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 2009 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 假设某大型商业企业由商品配送中心和连锁超市组成，其中商品配送中心包括采购、财务、配送等部门。为实现高效管理，设计了商品配送中心信息管理系统，其主要 功能描述如下：

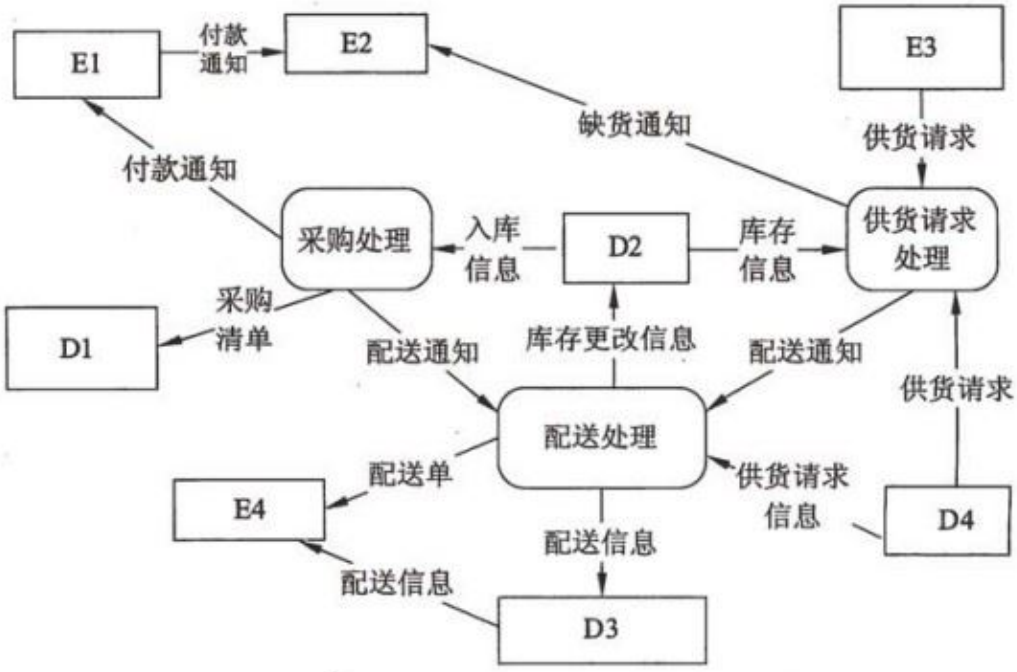
1. 系统接收由连锁超市提出的供货请求，并将其记录到供货请求记录文件。

2. 在接到供货请求后，从商品库存记录文件中进行商品库存信息查询。如果库存满足供货请求，则给配送处理发送配送通知：否则，向采购部门发出缺货通知。

3. 配送处理接到配送通知后，查询供货请求记录文件，更新商品库存记录文件，并向配送部门发送配送单，在配送货品的同时记录配送信息至商品配送记录文件。

4. 采购部门接到缺货通知后，与供货商洽谈，进行商品采购处理，合格商品入库，并记录采购清单至采购清单记录文件、向配送处理发出配送通知，同时通知财务部门给供货商支付货款。

该系统采用结构化方法进行开发，得到待修改的数据流图如下图所示。



数据流图

问题：1.1

使用【说明】中的词语，给出上图中外部实体 E1 至 E4 的名称和数据存储 D1 至 D4 的名称。

问题：1.2

以上数据流图中存在四处错误数据流，请指出各自的起点和终点；若将上述四条错 误数据流删除，为保证数据流图的正确性，应补充三条数据流，请给出所补充数据流的 起点和终点。(起点和终点请采用上述数据流图中的符号或名称)

错误数据流

起点	终点

补充的数据流

起点	终点

试题二　某集团公司拥有多个大型连锁商场，公司需要构建一个数据库系统以方便管理其业 务运作活动。

【需求分析结果】

1. 商场需要记录的信息包括商场编号(编号唯一)，商场名称，地址和联系电话。 某商场信息如下表所本。

商场信息表

2. 每个商场包含有不同的部门，部门需要记录的信息包括部门编号(集团公司分配)，部门名称，位置分布和联系电话。某商场的部门信息如下表所示。

部门信息表

3. 每个部门雇用多名员工处理日常事务，每名员工只能隶属于一个部门(新进员工在培训期不隶属于任何部门)。员工需要记录的信息包括员工编号(集团公司分配)， 姓名，岗位，电话号码和工资。员工信息如下表所示。

员工信息表

4. 每个部门的员工中有一名是经理，每个经理只能管理一个部门，系统需要记录每个经理的任职时间。

【概念模型设计】

实体联系图

【关系模式设计】

商场(商场编号，商场名称，地址，联系电话)

部门(部门编号，部门名称，位置分布，联系电话，(a))

员工：（员工编号，员工姓名，岗位，电话号码，工资，(b))

经理((c),任职时间)

商场编号	商场名称	地址	联系电话
PS2101	淮海商场	淮海中路 918 号	021-64158818
PS2902	西大街商场	西大街时代盛典大厦	029-87283220
PS2903	东大街商场	碑林区东大街 239 号	029-87450287
PS2901	长安商场	雁塔区长安中路 38 号	029-85264953

部门编号	部门名称	位置分布	联系电话
DT002	财务部	商场大楼六层	82504342
DT007	后勤部	商场地下副一层	82504347
DT021	安保部	商场地下副一层	82504358
DT005	人事部	商场大楼六层	82504446
DT001	管理部	商场裙楼三层	82504668

员工编号	姓名	岗位	电话号码	工资
XA3310	周　超	理货员	13609257638	1500.00
SH1075	刘　飞	防损员	13477293487	1500.00
XA0048	江雪花	广播员	15234567893	1428.00
BJ3123	张正华	部门主管	13345698432	1876.00

