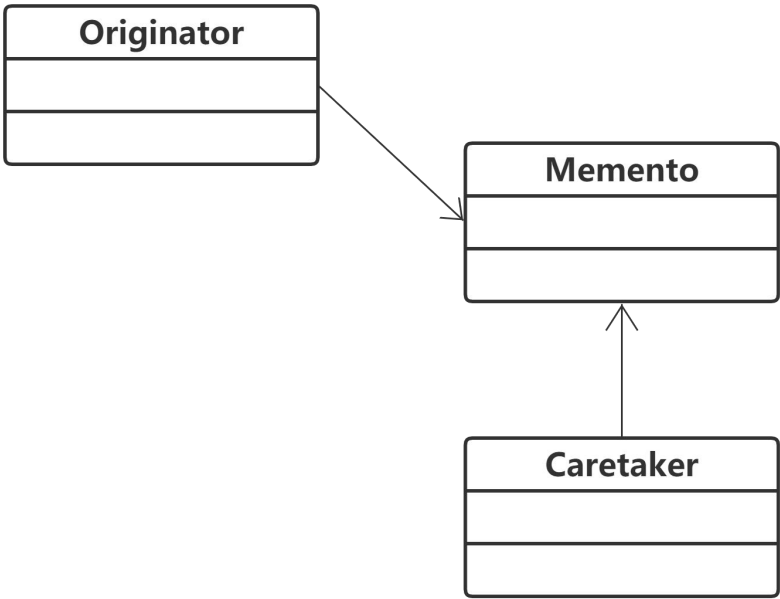


图5-1 类图

- 【问题 1】（6 分）
根据说明中的描述，给出图 3-1 中 U1~U6 所对应的用例名。
- 【问题 2】（5 分）
根据说明中的描述，给出图 3-2 中类 AddressBook 的主要属性和方法以及类 PersonAddress 的主要属性（可以使用说明中的文字）。
- 【问题 3】（4 分）
根据说明中的描述以及图 3-1 所示的用例图，请简要说明 extends 关系的含义是什么？

试题四（共 15 分）（202205）

- 阅读下列说明和 C 代码，回答问题至问题 3，将解答写在答题纸的对应栏内。
- 【说明】
某工程计算中经常要完成多个矩阵相乘（链乘）的计算任务，对矩阵相乘进行以下说明。

(1) 两个矩阵相乘要求第一个矩阵的列数等于第二个矩阵的行数，计算量主要由进行乘法运算的次数决定，假设采用标准的矩阵相乘算法，计算 $A_{m \times n} * B_{n \times p}$ 需要 $m * n * p$ 次行乘法运算的次数决定、乘法运算，即时间复杂度为 $O(m * n * p)$ 。

(2) 矩阵相乘满足结合律，多个矩阵相乘时不同的计算顺序会产生不同的计算量。以矩阵 $A1_{5 \times 100}$ ， $A2_{100 \times 8}$ ， $A3_{8 \times 50}$ 三个矩阵相乘为例，若按 $(A1 * A2) * A3$ 计算，则需要进行 $5 * 100 * 8 + 5 * 8 * 50 = 6000$ 次乘法运算，若按 $A1 * (A2 * A3)$ 计算，则需要进行 $100 * 8 * 50 + 5 * 100 * 50 = 65000$ 次乘法运算。

矩阵链乘问题可描述为：给定 n 个矩阵，对较大的 n ，可能的计算顺序数量非常庞大，用蛮力法确定计算顺序是不实际的。经过对问题进行分析，发现矩阵链乘问题具有最优子结构，即若 $A_1 * A_2 * \dots * A_n$ 的一个最优计算顺序从第 k 个矩阵处断开，即分为 $A_1 * A_2 * \dots * A_k$ 和 $A_{k+1} * A_{k+2} * \dots * A_n$ 两个子问题，则该最优解应该包含 $A_1 * A_2 * \dots * A_k$ 的一个最优计算顺序和 $A_{k+1} * A_{k+2} * \dots * A_n$ 的一个最优计算顺序。据此构造递归式，

$$\text{cost}[i][j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k < j} \text{cost}[i][k] + \text{cost}[k+1][j] + p_i * p_{k+1} * p_{j+1} & \text{if } i < j \end{cases}$$

