全国计算机技术与软件专业技术资格（水平）考试

中级 软件设计师 2014 年 下半年 下午试卷 案例

（考试时间 150 分钟）

试题一 （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、拔萨和未完成订单，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。

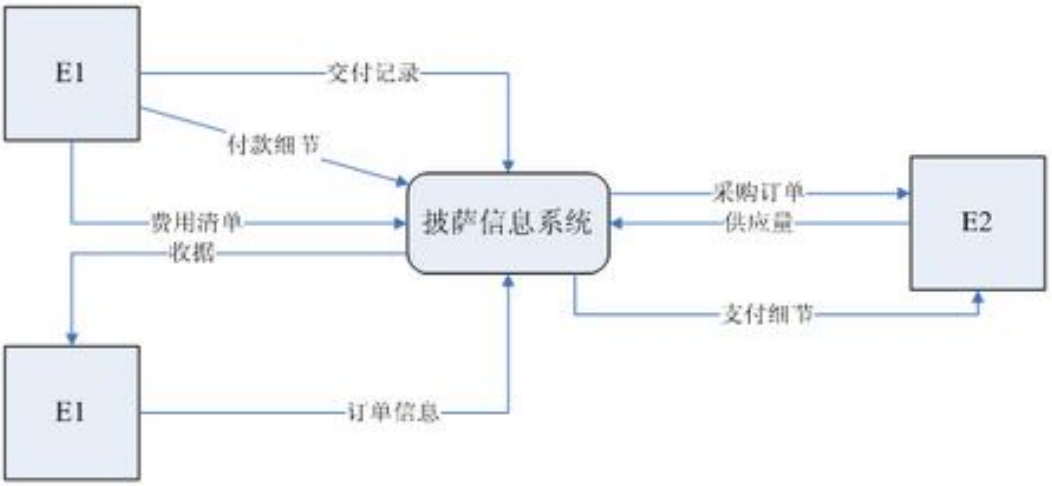


图1-1 上下文数据流图

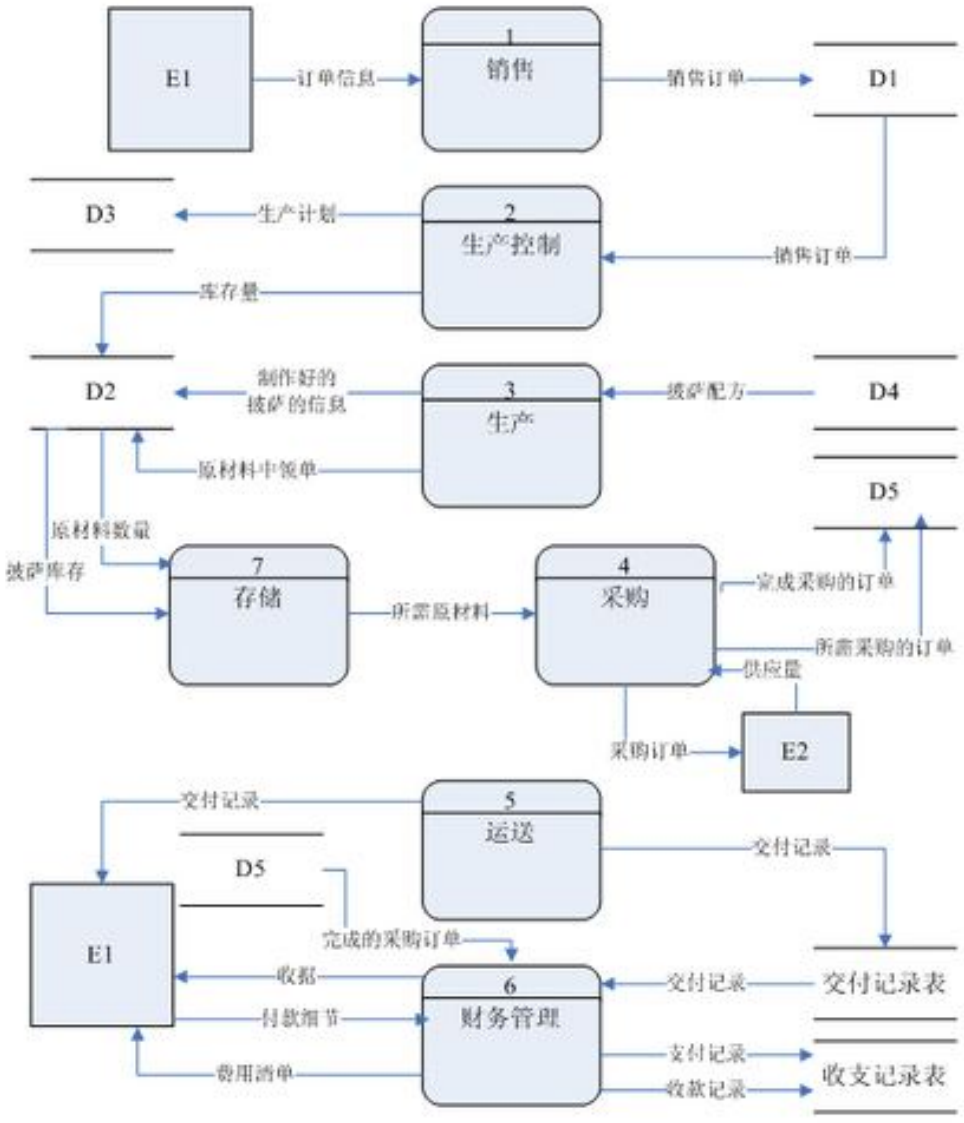


图1-2　0层数数据流图

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。

试题二 （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】

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。