问题： 2.1

根据问题描述，补充四个联系，竞善图 2-1 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 、 l:n 和 m:n 。

问题： 2.2

根据实体联系图，将关系模式中的空(a) （ c)补充完整，并分别给出部门、员工和经理关系模式的主键和外键。

问题： 2.3

为了使商场有紧急事务时能联系到轮休的员工，要求每位员工必须且只能登记一位紧急联

系人的姓名和联系电话，不同的员工可以登记相同的紧急联系人。则在图 2-1 中 还需添加的实体是(1),该实体和图 2-1 中的员工存在(2)联系(填写联系类型)。 给出该实体的关系模式。

试题三 某银行计划开发一个自动存提款机模拟系统(ATMSystem).系统通过读卡器 (CardReader)读取 ATM 卡；系统与客户(Customer)的交互由客户控制台(Customer- Console)实现;银行操作员(Operator)可控制系统的启动(SystemStartup)和停止(SystemShutdown)；系统通过网络和银行系统(Bank)实现通信。

当读卡器判断用户已将 ATM 卡插入后，创建会话(Session)。会话开始后，读卡器 进行读卡，并要求客户输入个人验证码(PIN)。系统将卡号和个人验证码信息送到银行系统进行验证。验证通过后，客户可从菜单选择如下事务(Transaction):

1. 从 ATM 卡账户取款(Withdraw);

2. 向 ATM 卡账户存款(Deposit):

3. 进行转账(Transfer);

4. 查询(Inquire) ATM 卡账户信息。

一次会话可以包含多个事务，每个事务处理也会将卡号和个人验证码信息送到银行系统进行验证。若个人验证码错误，则转个人验证码错误处理(InvalidPINProcess)。每个事务完成后，客户可选择继续上述事务或退卡。选择退卡时，系统弹出 ATM 卡，会话结束。

系统采用面向对象方法开发，使用 UML 进行建模。系统的顶层用例图如图 3-1 所示，一次会话的序列图(不考虑验证)如图 3-2 所示。

问题：3.1
根据【说明】中的描述，给出图 3-1 中 A1 和 A2 所对应的参与者， U1 至 U3 所对应的用例，以及该图中空(1)所对应的关系。(U1 至 U3 的可选用例包括： Session 、 Transaction, InsertCard 、 InvalidPINProcess 和 Transfer)
问题：3.2
根据【说明】中的描述，使用消息名称列表中的英文名称，给出图 3-2 中 6 9 对应的消息。