其中， $\text{cost}[i][j]$ 表示 $A_i + 1 * A_i + 2 * \dots * A_{j+1}$ 的最优计算的计算代价。最终需要求解 $\text{cost}[0][n-1]$ 。

【C 代码】

算法实现采用自底向上的计算过程。首先计算两个矩阵相乘的计算量，然后依次计算 3 个矩阵、4 个矩阵、...、 n 个矩阵相乘的最小计算量及最优计算顺序。下面是该算法的语言实现。

(1) 主要变量说明

n : 矩阵数

$\text{seq}[]$: 矩阵维数序列

$\text{cost}[i][j]$: 二维数组，长度为 $n * n$ ，其中元素 $\text{cost}[i][j]$ 表示 $A_i + 1 * A_i + 2 * \dots * A_{j+1}$ 的最优的计算代价。

$\text{trace}[][]$: 二维数组，长度为 $n * n$ ，其中元素 $\text{trace}[i][j]$ 表示 $A_i + 1 * A_i + 2 * \dots * A_{j+1}$ 的最算对应的划分位置，即 k 。

(2) 函数 cmm

```
#define N100
int cost[N][N];
int trace[N][N];
int cmm(int n,int seq[]){
    int tempCost;
    int tempTrace;
    int i,j,k,p;
    int temp;
    for(i = 0;i < n;i++){ cost[i][i] = 0;}
    for(p = 1;p < n;p++){
        for(i = 0; i < n - p; i++){
            (1) ;
            tempCost = -1;
            for(k = i; (2) ;k++){
                temp = (3) ;
                if(tempCost == -1 || tempCost > temp){
                    tempCost = temp;
                    tempTrace = k;
                }
            }
            cost[i][j] = tempCost;
            (4) ;
        }
    }
    return cost[0][n-1];
}
```

【问题 1】（8 分）

根据以上说明和 C 代码，填充 C 代码中的空 (1) ~ (4)。

【问题 2】（4 分）

根据以上说明和 C 代码，该问题采用了（（5））算法设计策略，时间复杂度为（（6））（用 O 符号表）。

【问题 3】（3 分）

考虑实例 $n = 4$ ，各个矩阵的维数为 A1 为 15×5 ，A2 为 5×10 ，A3 为 10×20 ，A4 为 20×25 ，即维度序列为 15，5，10，20 和 25。则根据上述 C 代码得到的一个最优计算顺序为（7）（用加括号方式表示计算顺序），所需要的乘法运算次数为（8）。

试题五（共 15 分）（202205）

阅读下列说明和 C++ 代码，将应填入（n）处的字句写在答题纸的对应栏内。

【说明】

在软件系统中，通常都会给用户提供取消、不确定或者错误操作的选择，允许将系统恢复到原先的状态。现使用备忘录（Memento）模式实现该要求，得到如图 6-1 所示的类图。Memento 包含了要被恢复的状态。Originator 创建并在 Memento 中存储状态。Caretaker 负责从 Memento 中恢复状态。

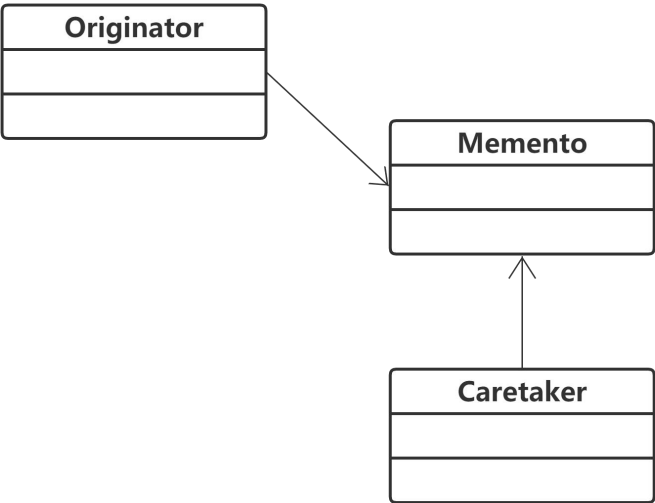


图5-1 类图

【C++ 代码】

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Memento {
private:
    String state;
public:
    Memento(String state) {this -> state = state;}
    string getState(){return state;}
}

class Originator{
private:
    String state;
public:
    void setState(string state) {this -> state = state;}
    string getState() { return state;}
    Memento saveStateToMemento(){
        return ____ (1) ____;
```

