



(扫码进入小程序，查看完整版真题答案+解析+估分)

2022 年软件设计师 下午题

问答题（共 15 题，共 33 分）

试题一（共 15 分）

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。

问题 1（4 分）

使用说明中的词语，给出图 1-1 的实体 E1~E4 的名称。

问题 2（4 分）

使用说明中的词语，给出图 1-2 中的数据存储 D1-D4 的名称。

问题 3（4 分）

根据说明和图中术语，补充图 1-2 中缺失的数据流及其起点和终点。

问题 4（3 分）

根据说明，采用结构化语言对“订单处理”的加工逻辑进行描述。



（扫码进入小程序，查看完整版真题答案+解析+估分）

试题二

按照下列图表，填写答题纸的对应栏内。

[说明]

为了提高接种工作，提高效率，并未了抗击疫情提供疫苗接种数据支撑，需要开发一个信息系统，下述需求完成该系统的数据库设计。

- （1）记录疫苗供应商的信息，包括供应商名称，地址和一个电话。
- （2）记录接种医院的信息，包括医院名称、地址和一个电话。
- （3）记录接种者个人信息，包括姓名、身份证号和一个电话。
- （4）记录接种者疫苗接种信息，包括接种医院信息，被接种者信息，疫苗供应商名称和接种日期，为了提高免疫力，接种者可能需要进行多次疫苗接种，（每天最多接种一次，每次都可以在全市任意一家医院进行疫苗接种）。

【概念模型设计】

阶段的信息，设计的实体联系图（不完整）如图 2-1 所示。

【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。



（扫码进入小程序，查看完整版真题答案+解析+估分）

[问题 3] (7 分)

若医院还兼有核酸检测的业务，检测石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。在址和一个电话。“检测”联系需要包括检测日期和检测结果等。

试题三

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 (AddressBookSystem)，管理系统 ddressBookSystem)，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。



（扫码进入小程序，查看完整版真题答案+解析+估分）

[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 Adresk 的主要属性和方法以及类 Pernodaddress 的主要属性（可以使用说明中的文字）。

[问题 3] (4 分)

的相形送明中的描述以及图 31 所示的用例图，请阅 enedecx 系 cxe 关的含义是什么？

试题四 (共 15 分)

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】

工程计算中经常要完成多个矩阵相乘的计算任务，对矩郑相乘进行以下说明。

(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 mnp 次行乘法运算的次数决定、乘法运算，即时间复杂度为 $O(mn \times p)$ 。

(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 $A_{15 \times 100}$, $A_{2100 \times 8}$, $A_{38 \times 50}$ 三个矩阵相乘为例，若按 $(A_1 A_2) A_3$ 计算，则需要进行 $51008 + 5850 - 6000$ 次乘法运算，若按 $A_1 (A_2 A_3)$ 计算，则需要进行 $100850 + 5100 \times 50 = 65000$ 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，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， $\text{cost}[ij]$ 表示 $A_i \dots A_j$ 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。



(扫码进入小程序，查看完整版真题答案+解析+估分)

(1) 主要变量说明 n : 矩阵数 $\text{seq}[]$: 矩阵维数序列 $\text{cost}[]$: 二维数组，长度为 nn ，其中元素 $\text{cost}[i][j]$ 表示 $A_i \dots A_j$ 的最优的计算代价 $\text{trace}[]$: 二维数组，长度为 nn ，其中元素 $\text{trace}[i][j]$ 表示 $A_i \dots A_j$ 的最算对应的划分位置，即 k

(2) 函数 cmm 实现 $N100\text{ost}[N][N]$

```

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][i+p] = 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 A_1 为 155， A_2 为 510， A_3 为 1020， A_4 为 2025，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)



(扫码进入小程序，查看完整版真题答案+解析+估分)

试题五 (共 15 分)

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。



(扫码进入小程序，查看完整版真题答案+解析+估分)

【说明】

在软件系统中，通常不会给用户提供取消、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 5-1 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。

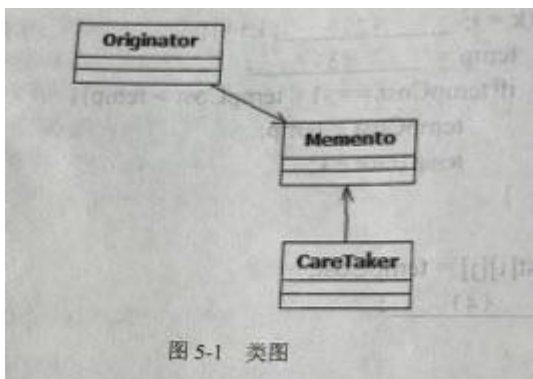


图 5-1 类图

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 addState(Memento memento){mementoList.push back (state)
```

```
__(4) _;return mementoList (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") ;
```




(扫码进入小程序，查看完整版真题答案+解析+估分)

```
cout <<"Current State:"<<"+" <<originator->getState()<<endl;
originator->getStateFromMemento(careTaker->get(0);
cout<<"First saved State:"<<originator->getStateee()<<endl;
originator->getStateFromMemento(careTaker->get(1);
cout<<"second save State"<<"+" <<originator->getState()<<endl;return 0;}
```

试题六 (共 15 分)

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。

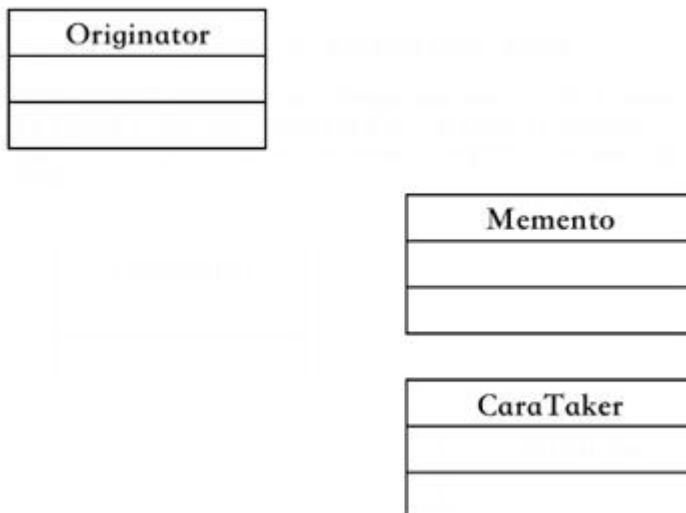


图 6-1 类图

图 6-1 类图

【Uava 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)
```



(扫码进入小程序，查看完整版真题答案+解析+估分)

```
public void getStateFromMemento (Memento Memento) {
state = (2)
```

```
class CareTaker
```

```
Public (3)
```

```
mementoLstadd 如 aney
```

```
public (4)
return memensoLint get (imdat
}
}
```

```
class MementoPaneDems{  
public static void main (Stringf]  
Ohgiator aniginutor-te Origmrfy  
CareThker careTukeznwCmiie)  
aniginuntorseStte (State1#)  
orginxtorsCSuat (State2#);  
careThiket add_ (5)  
oiginatoseSated" (State 3")  
careTaker add (6) _  
originutor_setState (State#4)
```



(扫码进入小程序，查看完整版真题答案+解析+估分)