【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：

【关系模式设计】

超市(超市名称，经理，地址，电话)

部门(_(a)_，部门经理，联系电话)

员工(_(b)_，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(_(c)_，配给时间，配给数量，业务员)

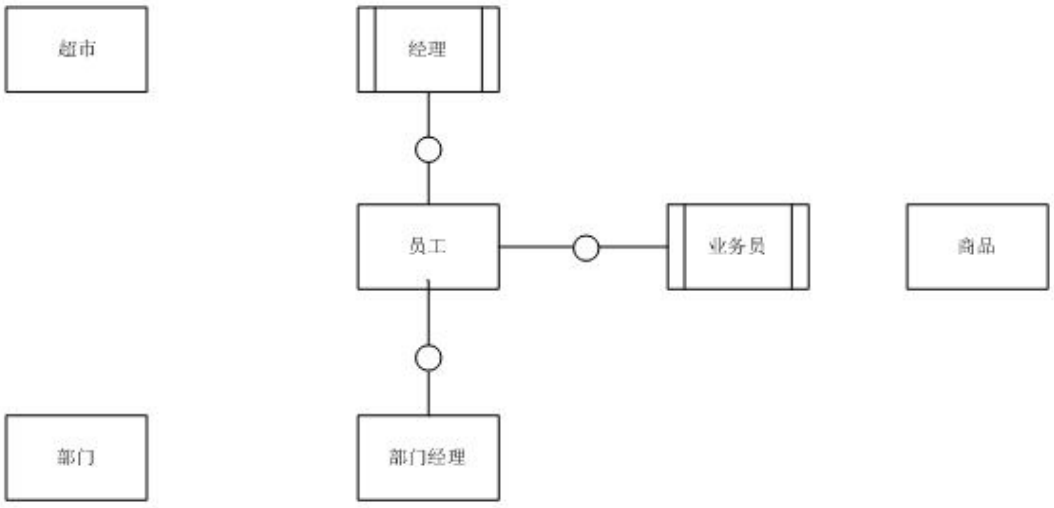


图1-1 实体联系图

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（d），超市关系应修改为（e）。

试题三 （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(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。