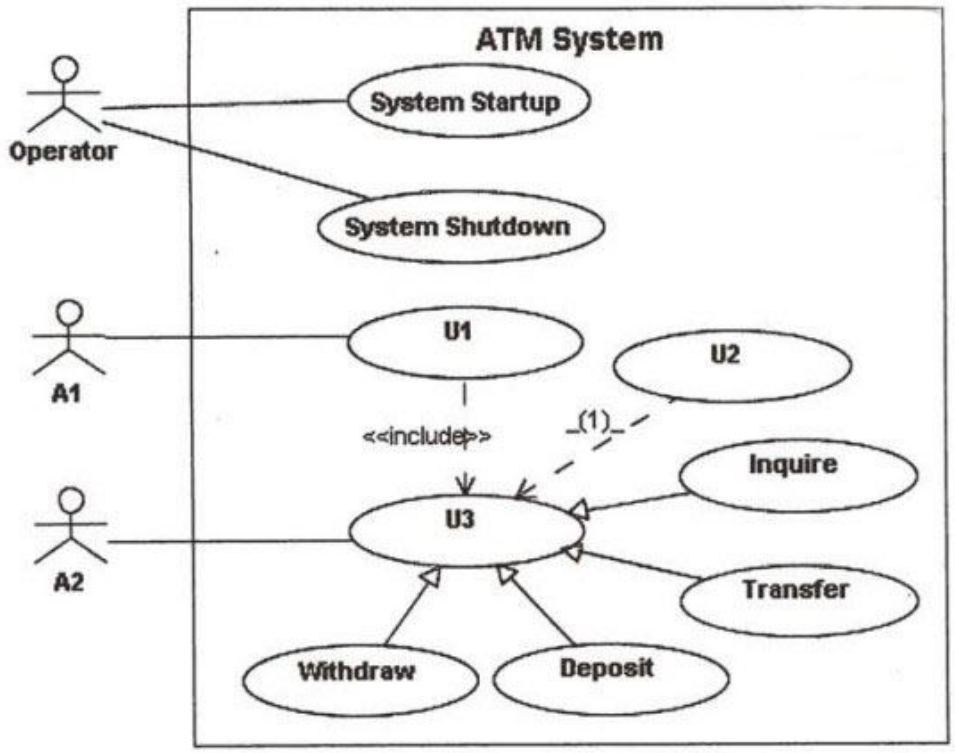


图 3-1　ATM 系统顶层用例图

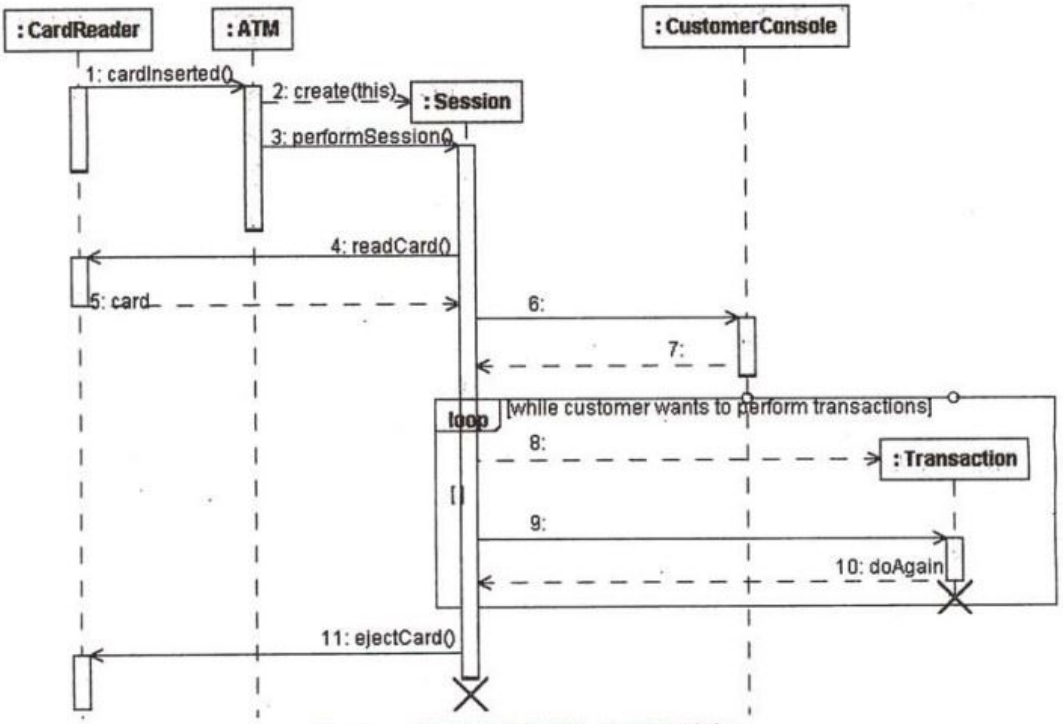


图 3-2　一次会话的序列图（无验证消息）

可能的消息名称列表

名　称	说　明	名　称	说　明
cardInserted()	ATM 卡已插入	performTransaction()	执行事务
performSession()	执行会话	readCard()	读卡
readPIN()	读取个人验证码	PIN	个人验证码信息
creat(atm, this, card, pin)	为当前会话创建事务	create(this)	为当前 ATM 创建会话
card	ATM 卡信息	doAgain	执行下一个事务
ejectCard()	弹出 ATM 卡		

问题：3.3

解释图 3-1 中用例 U3 和用例 Withdraw 、 Deposit 等四个用例之间的关系及其内涵。

试题四　现需在某城市中选择一个社区建一个大型超市，使该城市的其他社区到该超市的距离总和最小。用图模型表示该城市的地图，其中顶点表示社区，边表示社区间的路线，边上的权重表示该路线的长度。

现设计一个算法来找到该大型超市的最佳位置：即在给定图中选择一个顶点，使该顶点到其他各顶点的最短路径之和最小。算法首先需要求出每个顶点到其他任一顶点的最短路径，即需要计算任意两个顶点之间的最短路径；然后对每个顶点，计算其他各顶点到该顶点的最短路径之和；最后，选择最短路径之和最小的顶点作为建大型超市的最佳位置。

问题：4.1

本题采用 Floyd-Warshall 算法求解任意两个顶点之间的最短路径。已知图 G 的顶点集合为 $V=\{1,2,\cdots,n\}$，$W=\{w_{ij}\}_{n*n}$ 为权重矩阵。设 $d_{ij}^{(k)}$ 为从顶点 i 到顶点 j 的一条最短路径的权重。当 $k=0$ 时，不存在中间顶点，因此 $d_{ij}^{(0)}=w_{ij}$；当 $k>0$ 时，该最短路径上所有的中间顶点均属于集合 $\{1,2,\cdots,k\}$。若中间顶点包括顶点 k，则 $d_{ij}^{(k)}=d_{ik}^{(k-1)}+d_{kj}^{(k-1)}$；若中间顶点不包括顶点 k，则 $d_{ij}^{(k)}=d_{ij}^{(k-1)}$。于是得到如下递归式。

$$d_{ij}^{(k)}=\begin{cases} w_{ij} & k=0 \\ \min(d_{ij}^{(k-1)},d_{ik}^{(k-1)}+d_{kj}^{(k-1)}) & k>0 \end{cases}$$

因为对于任意路径，所有的中间顶点都在集合 $\{1,2,\cdots,n\}$ 内，因此矩阵 $D^{(n)}=\{d_{ij}^{(n)}\}_{n*n}$ 给出了任意两个顶点之间的最短路径，即对所有 $i,j\in V$，$d_{ij}^{(n)}$ 表示顶点 i 到顶点 j 的最短路径。

```
LOCATE  -SHOPPINGMALL(W, n)
1    D(0) = W
2    for ____(1)____
3        for i = 1 to n
4            for j = 1 to n
5            if d_ij^(k-1) ≤ d_ik^(k-1) + d_kj^(k-1)
6                        ____(2)____
7                else
8                        ____(3)____
9    for i = 1 to n
10       SP[i] = 0
11       for j = 1 to n
12               ____(4)____
13   min_SP = SP[1]
14   ____(5)____
15   for i = 2 to n
16       if min_SP > SP[i]
17           min_SP = SP[i]
18           min_v = i
19   return ____(6)____
```

问题： 4.2

【问题 1 】中伪代码的时间复杂度为(7)(用 0 符号表示)。

试题五 阅读下列说明和 C 函数代码，将应填入(n)处的字句写在答题纸的对应栏内。

【说明】

对二叉树进行遍历是二叉树的一个基本运算。遍历是指按某种策略访问二叉树的每个节点，且每个节点仅访问一次的过程。函数 InOrder()借助栈实现二叉树的非递归中序遍历运算。

设二叉树采用二叉链表存储，节点类型定义如下：

```
typedef struct BtNode{
   ElemType  data;                  /*节点的数据域，ElemType 的具体定义省略*/
   struct BtNode *lchild,*rchild;   /*节点的左、右孩子指针域*/
}BtNode, *BTree;
```

在函数 InOrder()中，用栈暂存二叉树中各个节点的指针，并将栈表示为不含头节点的单向链表（简称链栈），其节点类型定义如下：

```
typedef struct StNode{              /*链栈的节点类型*/
     BTree elem;                    /*栈中的元素是指向二叉链表节点的指针*/
     struct StNode *link;
}StNode;
```

