

2014年下半年 软件设计师 下午试卷 案例 [真题]

溯源编码：84221201908150252

文档生成日期：2019年08月21日

PC+微信+纸质，立体化学习场景，陪伴你每时每刻。

软考在线 <http://www.rkpass.cn>

.....

以下所有试题由 软考在线 免费智能真题库 提供

软考在线 -- 最专业的一站式软考复习平台

全网独家 免费智能真题库 定制学习计划

专业致力于全国计算机技术与软件专业资格(水平)考试

使用说明：

溯源编码：

在软考在线PC版，“文档溯源”功能中，输入文档溯源编码，即可获知本文档是否为最新文档。

软考在线每天都会完善试题内容质量，更新试题统计数据。同时定期更新文档。

“文档溯源”功能位置：首页->复习资料->试题文档->文档溯源

二维码：

微信扫一扫，直达更多延伸内容。

打印：

文档已排好版，直接打印即可(A4纸)。

第1题 2014下

(共15分)

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】

某大型披萨加工和销售商为了有效管理生产和销售情况，欲开发一披萨信息系统，其主要功能如下：

(1) 销售。处理客户的订单信息，生成销售订单，并将其记录在销售订单表中。销售订单记录了订购者、所订购的披萨、期望的交付日期等信息。

(2) 生产控制。根据销售订单以及库存的披萨数量，制定披萨生产计划（包括生产哪些披萨、生产顺序和生产量等），并将其保存在生产计划表中。

(3) 生产。根据生产计划和配方表中的披萨配方，向库存发出原材料申领单，将制作好的披萨的信息存入库存表中，以便及时进行交付。

(4) 采购。根据所需原材料及库存量，确定采购数量，向供应商发送采购订单，并将其记录在采购订单表中；得到供应商的供应量，将原材料数量记录在库存表中，在采购订单表中标记已完成采购的订单。