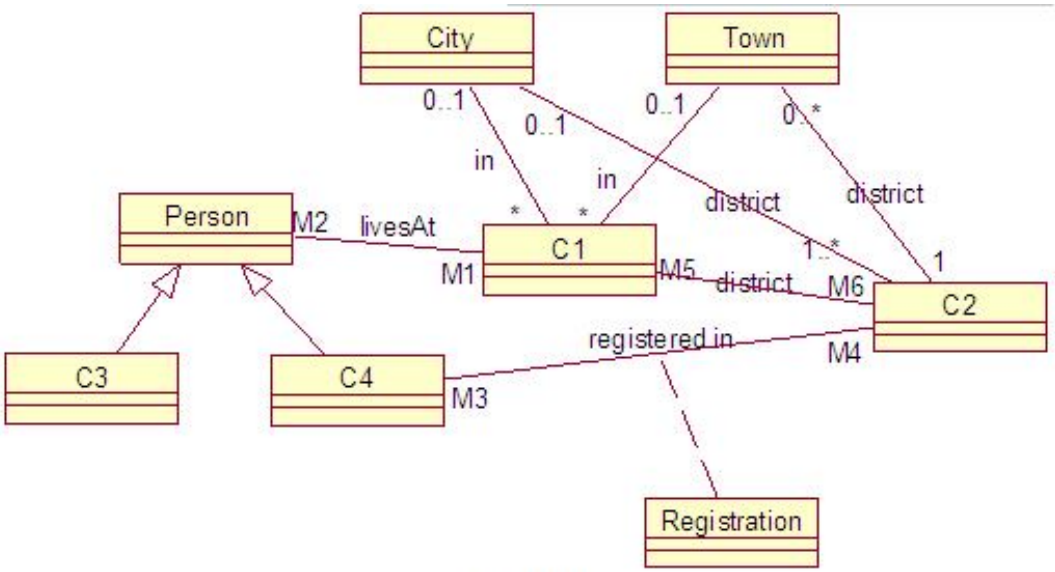


图1-1类图

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，需要对阁 3-1 所示的类图进行哪些修改？请用 100 字以内文字说明。

试题四

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予序列的长度，则数组a的最长递增子序列的长度为$\max_{0\le i<n}\{b[i]\}$；其中b[i]满足最优子结构，可递归定义为：

$$\begin{cases} b[0]=1 \\ b[i]=\max\limits_{\substack{0\le k<i \\ a[k]\le a[i]}}\{b[k]\}+1 \end{cases}$$

【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

i,j：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。

试题五

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】

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。

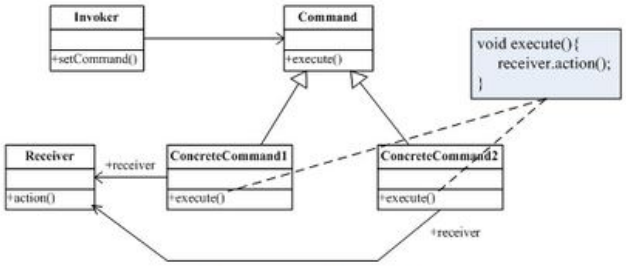


图1-1 Command模式类图

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=newLightOffCommand(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;
}
```

试题六

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。

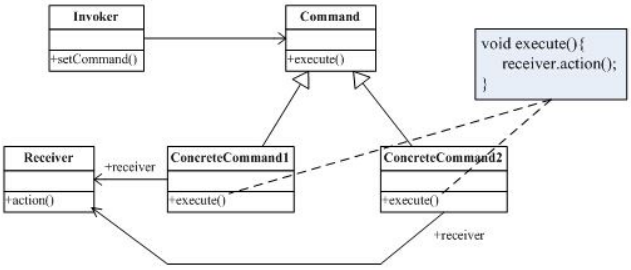


图1-1 Command模式类图

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 offlButtonWasPushed(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);
  }
}
```

试题一 答案： 解析：

答案

E1 ：客户； E2 ：供应商

【试题分析】

本题考查采用结构化方法进行系统分析与设计，主要考查数据流图(DFD)的应用，是比较传统的题目，考点与往年类似，要求考生细心分析题目中所描述的内容。

DFD 是一种便于用户理解、分析系统数据流程的图形化建模工具，是系统逻辑模型的重要组成部分。上下文 DFD （顶层 DFD)通常用来确定系统边界，将待开发系统看作一个大的加工(处理)，然后根据系统从哪些外部实体接收数据流，以及系统将数据流发送到哪些外部实体，建模出的上下文图中只有唯一的一个加工和一些外部实体，以及这两者之间的输入输出数据流。 0 层 DFD 在上下文确定的系统外部实体以及与外部实体的输入输出数据流的基础上，将上下文 DFD 中的加工分解成多个加工，识别这些加工的输入输出数据流，使得所有上下文 DFD 中的输入数据流，经过这些加工之后变换成上下文 DFD 的输出数据流。根据 0 层 DFD 中加工的复杂程度进一步建模加工的内容。