假设从栈顶到栈底的元素为 e_n、e_{n-1}、…、e_1，则不含头节点的链栈示意图如图 5-1 所示。

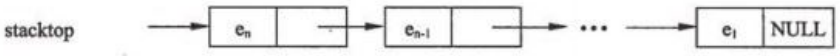


图 5-1 链栈示意图

问题： 5.1

【C 函数】

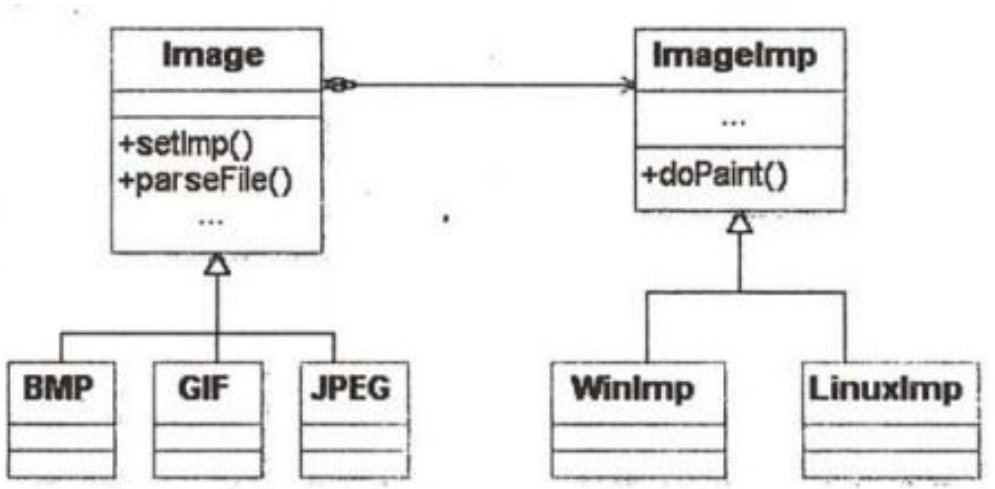
```
int  InOrder(BTree root)          /*实现二叉树的非递归中序遍历*/
{
    BTree ptr;                    /*ptr 用于指向二叉树中的节点*/
    StNode *q;                    /*q 暂存链栈中新创建或待删除的节点指针*/
    StNode *stacktop = NULL;      /*初始化空栈的栈顶指针 stacktop*/
    ptr = root;                   /*ptr 指向二叉树的根节点*/
    while ( __(1)__ || stacktop != NULL) {
    while (ptr != NULL) {
       q = (StNode *)malloc(sizeof(StNode));
       if (q == NULL)
return -1;
  q->elem = ptr;
   __(2)__;
    stacktop = q;             /*stacktop 指向新的栈顶*/
    ptr = __(3)__;            /*进入左子树*/
}
    q = stacktop;
   __(4)__;                   /*栈顶元素出栈*/
    visit(q);                 /*visit 是访问节点的函数，其具体定义省略*/
    ptr = __(5)__;            /*进入右子树*/
    free(q);                  /*释放原栈顶元素的节点空间*/
}
return 0;
}/*InOrder*/
```

试题六 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】

现欲实现一个图像浏览系统，要求该系统能够显示 BMP 、 JPEG 和 GIF 三种格式的文件，并且能够在 Windows 和 Linux 两种操作系统上运行。系统首先将 BMP 、 JPEG 和 GIF 三种格式的文件解析为像素矩阵，然后将像素矩阵显示在屏幕上。系统需具有较好的扩展性以支持新的文件格式和操作系统。为满足上述需求并减少所需生成的子类数目，采用桥接(Bridge)设计模式进行设计，所得类图如下图所示。

采用该设计模式的原因在于：系统解析 BMP 、 GIF 与 JPEG 文件的代码仅与文件格式相关，而在屏幕上显示像素矩阵的代码则仅与操作系统相关。



类图

问题： 6.1

现假设该系统需要支持 10 种格式的图像文件和 5 种操作系统，不考虑类 Matrix，若采用桥接设计樽式则至少需要设计(7) 个类。

【C++代码】

```
class Matrix{    //各种格式的文件最终都被转化为像素矩阵
    //此处代码省略
};
class ImageImp{
public:
    virtual void doPaint(Matrix m) = 0;  //显示像素矩阵 m
};

class WinImp : public ImageImp{
public:
    void doPaint(Matrix m){ /*调用 Windows 系统的绘制函数绘制像素矩阵*/ }
};
```

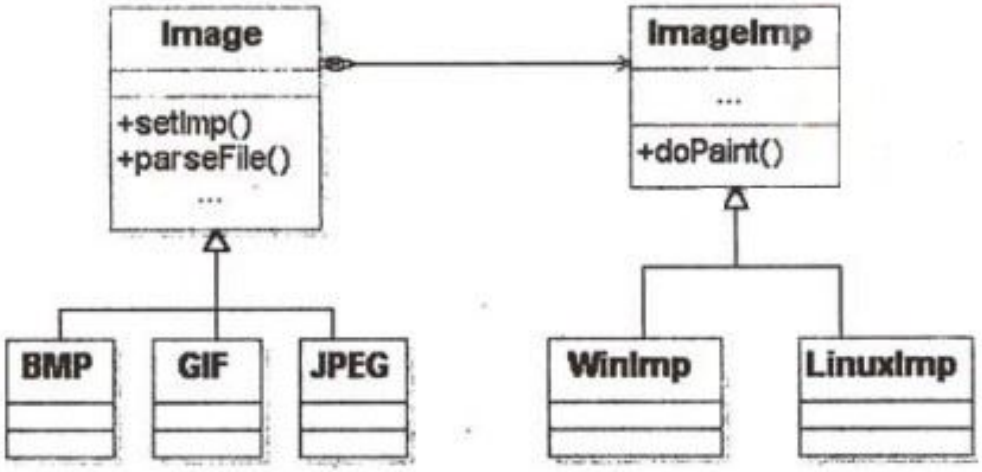
```
class LinuxImp : public ImageImp{
public:
    void doPaint(Matrix m){ /*调用 Linux 系统的绘制函数绘制像素矩阵*/ }
};

class Image {
public:
    void setImp(ImageImp *imp){ (1) = imp;}
    virtual void parseFile(string fileName) = 0;
protected:
    (2) *imp;
};

class  BMP : public Image{
public:
    void parseFile(string fileName){
        //此处解析 BMP 文件并获得一个像素矩阵对象 m
        (3) ;// 显示像素矩阵 m

    }
};
class  GIF : public Image{
    //此处代码省略
};

class  JPEG : public Image{
    //此处代码省略
};

void main(){
    //在 Windows 操作系统上查看 demo.bmp 图像文件
    Image *image1 = (4) ;
    ImageImp *imageImp1 = (5) ;
    (6) ;

        image1->parseFile("demo.bmp");

}
```