(5) 运送。根据销售订单将披萨交付给客户，并记录在交付记录表中。

(6) 财务管理。在披萨交付后，为客户开具费用清单，收款并出具收据；依据完成的采购订单给供应商支付原材料费用并出具支付细节；将收款和支付记录存入收支记录表中。

(7) 存储。检查库存的原材料、披萨和未完成订单，确定所需原材料。

现采用结构化方法对披萨信息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。



本题

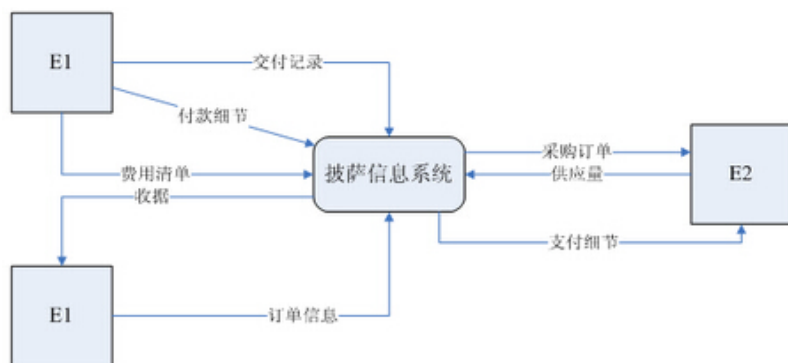


图1-1 上下文数据流图

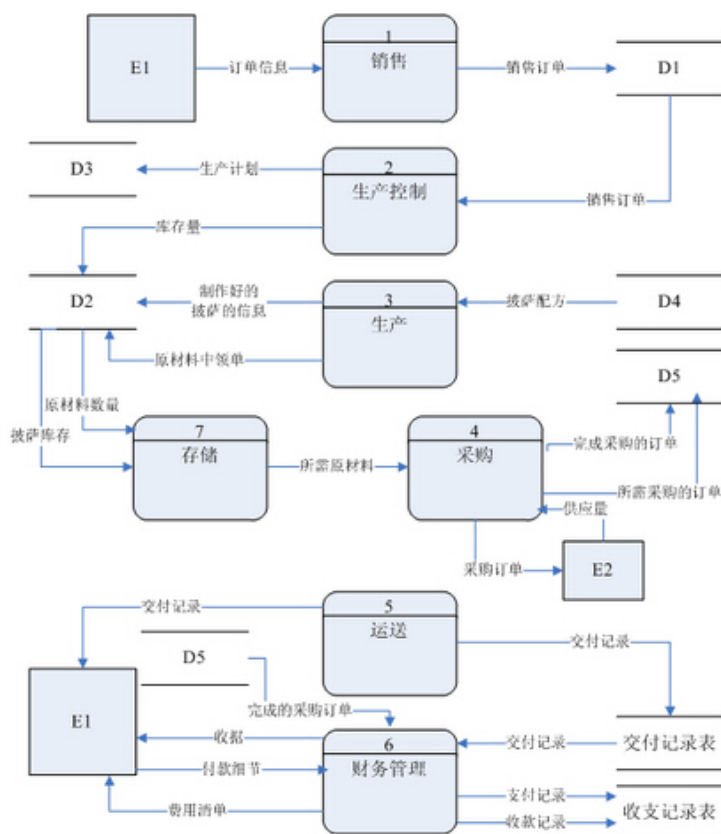


图1-2 0层数数据流图

问题1.1 根据说明中的词语，给出图1-1中的实体E1~E2的名称。

问题1.2 根据说明中的词语，给出图1-2中的数据存储D1~D5的名称。

问题1.3 根据说明和图中词语，补充图1-2中缺失的数据流及其起点和终点。

第2题 2014下

(共15分)

阅读下列说明，回答问题1至问题3，将解答填入答题纸的对应栏内。

【说明】

【说明】某集团公司在全国不同城市拥有多个大型超市，为了有效管理各个超市的业务工作，需要构

建一个超市信息管理系统。

【需求分析结果】

- (1) 超市信息包括：超市名称、地址、经理和电话，其中超市名称唯一确定超市关系的每一个元组。每个超市只有一名经理。
- (2) 超市设有计划部、财务部、销售部等多个部门，每个部门只有一名部门经理，有多名员工，每个员工只属于一个部门。部门信息包括：超市名称、部门名称、部门经理和联系电话。超市名称、部门名称唯一确定部门关系的每一个元组。
- (3) 员工信息包括：员工号、姓名、超市名称、部门名称、职位、联系方式和工资。其中，职位信息包括：经理、部门经理、业务员等。员工号唯一确定员工关系的每一个元组。
- (4) 商品信息包括：商品号、商品名称、型号、单价和数量。商品号唯一确定商品关系的每一个元组。一名业务员可以负责超市内多种商品的配给，一种商品可以由多名业务员配给。



本题

【概念模型设计】

根据需求分析阶段收集的信息，设计的实体联系图和关系模式（不完整）如下：

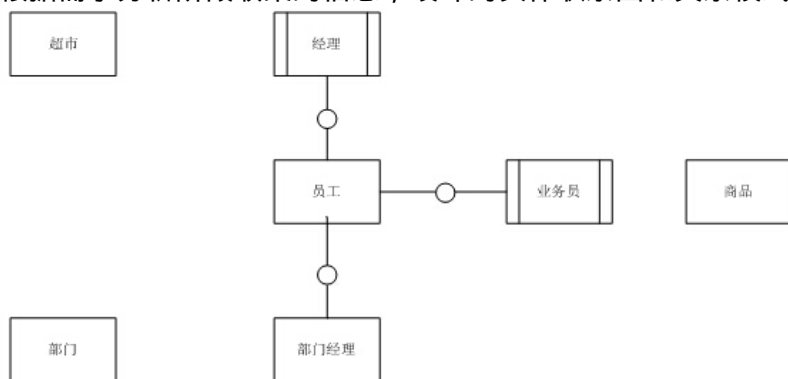


图1-1 实体联系图

【关系模式设计】

- 超市（超市名称，经理，地址，电话）
- 部门（<u>（a）</u>，部门经理，联系电话）
- 员工（<u>（b）</u>，姓名，联系方式，职位，工资）
- 商品（商品号，商品名称，型号，单价，数量）
- 配给（<u>（c）</u>，配给时间，配给数量，业务员）

问题2.1 根据问题描述，补充四个联系，完善图1-1的实体联系图。联系名可用联系1、联系2、联系3和联系4代替，联系的类型分为1:1、1:n和m:n（或1:1、1:*和*:*）。

问题2.2 （1）根据实体联系图，将关系模式中的空（a）~（c）补充完整；
（2）给出部门和配给关系模式的主键和外键。

问题2.3 （1）超市关系的地址可以进一步分为邮编、省、市、街道，那么该属性是属于简单属性还是复合属性？请用100字以内文字说明。

（2）假设超市需要增设一个经理的职位，那么超市与经理之间的联系类型应修改为<u>（d）</u>，超市关系应修改为<u>（e）</u>。

第3题 2014下

（共15分）

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】

某公司欲开发一个管理选民信息的软件系统。系统的基本需求描述如下：

- （1）每个人(Person)可以是一个合法选民(Eligible)或者无效的选民(Ineligible)。
- （2）每个合法选民必须通过该系统对其投票所在区域（即选区，Riding）进行注册(Registration)。每个合法选民仅能注册一个选区。
- （3）选民所属选区由其居住地址(Address)决定。假设每个人只有一个地址，地址可以是镇