```

    }
    void getStateFromMemento(Memento Memento){
        state = ____ (2) ____;
    }
}

class CareTaker {
private:
    vector < Memento > mementoList;
public:
    void ____ (3) ____{
        mementoList.push_back(state);
    }
    void ____ (4) ____{
        return memensoList. (index);
    }
}

int main{
    Originator * originator = new Originator();
    CareTaker * careTaker = new careTaker();
    originator -> setState("State #1");
    originator -> setState("State #2");
    careTaker -> add(____ (5) ____);
    originator -> setState("State #3");
    careTaker -> add(____ (6) ____);
    originator -> setState("State #4");
    cout << "Current State" << " + " << originator -> getState() << endl;
    originator -> getStateFromMemento(careTaker -> get(0);
    cout << "Frist saved State" << originator -> getState() << end;
    originator -> getStateFromMemento(careTaker -> get(1));
    cout << "Second saved State" << " + " << originator -> getState() << endl;
    return 0;
}

```

试题六（共 15 分）（202205）

阅读下列说明和 Java 代码，将应填入（n）处的字句写在答题纸的对应栏内。

【说明】

在软件系统中，通常都会给用户提供一个取消、不确定或者错误操作的选择，允许将系统恢复到原先的状态。现使用备忘录（Memento）模式实现该要求，得到如图 6-1 所示的类图。Memento 包含了要被恢复的状态。Originator 创建并在 Memento 中存储状态。Caretaker 负责从 Memento 中恢复状态。

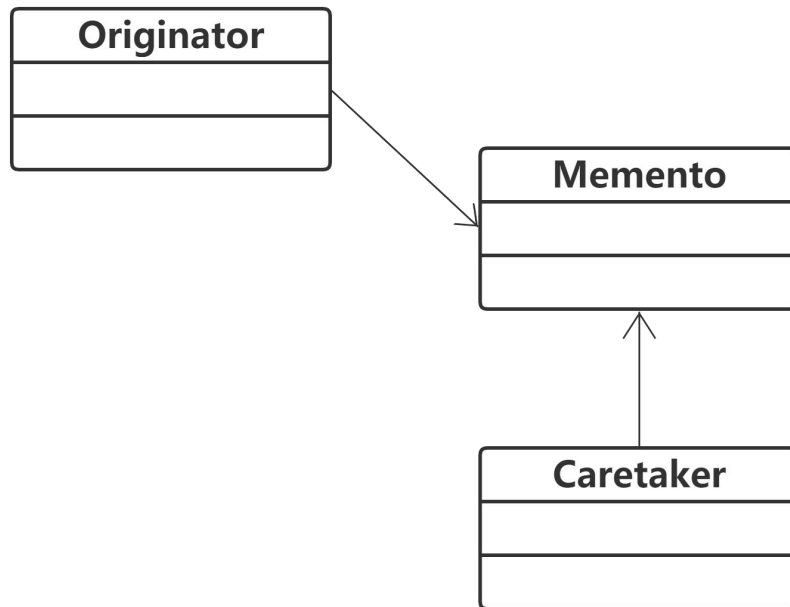


图6-1 类图

【Java 代码】

```
import java.util.* ;
class Memento {
private String state;
public Memento(String state) {this.state = state;}
    public String getState() {return state; }
}

class Originator{
    private String state;
    public void setState(String state) {this.state = state; }
    public String getState() { return state; }
    public Memento saveStateToMemento(){
        return ____ (1) ____;
    }
    public void getStateFromMemento (Memento Memento) {
        state = ____ (2) ____;
    }
}

class CareTaker {
    private List < Memento > mementoList = new ArrayList < Memento > ();
    public ____ (3) ____ {
        mementoList.add(state);
    }
    public ____ (4) ____ {
        return memensoList.get(index);
    }
}
```