试题七 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】

现欲实现一个图像浏览系统，要求该系统能够显示 BMP 、 JPEG 和 GIF 三种格式的文件，并且能够在 Windows 和 Linux 两种操作系统上运行。系统首先将 BMP 、 JPEG 和 GIF 三种格式的文件解析为像素矩阵，然后将像素矩阵显示在屏幕上。系统需具有较好的扩展性以支持新的文件格式和操作系统。为满足上述需求并减少所需生成的子类数目，采用桥接(Bridge)设计模式进行设计，所得类图如下图所示。

采用该设计模式的原因在于：系统解析 BMP 、 GIF 与 JPEG 文件的代码仅与文件格式相关，而在屏幕上显示像素矩阵的代码则仅与操作系统相关。



类图

问题： 7.1

【Java 代码】

```
class Matrix{    //各种格式的文件最终都被转化为像素矩阵
    //此处代码省略
};
abstract class ImageImp{
    public abstract void doPaint(Matrix m);  //显示像素矩阵 m
};

class WinImp extends ImageImp{
    public void doPaint(Matrix m){       /*调用 Windows 系统的绘制函数绘制像
素矩阵*/ }
};

class LinuxImp extends ImageImp{
    public void doPaint(Matrix m){/*调用 Linux 系统的绘制函数绘制像素矩阵*/}
};

abstract class Image {
    public void setImp(ImageImp imp){
        __(1)__ = imp; }
    public abstract void parseFile(String fileName);
    protected __(2)__ imp;
};
class  BMP extends Image{
    public void parseFile(String fileName){
            //此处解析 BMP 文件并获得一个像素矩阵对象 m
            __(3)__;// 显示像素矩阵 m

        }
};
class  GIF extends Image{
//此处代码省略
};

class  JPEG extends Image{
    //此处代码省略
};

public class javaMain{
    public static void main(String[] args){
        //在 Windows 操作系统上查看 demo.bmp 图像文件
```

```
        Image image1 = __(4)__;
        ImageImp imageImp1 = __(5)__;
        __(6)__;
        image1.parseFile("demo.bmp");
    }
}
```

试题一 答案：　解析：　E1：财务部门 E2：采购部门

E3：连锁超市 E4：配送部门

D1；采购清单记录文件 D2：商品库存记录文件

D3：商品配送记录文件 D4：供货请求记录文件

本题考查 DFD 的分析与设计，问题一主要考查 DFD 中的外部实体和数据存储，由于在题干中已经提到“系统接收由连锁超市提出的供货请求，并将其记录到供货请求记录文件”，因此可以明确出“连锁超市”外部实体和“供货请求记录文件”数据存储；对应到 DFD 图中为 E3 和 D4。描述中的第二项提出“从商品库存记录文件中进行商品库 存信息查询。如果库存满足供货请求，则给配送处发送配送通知；否则，向采购部门发出缺货通知”，因为配送通知需要发送到采购部门，因此采购部门将成为系统的外部实体； 同时，商品库存记录文件能够提供库存信息，所以 DFD 图中 E2 和 D2 分别为采购部门和商品配送记录文件。第三项需求“配送处理接到配送通知后，查询供货请求记录文件，更新商品库存记录文件，并向配送部门发送配送单，在配送货品的同时记录配送信息至 商品配送记录文件”，所以配送处理需要查询供货请求记录文件，更新商品库存记录文件与商品配送记录文件，因此 D3 为商品配送记录文件；采购处理需要记录采购清单同时通知财务部门，所以 E1 应该为财务部门， D1 为采购清单记录文件，剩下的 E4 则为配送部门。

DFD 中出现的错误数据流为： E1 到 E2，E1 与 E2 的数据流不属于系统的范围； D3 到 E4，多余的数据流； D2 到采购处理，数据流方向错误； D4 到供货请求处理，数据流方向错误。

需要补充的数据流为： E2 到采购处理，因为 E2 是采购部门，采购部门需要给采购处提供入库商品信息；采购处到 D2 需要一条数据流，因为采购处理需要更改库存信息；供货请求处理到 D4 需要一条数据流，因为供货请求处理需要记录供货请求信息。

本题考查 DFD 的分析与设计，问题一主要考查 DFD 中的外部实体和数据存储，由于在题干中已经提到“系统接收由连锁超市提出的供货请求，并将其记录到供货请求记录文件”，