在建分层 DFD 时，根据需求情况可以将数据存储建模在不同层次的 DFD 中，注意要在绘制下层数据流图时要保持父图与子图平衡。父图中某加工的输入输出数据流必须与其子图的输入输出数据流在数量和名字上相同，或者父图中的一个输入(或输出)数据流对应于子图中几个输入(或输出)数据流，而子图中组成这些数据流的数据项全体正好是父图中的这一个数据流。

本问题考查上下文 DFD ，要求确定外部实体。考察系统的主要功能，不难发现，系统中涉及到客户和供应商，没有提到其他与系统交互的外部实体。根据描述(1) 中“处理客户的订单信息”等信息、从供应商出进行采购时“向供应商发送采购订单”，从而可确定 E1 为“客户”实体， E2 为“供应商”实体。

答案

D1 ：销售订单表； D2 ：库存表； D3 ：生产计划表； D4 ：配方表； D5 ：采购订单表

【试题分析】

本问题要求确定 0 层数据流图中的数据存储。分析说明中和数据存储有关的描述，根据说明中(1) 客户的订单信息，生成销售订单，“并将其记录在销售订单表”，可知 D1 为销售订单表；说明中(2) 根据销售订单以及库存的匹萨数量制定匹萨生产计划，并将其保存在生产计划表中；说明(3) 中“根据生产计划和配方表中的匹萨配方”、“将制作好的匹萨的信息存入库存表中”，可知 D2 为库存表、 D3 为生产计划表、 D4 为匹萨配方表；说明(4) 中向供应商发送采购订单，“并将其记录在采购订单表中”，可能 D5 为采购订单表。

答案

【试题分析】

本问题要求补充缺失的数据流及其起点和终点。对照图 1-1 和图 1-2 的输入、输出数据流，数量不同，考查图 1-1 中输出至 E2 的数据流，有“采购订单”和“支付细节”，而图 1-2 中缺少了 “支付细节”数据流，所以需要确定此数据流或者其分解的数据流的起点。考查说明中的功能，功能(3) 根据生产计划和配方表中的匹萨配方，向库存发出原材料申领单，对照图 1-2, 不难发现，处理 3 缺少了输入流生产计划，而生产计划存储在“生产计划表”中。功能(4) 根据所需原材料及库存量，确定采购数量，可以看出处理(4) 缺少了从库存表中获取的库存量；得到供应商的供应量，将原材料数量记录在库存表中，可以发现，缺少了从处理(4) 到库存表的数据流“原材料数量”。图 1-2 中处理(5) 没有输入流，考察功能(5) 根据销售订单将匹萨交付给客户，可知，处理 5 的输入数据流为“销售订单”，其来源为数据存储订单记录表。功能(6) 中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，可以看出缺少了支付细节输出流，结合对比图 1-1 和图 1-2, 可知支付细节并未分解。功能(7) 检查库存的原材料、匹萨和未完成订单，不难发现，处理 7 缺少输入流未完成订单，客户订单在数据存储“销售订单表”中。

数 据 流	起 点	终 点
生产计划	D3 或 生产计划表	3 或 生产
未完成订单	D1 或 销售订单表	7 或 存储
销售订单	D1 或 销售订单表	5 或 运送
原材料数量	4 或 采购	D2 或 库存表
库存量	D2 或 库存表	4 或 采购
支付细节	6 或 财务管理	E2 或 供应商

试题二 答案： 解析：

答案

联系名称可不做要求，但不能出现重名。

【试题分析】

本题考查数据库概念结构设计及其向逻辑结构转换的过程。

此类题目要求考生认真阅读题目对现实问题的描述，经过分类、聚集、概括等方法，从中确定实体及其联系。题目已经给出了 4 个实体，需要根据需求描述，给出实体间的联系。根据题意，“每个超市有一位经理”并且部门经理也是超市的员工，可以得出超市与经理之间的管理联系类型为 1:1 。又由于“每个部门有一名部门经理”并且部门经理也是超市的员工，可以得出部门与部门经理之间的负责联系类型为 1:1 。