本题

(Town)或者城市(City)。

(4) 某些选区可能包含多个镇；而某些较大的城市也可能包含多个选区。
现采用面向对象方法对该系统进行分析与设计，得到如图1-1所示的初始类图。

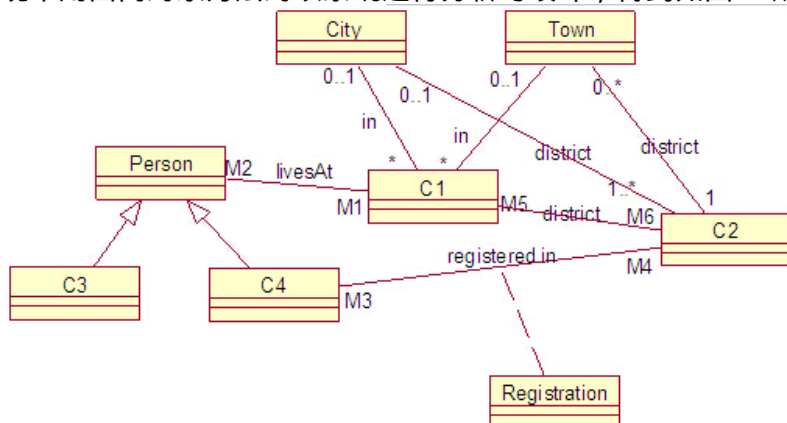


图1-1类图

问题3.1 根据说明中的描述，给出图1-1中C1～C4所对应的类名（类名使用说明中给出的英文词汇）。

问题3.2 根据说明中的描述，给出图1-1中M1～M6处的多重度。

问题3.3 对该系统提出了以下新需求：

- （1）某些人拥有在多个选区投票的权利，因此需要注册多个选区；
 - （2）对于满足（1）的选民，需要划定其“主要居住地”，以确定他们应该在哪个选区进行投票。
- 为了满足上述需求，需要对图3-1所示的类图进行哪些修改？请用100字以内文字说明。

第4题 2014下



本题

阅读下列说明和C代码，回答问题1至问题3，将解答写在答题纸的对应栏内。

【说明】

计算一个整数数组a的最长递增子序列长度的方法描述如下：

假设数组a的长度为n，用数组b的元素b[i]记录以a[i](0≤i<n)为结尾元素的最长递增子序列的长度，则数组a的最长递增子序列的长度为 $\max_{0 \leq i < n} \{b[i]\}$ ；其中b[i]满足最优子结构，可递归定义为：

$$\begin{cases} b[0] = 1 \\ b[i] = \max_{0 \leq k < i} \{b[k]\} + 1 \end{cases}$$

【C代码】

下面是算法的C语言实现。

(1) 常量和变量说明

a：长度为n的整数数组，待求其最长递增子序列

b：长度为n的数组，b[i]记录以a[i](0≤i<n)为结尾元素的最长递增子序列的长度，其中0≤i<n

len：最长递增子序列的长度

ij：循环变量

temp：临时变量

(2) C程序

```
#include <stdio.h>
int maxL(int* b, int n) {
    int i, temp=0;
    for(i=0; i<n; i++) {
        if(b[i]>temp)
            temp=b[i];
    }
    return temp;
}

int main() {
    int n, a[100], b[100], i, j, len;
    scanf("%d", &n);
    for(i=0; i<n; i++) {
        scanf("%d", &a[i]);
    }
    (1) ____;
    for(i=1; i<n; i++) {
        for(j=0, len=0; (2) ____; j++) {
            if((3) ____ && len<b[j])
                len=b[j];
        }
        (4) ____;
    }
    printf("len:%d\n", maxL(b,n));
    printf("\n");
}
```

问题4.1 根据说明和C代码，填充C代码中的空(1)~(4)。

问题4.2 根据说明和C代码，算法采用了(5)设计策略，时间复杂度为(6) (用O符号表示)。

问题4.3 已知数组a={3,10,5,15,6,8}，根据说明和C代码，给出数组b的元素值。

第5题 2014下

阅读下列说明和C++代码，将应填入(n)处的语句写在答题纸的对应栏内。

【说明】

某灯具厂商欲生产一个灯具遥控器，该遥控器具有7个可编程的插槽，每个插槽都有开关按钮，对应着一个不同的灯。利用该遥控器能够统一控制房间中该厂商所有品牌灯具的开关，现采用Command (命令) 模式实现该遥控器的软件部分。Command模式的类图如图1-1所示。