因此可以明确出“连锁超市”外部实体和“供货请求记录文件”数据存储；对应到 DFD 图中为 E3 和 D4 。描述中的第二项提出“从商品库存记录文件中进行商品库 存信息查询。如果库存满足供货请求，则给配送处发送配送通知；否则，向采购部门发出缺货通知”，因为配送通知需要发送到采购部门，因此采购部门将成为系统的外部实体； 同时，商品库存记录文件能够提供库存信息，所以 DFD 图中 E2 和 D2 分别为采购部门和商品配送记录文件。第三项需求“配送处理接到配送通知后，查询供货请求记录文件，更新商品库存记录文件，并向配送部门发送配送单，在配送货品的同时记录配送信息至 商品配送记录文件”，所以配送处理需要查询供货请求记录文件，更新商品库存记录文件与商品配送记录文件，因此 D3 为商品配送记录文件；采购处理需要记录采购清单同时通知财务部门，所以 E1 应该为财务部门， D1 为采购清单记录文件，剩下的 E4 则为配送部门。 DFD 中出现的错误数据流为： E1 到 E2，E1 与 E2 的数据流不属于系统的范围； D3 到 E4, 多余的数据流；D2 到采购处理，数据流方向错误； D4 到供货请求处理，数据流方向错误。需要补充的数据流为： E2 到采购处理，因为 E2 是采购部门，采购部门需要给采购处提供入库商品信息；采购处到 D2 需要一条数据流，因为采购处理需要更改库存信息；供货请求处理到 D4 需要一条数据流，因为供货请求处理需要记录供货请求信息。

错误数据流

起点	终点
E1	E2
D3	E4
D2	采购处理
D4	供货请求处理

补充的数据流

起点	终点
E2	采购处理
采购处理	D2
供货请求处理	D4

试题二 答案： 解析： （图中的 m 、 n 也可用*表示，对联系名称可不做要求，但不能出现重名）

由“每个商场包含有不同的部门”可知商场与部门间为 1:m 联系；由“每个部门雇用了多名员工处理日常事务”可知部门与员工间为 1:n 联系；由“每个部门的员工中有 一个经

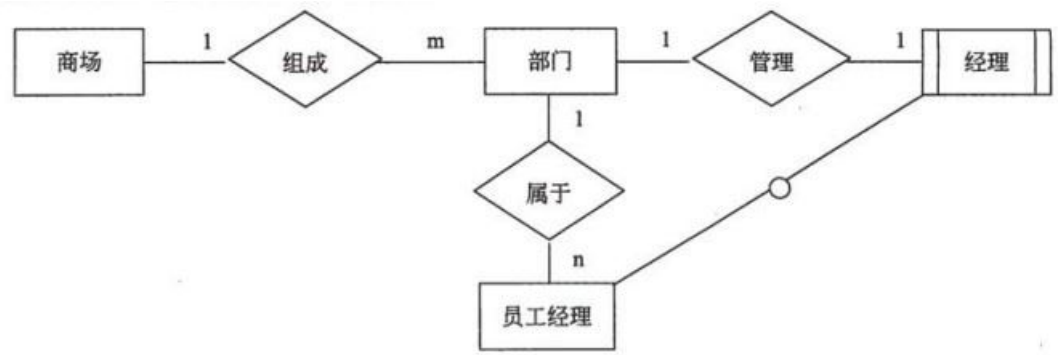
理……每个经理只能管理一个部门”可知部门与经理间为 1:1 联系，并且员工是经理的超类型，经理是员工的子类型。

(a) 商场编号
(b) 部门编号
(c) 员工编号
部门关系模式的主键：部门编号 外键：商场编号
员工关系模式的主键：员工编号 外键：部门编号
经理关系模式的主键：员工编号 外键：员工编号

商场的属性信息中，商场编号由集团公司分配，不会重复，可作为商场的主键属性； 部门的属性信息中，部门编号由集团公司分配，不会重复，可作为部门的主键属性，商场与部门的联系需要通过将商场的主键(商场编号)加入到部门中来表达；员工的属性信息中，员工编号由集团公司分配，不会重复，可作为员工的主键属性，部门与员工的联系需要通过将部门的主键(部门编号)加入到员工中来表达；经理除了包含员工的属性信息外，还需要任职时间属性。完整的关系模式如下：
商场(商场编号，商场名称，地址，联系电话)
部门(部门编号，部门名称，位置分布，联系电话，商场编号)
员工(员工编号，姓名，岗位，电话号码，工资，部门编号)
经理(员工编号，任职时间)

(d) 紧急联系人
(e) 1:n
关系模式：紧急联系人(员工编号，姓名，联系电话)

员工的紧急联系人信息通过添加紧急联系人关系来实现，由“每位员工必须且只能登记一位紧急联系人的姓名和联系电话”，但可能存在多位员工登记同一位家属，可知员工与家属间为 n:1 联系；由“不同员工可以登记相同的紧急联系人”可知，员工编号可作为家属的主键属性。所以需要添加的关系模式如下：
紧急联系人(员工编号，姓名，联系电话)



试题三 答案： 解析： A1： Customer

A2： Bank

U1： Session

U2: Invalid PIN Process

U3： Transaction

(1)： «extend»

构建用例图时，常用的方式是先识别参与者，然后确定用例以及用例之间的关系。识别参与者时，考查和系统交互的人员和外部系统。本题中，与系统交互的人员包括客户(Customer)和银行操作员(Operator),与本模拟系统交互的外部系统包括银行。系统(Bank)。考查用例时，通过判断哪一个特定参与者发起或者触发了与系统的哪些交互，来识别用例并建立和参与者之间的关联。考查用例之间的关系时，《 include>> (包含)定义了用例之间的包含关系，用于一个用例包含另一个用例的行为的建模；如果可以从一个用例的执行中，在需要时转向执行另一个用例，执行完返回之前的用例继续执行，用例 间即存在《 extend 》关系。