```

class MementoPaneDems{
    public static void main(String[] args) {
        Originator originator = new Originator();
        CareTaker careTaker = new careTaker();
        originator.setState("State #1");
        originator.setState("State #2");
        careTaker.add(__(5));
        originator.setState("State #3");
        careTaker.add(__(6));
        originator.setState("State #4");
        System.out.println("Current State" + originator.getState());
        originator.getStateFromMemento(careTaker.get(0));
        System.out.println("Frist saved State" + originator.getState());
        originator.getStateFromMemento(careTaker.get(1));
        System.out.println("Second saved State" + originator.getState());
    }
}

```

参考答案

试题一

【问题 1】

E1: 商户 E2: 外卖平台 E3: 用户 E4: 支付系统

【问题 2】

D1: 用户/商户信息 D2: 订餐订单信息 D3: 餐品信息 D4: 评价信息

【问题 3】

每条数据流 1 分

数据流名称	起点	重点
餐单	P3 或订餐	E3 或用户
订餐请求	P3 或订餐	P4 或订单处理
更新订单状态	P4 或订单处理	D2 或订单信息
配送码	P4 或配送	E3 或用户

【问题 4】

收到订餐请求后，向外卖平台请求配送；外卖平台接到请求后发布配送单，由平台骑手接单；外卖平台根据是否有骑手接单返回接单状态；

IF(外卖平台接单成功) THEN{ 系统给支付系统发送支付请求，接收支付状态；

IF(支付成功) THEN{ 更新订单状态为已接单；向商户发送订餐请求并由商户打印订单；给用户发送订单状态;} ELSE{ 更新订单状态为下单失败；向外卖平台请求取消配送；向用户发送下单失败;} ENDIF}

ELSE IF(系统接到外卖平台返回接单失败或超时未返回接单状态)

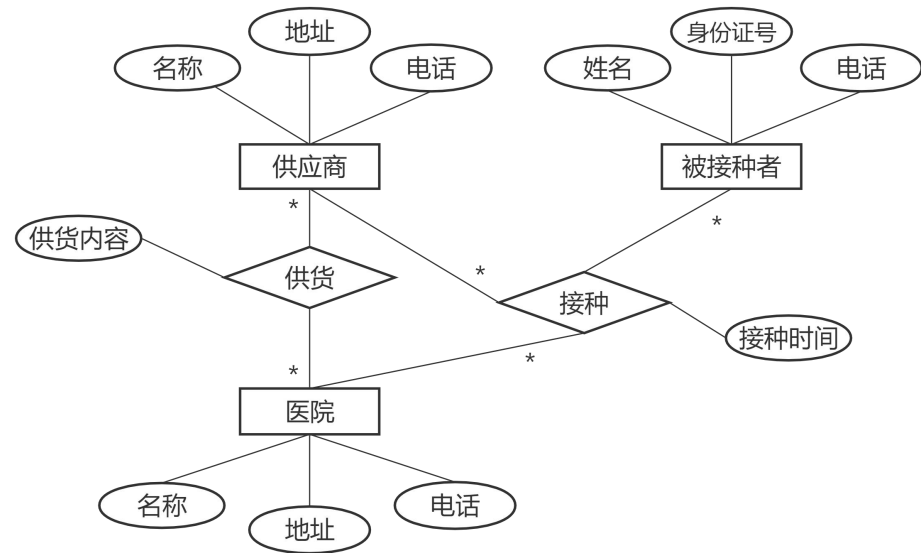
THEN{ 更新订单状态为下单失败；向用户发送下单失败;} ENDIF

}ENDIF

}ENDIF

试题二

【问题 1】

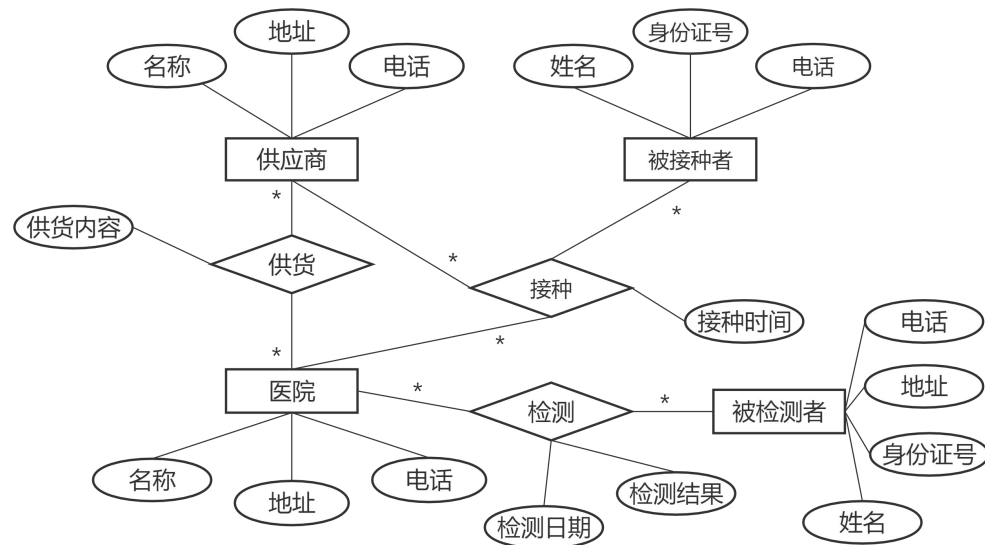


【问题 2】

- a.（医院名称）
- b.（接种日期）

	主键	外键
供货	（供货商名称，医院名称）	供应商名称，医院名称
接种	（接种者身份证号，接种时间）	接种者身份证号，供应商名称，医院名称

【问题 3】



新增关系模式：

- 被检测者（身份证号，姓名，地址，电话）
- 检测（被检测者身份证号，医院名称，检测日期，检测结果）。

试题三

【问题 1】

- U1：按姓氏字典顺序排序；
- U2：按邮政编码排序（U1 和 U2 可互换）
- U3：创建
- U4：修改
- U5：GUI

U6：保存

【问题 2】