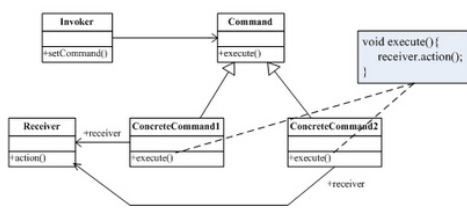


图1-1 Command模式类图



本题

问题5.1

```

【C++代码】
class Light {
public:
    Light(string name) { /* 代码省略 */ }
    void on() { /* 代码省略 */ } // 开灯
    void off() { /* 代码省略 */ } // 关灯
};
class Command {
public:
    _____ (1) _____;
};
class LightOnCommand:public Command { // 开灯命令
private:
    Light* light;
public:
    LightOnCommand(Light* light) { this->light=light; }
    void execute() { _____ (2) _____; }
};
class LightOffCommand:public Command { // 关灯命令
private:
    Light *light;
public:
    LightOffCommand(Light* light) { this->light=light; }
    void execute() { _____ (3) _____; }
};
class RemoteControl{ // 遥控器
private:
    Command* onCommands[7];
    Command* offCommands[7];
public:
    RemoteControl() { /* 代码省略 */ }
    void setCommand(int slot, Command* onCommand, Command* offCommand) {
        _____ (4) _____=onCommand;
        _____ (5) _____=offCommand;
    }
    void onButtonWasPushed(int slot) { _____ (6) _____; }
    void offButtonWasPushed(int slot) { _____ (7) _____; }
};
int main() {
    RemoteControl* remoteControl=new RemoteControl();
    Light* livingRoomLight=new Light("Living Room");
    Light* kitchenLight=new Light("kitchen");
    LightOnCommand* livingRoomLightOn=new LightOnCommand(livingRoomLight);
    LightOffCommand* livingRoomLightOff=new LightOffCommand(livingRoomLight);
    LightOnCommand* kitchenLightOn=new LightOnCommand(kitchenLight);
    LightOffCommand* kitchenLightOff=new LightOffCommand(kitchenLight);
    remoteControl->setCommand(0, livingRoomLightOn, livingRoomLightOff);
    remoteControl->setCommand(1, kitchenLightOn, kitchenLightOff);
    remoteControl->onButtonWasPushed(0);
    remoteControl->offButtonWasPushed(0);
    remoteControl->onButtonWasPushed(1);
    remoteControl->offButtonWasPushed(1);
    /* 其余代码省略 */
    return 0;
}

```

第6题 2014下

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】

某灯具厂商欲生产一个灯具遥控器，该遥控器具有7个可编程的插槽，每个插槽都有开关灯具的开关，现采用Command（命令）模式实现该遥控器的软件部分。Command模式的类图如图1-1所示。

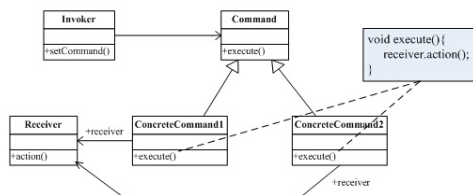


图1-1 Command模式类图



本题

问题6.1

【Java代码】

```
class Light {
    public Light() {}
    public Light(String name) { /* 代码省略 */ }
    public void on() { /* 代码省略 */ } // 开灯
    public void off() { /* 代码省略 */ } // 关灯
    // 其余代码省略
}

__ ( 1 ) __ {
    public void execute();
}

class LightOnCommand implements Command { // 开灯命令
    Light light;
    public LightOnCommand(Light light) { this.light=light; }
    public void execute() { __ ( 2 ) __ ; }
}

class LightOffCommand implements Command { // 关灯命令
    Light light;
    public LightOffCommand(Light light) { this.light=light; }
    public void execute(){ __ ( 3 ) __ ; }
}

class RemoteControl { // 遥控器
    Command[] onCommands=new Command[7];
    Command[] offCommands=new Command[7];
    public RemoteControl() { /* 代码省略 */ }
    public void setCommand(int slot, Command onCommand, Command offCommand) {
        __ ( 4 ) __ =onCommand;
        __ ( 5 ) __ =offCommand;
    }
    public void onButtonWasPushed(int slot) {
        __ ( 6 ) __ ;
    }
    public void offButtonWasPushed(int slot){
        __ ( 7 ) __ ;
    }
}

class RemoteLoader {
    public static void main(String[] args) {
        RemoteControl remoteControl=new RemoteControl();
        Light livingRoomLight=new Light("Living Room");
        Light kitchenLight=new Light("kitchen");
        LightOnCommand livingRoomLightOn=new LightOnCommand(livingRoomLight);
        LightOffCommand livingRoomLightOff=new LightOffCommand(livingRoomLight);
        LightOnCommand kitchenLightOn=new LightOnCommand(kitchenLight);
        LightOffCommand kitchenLightOff=new LightOffCommand(kitchenLight);
        remoteControl.setCommand(0, livingRoomLightOn, livingRoomLightOff);
        remoteControl.setCommand(1, kitchenLightOn, kitchenLightOff);
        remoteControl.onButtonWasPushed(0);
        remoteControl.offButtonWasPushed(0);
        remoteControl.onButtonWasPushed(1);
        remoteControl.offButtonWasPushed(1);
    }
}
```