本题中，客户一旦插卡成功，系统就创建会话(Session),会话中可以执行用户从菜单选择的 Withdraw、 Deposit 、 Transfer 和 Inquire 等事务(Transaction)。由图中 U3 和 Withdraw 之间的扩展关系，可知 U3 为 Transaction;又由 U1 和 U3 之间的《 include 》关系，得知 U1 为 Session,进而判定图中 A1 为 Customer，A2 为 Bank 。每个事务处理 也会将卡号和个人验证码信息送到银行系统进行验证，若个人验证码错误，则转个人验证码错误处理(InvalidPINProcess ，图中 U2)，所以(1) 处应填《 eitend 》。

6： readPIN()

7： PIN

8： creat(atm, this, card, pin)

9： performTransaction()

序列图是场景的图形化表示，描述了以时间顺序组织的对象之间的交互活动。构造序列图时遵循如下指导原则：确定顺序图的范围，描述这个用例场景或一个步骤；绘制参与者和接口类，如果范围包括这些内容的话；沿左手边列出用例步骤；对控制器类及必须在顺序中协作的每个实体类，基于它拥有的属性或已经分配给它的行为绘制框；为持续类和系统类绘制框；绘制所需消息，并把每条消息指到将实现响应消息的责任的类上；添加活动条指示每个对象实例的生命期；为清晰起见，添加所需的返回消息；如果需要，为循环、可选步骤和替代步骤等添加框架。

本题中，根据说明中的描述，从 ATM 机判断卡已插入(cardlnserted())开始会话， 即为当前 ATM 创建会话(create(this))并开始执行会话(performSession());读卡器读 卡(readCard())获得 ATM 卡信息(card),然后从控制台读取个人验证码输入(readPIN() ，图中标号 6 处)并获得个人验证码信息(PIN，图中标号 7 处)；然后根据用户选择启动并执行事务，即为当前会话创建事务(creat(atm， this， card， pin)，图中标号 8 处)和执行事务(performTransaction() ，图中标号 9 处)；可以选择继续执行某个事务(doAgain) 循环，或者选择退卡(ejectCard())。

Transaction 是一个抽象泛化用例，具有其他事务类型共有的属性和行为，每个具体的事务类型继承它，并实现适合自己的特定的操作。

用例之间的继承关系表示子类型“是一种”父类型。其中父类型通常是一个抽象泛化用例，具有子类型共有的属性和行为，每个具体的子类型继承它，并实现适合自己的特定的操作。

本题中 Transaction 和 Withdraw 、 Deposit 等四个用例之间的关系即为继承关系，Transaction 即是一个抽象泛化用例，具有其他事务类型共有的属性和行为，每个具体的事务类型继承它，并实现适合自己的特定的操作。

试题四 答案： 解析：

(7) O(nsup3)

本问题考查【问题 1 】中的伪代码第 2　8 行，计算任意两点之间的最短路径，有三重循环，故时间复杂度为 0(nsup3)。第 9　12 行，计算每个点到任意其他点的最短路径之和，有两重循环，故时间复杂度为 $0(n^2)$。第 15　18 行，在所有点的最短路径之和中找到最小的最短路径之和，时间复杂度为 $0(n)$。故算法总的时间复杂度为 $0(nsup3)$。

（1）k = 1 to n　（2）$d_{ij}^{(k)} = d_{ij}^{(k-1)}$　（3）$d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

（4）$SP[i] = SP[i] + d_{ij}^{(n)}$　（5）min_v = 1　（6）min_v

本问题考查算法流程。第（1）空表示主循环，k 是循环控制变量，故第（1）空填 k = 1 to n。第（2）和（3）空根据题意和递归式，可分别得到答案为 $d_{ij}^{(k)} = d_{ij}^{(k-1)}$ 和 $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$。计算了任意两个顶点之间的最短路径之后，对每个顶点，开始统计其到所有其他顶点的最短路径之和，因此第（4）空填 $SP[i] = SP[i] + d_{ij}^{(n)}$。第 13 和第 14 行初始化，假设最小的到所有其他顶点的最短路径之和为第一个顶点的最小路径之和，大型超市的最佳位置为第一个顶点，故第（5）空填 min_v = 1。最后要求返回大型超市的最佳位置，即到所有其他顶点的最短路径之和最小的顶点，故第（6）空填 min_v。

试题五 答案：　解析：　(1) ptr != NULL,或 ptr !=0，或 ptr

(2) q->link = stacktop

(3) ptr->1child

(4) stacktop = stacktop->link.或 stacktop = q->link

(5) q->elem->rchild

本题考查基本数据结构和 C 语言程序设计能力。

对非空二叉树进行中序遍历的方法是：先中序遍历根节点的左子树，然后访问根节点，最后中序遍历根节点的右子树。用递归方式描述的算法如下：

从以上算法的执行过程可知，从树根出发进行遍历时，递归调用 In_Order_ Traversing(root->LeftChild)使得遍历过程沿着左孩子分支一直走向下层节点，直到到达二叉树中最左下方的节点(设为 f)的空左子树为止，然后返回 f 节点，再由递归调用 In_Order_ Traversing(root->RightChild)进入 f 的右子树，并重复以上过程。在递归算法执行过程中，辅助实现递归调用和返回处理的控制栈实际上起着保存从根节点到当前节点的路径信息。

用非递归算法实现二叉树的中序遍历时，可以由一个循环语句实现从指定的根节点出发，沿着左孩子分支一直到头(到达一个没有左子树的节点)的处理，从根节点到当前节点的路径信息(节点序列)可以明确构造一个栈来保存。

本题目的难点在于将栈的实现和使用混合在一起来处理，而且栈采用单链表存储结构。下面分析题中给出的代码。

空(1) 是遍历的条件之一，由于另外一个条件 stacktop != NULL 初始时是不成立的， 因此空(1) 所表示的条件必须满足，由于是对非空二叉树进行遍历，显然该条件代表二叉树非空，即 ptr != NULL 或其等价表示形式。