由“每个部门有多名员工，每个员工属于一个部门”可以得出部门与员工间的所属联系类

型为 1:*;并且员工是经理的超类型，经理是员工的子类型。

由“一名业务员可以负责超市内多种商品配置，一种商品可以由多名业务员配置”，可以得出，业务员与商品之间的配置联系类型为*:*。

完整的 ER 图如下：

答案

(1) (a)超市名称，部门名称

(b)员工号，超市名称，部门名称

(c)商品号

(2) 部门关系模式的主键：超市名称，部门名称

外键：部门经理，超市名称

配给关系模式的主键：商品号，配给时间，业务员

外键：商品号，业务员

【试题分析】

(1) 完整的关系模式如下：

超市(超市名称，经理，地址，电话)

部门(超市名称，部门名称，部门经理，联系电话)

员工(员工号，超市名称，部门名称，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)

配置(商品号，配置时间，配置数量，业务员)

(2) 部门和配置关系模式的主键和外键的分析如下：

在部门关系模式中，由于每个超市都设有计划部、财务部、销售部等多个部门，因此为了区分部门关系的每一个元组，需要“超市名称、部门名称”作为部门的主键。又因为部门经理也是员工，因此部门经理为员工关系的外键。

在配置关系模式中，“商品号，配置时间，业务员”唯一标识配置关系的每一个元组，故为配置关系的主键，外键为商品号，业务员。

答案

(1) 该属性属于复合属性，因为简单属性是原子的、不可再分的。

(2) (d) 1:n

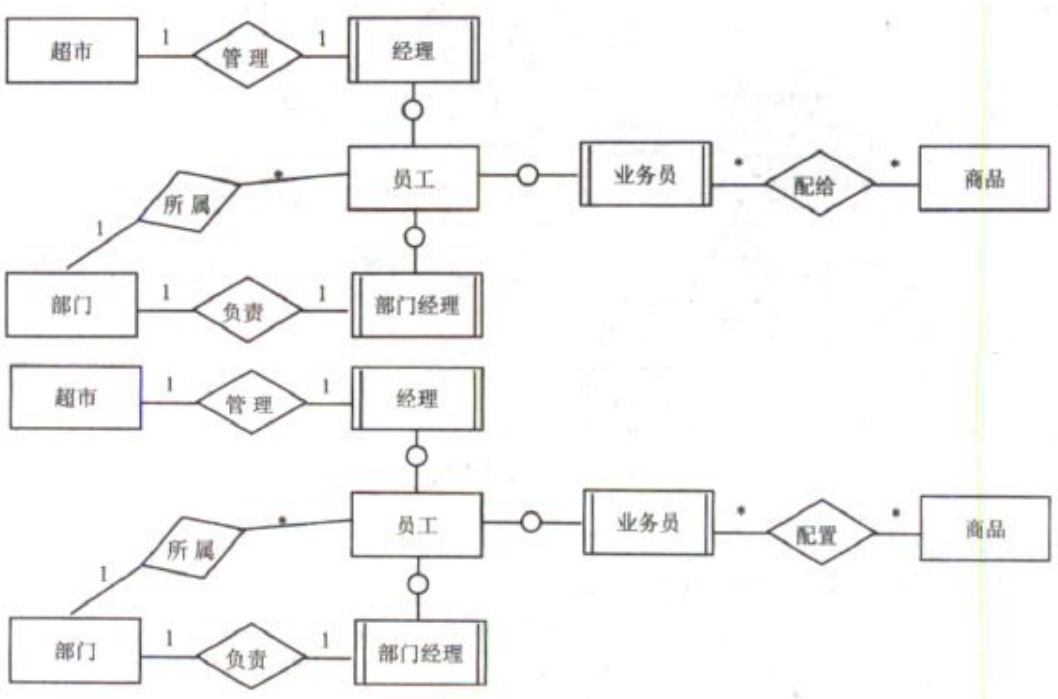
(e) 超市名称，经理，电话

【试题分析】

(1) 超市的地址属性不属于简单属性。因为根据题意，超市关系的地址可以进一步分为邮编、省、市、街道，而简单属性是原子的、不可再分的，复合属性可以细分为更小的部分(即划分为别的属性)。本小题的超市的地址可以进一步分为邮编、省、市、街 道，故属于