类 PersonAddress 也就是员工地址记录，根据题干描述“员工的地址记录包括姓名、住址、城市、省份、邮政编码以及联系电话等信息”，因此类 PersonAddress 的主要属性包括姓名、住址、城市、省份、邮政编码以及联系电话等。

类 AddressBook 也就是地址簿用于管理公司所有员工的地址记录，由于“管理员在系统中为公司的不同部门建立员工的地址簿”，因此类 AddressBook 的主要属性除了员工地址记录信息以外，还需要增加部门信息。此时类 AddressBook 的主要属性包括：部门名/编号，姓名、住址、城市、省份、邮政编码以及联系电话等。“管理地址记录。根据公司的人员变动情况，对地址记录进行添加、修改、删除等操作”，因此类 AddressBook 的需要包括添加、修改、删除等操作。又因为“管理员在系统中为公司的不同部门建立员工的地址簿的操作，包括（1）创建地址簿。（2）打开地址簿。（3）修改地址簿。”，因此类 AddressBook 的需要包括创建地址簿、打开地址簿、修改地址簿。综上，类 AddressBook 的方法包括添加、修改、删除、创建、打开等。

【问题 3】

extend 属于用例图的三种关系之一，表示的是扩展关系。描述为如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能会发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，关系图示指向为扩展用例指向基本用例。include 属于用例图的三种关系之一，表示的是包含关系。描述为当可以从两个或两个以上用例中提取公共行为的时候，应该使用包含关系来表示它们。其中这个提取出来的公共用例称之为抽象用例，而把原始用例称为基本用例和扩展用例。还需结合用例图说明扩展用例、包含用例的应用。此处题干说明不全，暂未确定。

试题四

【问题 1】

1. $j = i + p$
2. $k < j$
3. $cost[i][k] + cost[k+1][j] + sep[i] * sep[k+1] * sep[j+1]$
4. $trace[i][j] = tempTrace;$

【问题 2】

5. 动态规划算法
6. $O(n^3)$

【问题 3】

7. $A1 * ((A2 * A3) * A4)$
8. 5375

试题五

- (1) new Memento(state)
- (2) Memento -> getState()
- (3) add(Memento * state)
- (4) Memento * get(int index)
- (5) originator -> saveStateToMemento()
- (6) originator -> saveStateToMemento()

试题六

- (1) new Memento(state)
- (2) Memento.getState()
- (3) void add(Memento state)
- (4) Memento get(int index)
- (5) originator.saveState ToMemento()
- (6) originator.saveState ToMemento()

淘宝 APP 搜：富国软设

富国老师软件设计师唯一淘宝店地址：

<https://shop441313851.taobao.com/>

或淘宝 APP 扫码



软件设计师学习交流 QQ 群：1169209218

富国老师微信：fuguoruankao

【软考达人】

软考资料免费获取

- 1、最新软考题库
- 2、软考备考资料
- 3、考前压轴题
- 4、免费督考群



微信扫一扫，立马获取



最新免费题库



备考资料+督考群

PC版题库：ruankaodaren.com