临时指针 ptr 初始时指向整个二叉树的根节点，此后用以下代码表示一直沿左孩子指针链向下走的处理，临时指针 q 用于在链栈中加入新元素时使用。处理思路是：若当前节点有左子树，则将当前节点的指针存入栈中，然后进入当前节点的左子树。入栈时，先申请元素在链栈中的节点空间，然后设置节点数据域的值(即当前节点的指针)，最后将新申请的节点加入链栈首部。

当上述过程进入一棵空的子树时(ptr 为空指针)，循环结束。此后，应该从空的子树返回其父节点并进行访问。由于进入空的左子树前已将其父节点指针压入栈中，因此,栈顶元素即为该父节点，对应的处理就是弹找。相应地，在链栈中要删除表头节点并释放节点空间：

由于还需要通过 q 指针进入被删除节点的右子树，因此，释放节点空间的操作 free(q) 操作之前，使 ptr 指向 q 所指节点的右子树指针，以得到被删除节点的数据域信息，即空(5)所在语句 ptr = q->elem->rchildo

指针是 C 语言中灵活且非常强大的工具，是否熟练掌握 C 语言的判断条件之一就是对指针的理解和使用。软件设计师需要熟练掌握这些内容。

```
void In_Order_Traversing(BiTree root)
{   //root 是指向二叉树根节点的指针
     if (root != NULL) {
          In_Order_Traversing(root->LeftChild);
          visit(root);
          In_Order_Traversing(root->RightChild);
     }
}
```

```
while (ptr != NULL) {
      q = (StNode *)malloc(sizeof(StNode)); /*为新入栈的元素创建节点*/
      if (q == NULL)              /*若创建新节点失败，则退出*/
         return -1;
      q->elem = ptr;              /*在栈顶保存指向当前节点的指针*/
      q->link = stacktop ;        /*新节点加入栈顶*/
stacktop = q;                     /*更新栈顶指针，即 stacktop 指向新的栈顶*/
ptr = ptr->lchild;                /*进入当前节点的左子树*/
}
 q = stacktop;                   /*q 指向链栈中需要删除的节点，即栈顶元素*/
 stack = stacktop->link ;        /*栈顶元素出栈*/
 visit(q);                       /*访问节点*/
 free(q);                        /*释放节点空间*/
```

试题六 答案：　解析： (1) this->imp

(2) Imagelmp

(3) imp->doPaint(m)

(4) newBMP()

(5) new Winlmp()

(6) image 1 ->setlmp(imagelmp1)

(7) 17

根据题目描述，在设计该图像显示系统时主要分为两个步骤：一是读取各种文件并将文件内容转换成像素矩阵，因为各种图片格式不同，因此需要针对每一种图片格式编写文件读取代码，而该代码与操作系统平台无关。将像素矩阵显示到屏幕上时，由于和操作系统相关，因此需要把该代码和读取文件代码相分离。设计中的 Image 类表示抽象的图像概念，Image 类中就包含了读取文件接口和设置实现平台接口； Image 的子类 BMP 、 GIF 和 JPEG

分别负责读取各种不同格式的文件； Imagelmp 的主要任务是将像素矩阵显示在屏幕上，因此，它存在两个子类，分别实现 Windows 系统和 Linux 系统上的图像显示代码。空缺(1)处主要是设置将在哪个平台上进行实现，因此该处应该存储参数所传递的对象，由于该类的成员变量也是 imp，与参数相同，因此需要填写 this->imp;同理，该成员变量的类型和参数的类型应该保持相同，空(2)处应该填写 Imagelmp;空(3)处需要根据 imp 成员变量存储的实现对象来显示图像；在空(4)处需要生成一个 BMP 对象；由于需要在 Windows 平台上实现，因此空(5)处需要生成一个 Winlmp 对象，同时，还需设置该 BMP 对象，应采用 Winlmp 对象来实现显示。采用桥接模式能够将文件分析代码和图像显示代码分解在不同的类层次结构中，如果不考虑中间使用的 Matrix 等类，那么最后需要设计的类包括 2 个父类，对应文件格式子类，对应操作系统平台类，因此 10 种图像格式和 5 种操作系统需要 17 个类。

试题七 答案： **解析：** (1) this.imp

(2) Imagelmp

(3) imp.doPaint(m)

(4) newBMP()

(5) new Winlmp()

(6) image1 .setlmp(imagelmp1)

根据题目描述，在设计该图像显示系统时主要分为两个步骤：一是读取各种文件并将文件内容转换为像素矩阵，因为各种图片格式不同，因此需要计对每一种图片格式编 写文件读取代码，而该代码与操作系统平台无关。将像素矩阵显示到屏幕上时，由于和 操作系统相关，因此需要把该代码和读取文件代码相分离。设计中的 Image 类表示抽象的图像概念，Image 类中就包含了读取文件接口和设置实现平台接口； Image 的子类 BMP 、 GIF 和 JPEG 分别负责读取各种不同格式的文件； Imagelmp 的主要任务是将像素矩阵显 示在屏幕上，因此，它存在两个子类，分别实现 Windows 系统和 Linux 系统上的图像显 示代码。空缺(1）处主要是设置将在哪个平台上进行实现，因此该处应该存储参数所传递的对象，由于该类的成员变量也是 imp，与参数相同，因此需要填写 this.imp;同理，该成员变量的类型和参数的类型应该保持相同，空(2）处应该填写 Imagelmp；空(3）处需要根据 imp 成员变量存储的实现对象来显示图像；在空(4)处需要生成一个 BMP 对象；由于需要在 Windows 平台上实现，因此空(5）处需要生成一个 Winlmp 对象，同时，还需设置该 BMP 对象，应采用 Winlmp 对象来实现显示。采用桥接模式能够将文件分析代码和图像显示代码分解在不同

的类层次结构中，如界不考虑中间使用的 Matrix 等类，那么最后需要设计的类包括 2 个父类，对应文件格式数目的子类，对应操作系统数目的平台类，因此 10 种图像格式和 5 种操作系统需要 17 个类。