复合属性。

(2）因为根据题意，超市需要增设一位经理的职位，那么超市与经理之间的联系类型应修改为 1:n，超市的主键应修改为超市名称，经理，电话。



试题三 答案： 解析：

答案

C1： AddressC2： RidingC3： IneligibleC4： Eligible

【试题分析】

本题属于经典的考题，主要考查面向对象分析方法与设计的基本概念与应用。在建模方面，本题中只涉及到了 UML 类图。类图上的考点也是比较常规的对类的识别以及多重度的确定，题目难度不大。

根据【说明】中的“(1) 每个人(Person)可以是一个合法选民(Eligible)或者无效的选民(Ineligible) ”，可以推断出这里有一个“一般/特殊”关系，应采用继承结构。 再对照类图， C3 、 C4 处显而易见应该是 Ineligible 和 Eligible 。由于 C4 和 C2 之间的关联关系，这里 C3 和 C4 的答案是不能互换的。

根据【说明】中的“(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(Town)或者城市(City) ”，可以推断出 C1 、 City 、 Town 这 3 个类描述的是与地址相关的内容，因此 C1 处应该是 Address 。对应地， C2 处应该是 Riding ，这个由 C2 与 City 、 C2 与 Town 之间的联系名称“district”也能推断出来。

答案

M1： 1， M2：*或 0..*， M3：*， M4： 1， M5：*， M6： 1。

【试题分析】

对于联系的多重度的判定，应注意题目中关于不同概念之间关联数量的描述。

M1、M2 这一对多重度，刻画的是“Person”和“Address”之间的关系。由【说明】中的“假设每个人只有一个地址”，可以得出 M1 和 M2 处分别为 1 和*。

M3 和 M4 描述的是合法选民与选区之间的关系。由【说明】中的“每个合法选民仅能注册一个选区”，可知 M3 和 M4 分别为*和 1。

M5 和 M6 描述的是选区和地址之间的关系。在【说明】中假设，每个合法选民在选区中只注册一个地址，因为 M5 和 M6 处分别为*和 1。

答案

(1) M1 处改为 1..*，在 Registration 类中增加 address 属性，指明注册时使用的是哪个地址。

(2) 增加一个类“主要居住地”，作为类 Address 的子类；类 Person 与类“主要居住地”之间具有关系联系，且每个人只有一个主要居住地。

【试题分析】

本问题考查当需求发生变化时，对设计模型的修改。这里给出了两个需求变更，分别对初始类图进行修改。

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。由于选区由住址确定，能够在多个选区注册，意味着其居住地址不止一个。所以“Person”和“Address”之间的多重度会发生变化。在选区注册时所使用的地址也不唯一了，因此需要增加属性来记录在注册选区时所使用的地址，从而对 C2 和 C4 之间的关联类进行修改，增加其属性。

需求 2:对于满足需求 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。这个需求对选民的地址信息提出了更为详细的需求，按照面向对象方法将“不变部分和可变部分分离”的思想，在类图中增加一个新的类，并采用继承机制继承原有 Address 类中的共性元素。

试题四 答案： 解析：

答案

(1) b[0]=1

(2) j

(3) a[j]<=a[i]

(4)b[i]=len+1

【试题分析】

本题考查算法设计与分析以及用 C 程序设计语言来实现算法的能力。

此类题目要求考生认真阅读题目对问题的描述，理解算法思想，并会用 C 程序设计语言来实现。

根据题干描述，用一个数组 *b* 来记录数组 *a* 每个子数组的最长递增子序列的长度，即 *b[i]* 记录 *a[0..i]*的最长递增子序列的长度。首先，只有一个元素的数组的最长递增子序列的长度为 *1*，即给 *b[0]*直接赋值 *1*。因此，空(*1*）处填写“*b[0] = 1*”。两重 *for* 循环中，第一重是从 *a* 数组的第二个元素开始，考虑每个子数组 *a[0..i]*的最长递增子序列的长度，第二重是具体的计算过程。考虑子数组 *a[0..i]*，其最长递增子序列的长度应该等于子数组 *a[0..i-1]*中的比元素 *a[i]*小的元素的最长递增子序列的长度加 *1*，当然，可能存在多个元素比元素 *a[i]*小，那么存在多个最长递增子序列的长度，此时，取最大者。因此，空处填写“*j*

来实现的。另外，计算的过程不是采用递归的方式，而是以一种自底向上的方式进行的。

答案

(5) 动态规划法

(6) $O(n^2)$

【试题分析】

从题干说明和 C 程序来看，这是一个最优化问题，而且问题具有最优子结构，一个序列的最长递增子序列由其子序列的最长递增子序列构成。在计算过程中采用了自底向上的方式来进行，这具有典型的动态规划特征。因此采用的是动态规划设计策略。

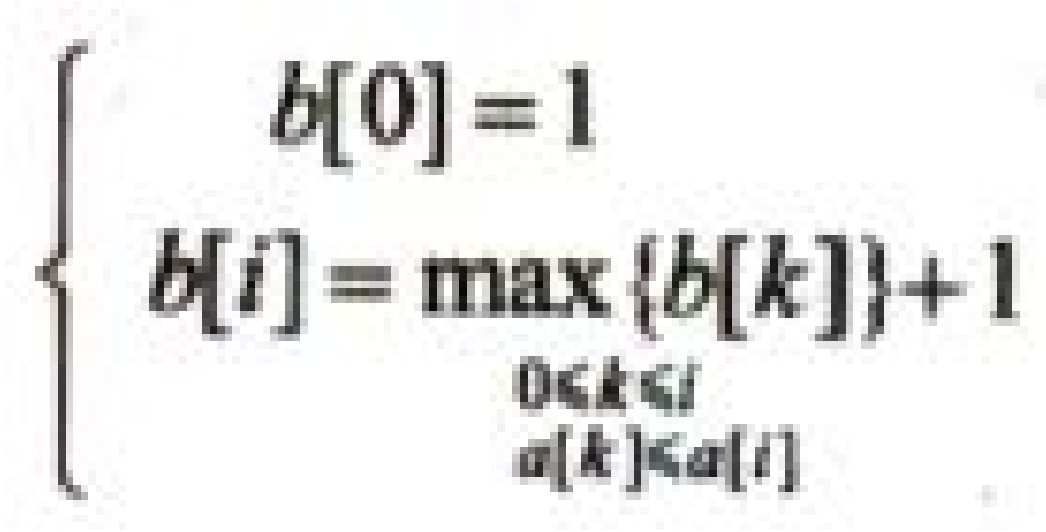
C 程序中，有两重 *for* 循环，因此时间复杂度为 $O(n^2)$。

答案

b={1,2,2,3,3,4}

【试题分析】

输入数组为数组 *a = {3,10,5, 15,6, 8}*，很容易得到，子数组 *a[0..0], a[0..1],…, a[0..5]* 的最长递增子序列的长度分别为 *1,2,2,3,3,4*, 因此答案为 *b={1,2,2,3,3,4}*。该题可以根据题干说明、C 代码来计算。由于输入很简单，答案也可以从输入直接计算出来。



试题五 答案：　解析：

答案

(1) virtual void execute() =0

(2) light->on()

(3) light->off()

(4) onCommands[slot]

(5) offCommands[slot]

(6) onCommands[slot]->execute()

(7) offCommands[slot]->execute()

【试题分析】

本题考查命令(Command)模式的基本概念和应用。

命令模式把一个请求或者操作封装到一个对象中。命令模式允许系统使用不同的请求把客户端参数化，对请求排队或者记录请求日志，可以提供命令的撤销和恢复功能。

在软件系统中，行为请求者与行为实现者之间通常呈现一种紧耦合的关系。但在某些场合，比如要对行为进行记录撤销重做事务等处理，这种无法抵御变化的紧耦合是不合适的。这种情况下，使用 Command 模式将行为请求者与行为实现者进行解耦。

题目中给出了 Command 模式的类图，其中：

Command 类为所有命令声明了一个接口。调用命令对象的 execute() 方法，就可以让接收者进行相关的动作。

ConcreteCommand 类定义了动作和接收者之间的绑定关系。调用者只要调用 execute() 就可以发出请求，然后由 ConcreteCommand 调用接收者的一个或多个动作。

Invoker 持有一个命令对象，并在某个时间点调用命令对象的 execute() 方法，将请求付诸实行。

Receiver 知道如何进行必要的工作，实现这个请求。任何类都可以当接收者。

了解了 Command 模式的内涵之后，下面来看程序。

由于 Command 类的主要作用是为所有的 ConcreteCommand 定义统一的接口，在 C++ 中通常采用抽象类来实现。 C++的抽象类是至少具有一个纯虚拟函数的类。本题中的接口就是 execute() 方法，所以(1) 处要求填写的是纯虚拟函数 execute 的定义方式，即 virtual voidexecute() = 0 。

类 LightOnCommand 、 LightOffCommand 对应的就是模式中的 ConcreteCommand 。

ConcreteCommand 中 execute() 方法的代码在类图中已经给出，现在需要确定 receiver 是谁。类 Light 充当的是 Receiver ，其中定义了两种 action: on 和 off 。所以(2) 、(3) 对应代码分别为 light->on() 、 light->off() 。

类 RemoteControl 对应的是模式中的 Invoker ，在该类中设置需要控制的命令对象。(4) 处对应的代码为 onCommandstslot]，设置“开灯”命令对象：(5) 处对应的代码为 offCommands[slot], 设置‘关灯”命令对象。类 RemoteControl 中的方法 onButtonWasPushed 和 offButtonWasPushed ，分别完成对开灯、关灯命令对象的 execute 方法的调用，所以(6) 、(7) 处分别对应代码 onCommands[slot]->execute() 、 offCommands[slot]->execute() 。

试题六 答案： 解析：

答案

(1) interface Command

(2) light.on()

(3) light.off()

(4) onCommands[slot]

(5) offCommands[slot]

(6) onCommands[slot].execute()

(7) offCommands[slot].execute()

【试题分析】

本题考查命令(Command)模式的基本概念和应用。

命令模式把一个请求或者操作封装到一个对象中。命令模式允许系统使用不同的请求把客户端参数化，对请求排队或者记录请求日志，可以提供命令的撤销和恢复功能。

在软件系统中，行为请求者与行为实现者之间通常呈现一种紧耦合的关系。但在某些场合，比如要对行为进行记录撤销重做事务等处理，这种无法抵御变化的紧耦合是不合适的。这种情况下，使用 command 模式将行为请求者与行为实现者进行解耦。

题目中给出了 Command 模式的类图，其中：

Command 类为所有命令声明了一个接口。调用命令对象的 execute() 方法，就可以让接收者进行相关的动作。

ConcreteCommand 类定义了动作和接收者之间的绑定关系。调用者只要调用 execute() 就可以发出请求，然后由 ConcreteCommand 调用接收者的一个或多个动作。

Invoker 持有一个命令对象，并在某个时间点调用命令对象的 execute() 方法，将请求付诸实行。

Receiver 知道如何进行必要的工作，实现这个请求。任何类都可以当接收者。

了解了 Command 模式的内涵之后，下面来看程序。

由于 Command 类的主要作用是为所有的 ConcreteCommand 定义统一的接口，在 Java 中通常采用接口(Interface)来实现，所以(1) 处对应的代码为 interfaceCommand 。

类 LightOnCommand 、 LightOffCommand 对应的就是模式中的 ConcreteCommand 。

ConcreteCommand 中 execute() 方法的代码在类图中已经给出，现在需要确定 receiver 是谁。类 Light 充当的是 Receiver ，其中定义了两种 action: on 和 off 。所以(2) 、(3) 对应代码分别为 light.on() 和 light.off() 。

类 RemoteControl 对应的是模式中的 Invoker ，在该类中设置需要控制的命令对象。(4) 处对应的代码为 onCommands[slot]，设置“开灯”命令对象；(5) 处对应的代码为 offPCommands[slot],设置″‘关灯”命令对象。类 RemoteControl 中的方法 onButtonWasPushed 和 offButtonWasPushed ，分别完成对开灯、关灯命令对象的 execute 方法的调用。所以(6) 、(7) 处分别对应代码 onCommands[slot].execute() 、offCommands[slot],execute() 。