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 2015 年 下半年 下午试卷 案例

（考试时间 150 分钟）

试题一 【说明】

某慕课教育平台欲添加在线作业批改系统，以实现高效的作业提交与批改，并进行统计。学生和讲师的基本信息已经初始化为数据库中的学生表和讲师表。系统的主要功能如下：

(1）提交作业。验证学生标识后，学生将电子作业通过在线的方式提交，并进行存储。系统给学生发送通知表明提交成功，通知中包含唯一编号；并通知讲师有作业提交。

(2）下载未批改作业。验证讲师标识后，讲师从系统中下载学生提交的作业。下载的作业将显示在屏幕上。

(3）批改作业。讲师按格式为每个题目进行批改打分，并进行整体评价。

(4）上传批改后的作业。将批改后的作业(包括分数和评价)返回给系统，进行存储。

(5）记录分数和评价。将批改后的作业的分数和评价记录在学生信息中，并通知学生作业已批改口

(6）获取已批改作业。根据学生标识，给学生查看批改后的作业，包括提交的作业、分数和评价。

(7）作业抽检。根据教务人员标识抽取批改后的作业样本，给出抽检意见，然后形成抽检报告给讲师。

现采用结构化方法对在线作业批改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。

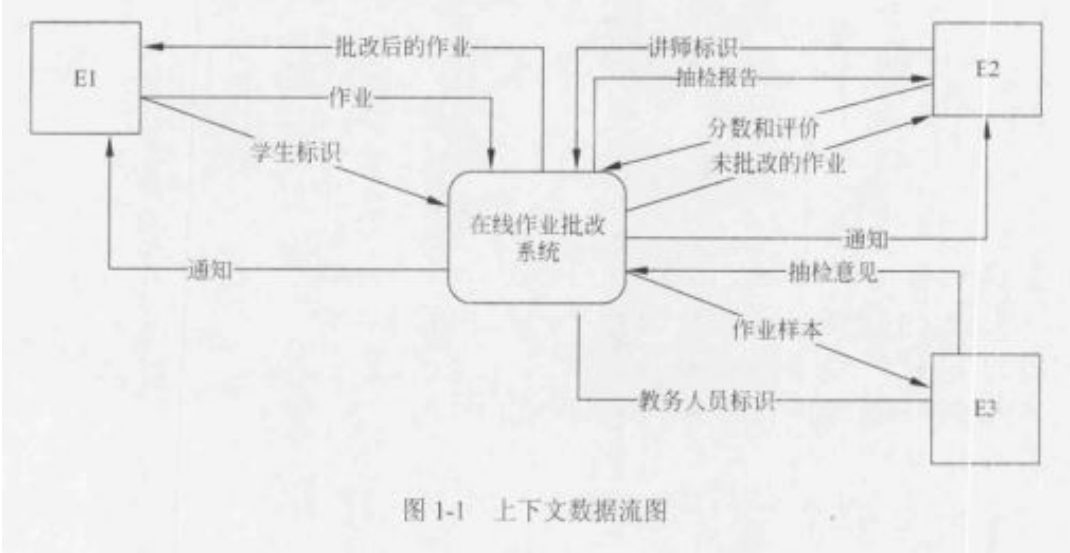


图 1-1　上下文数据流图

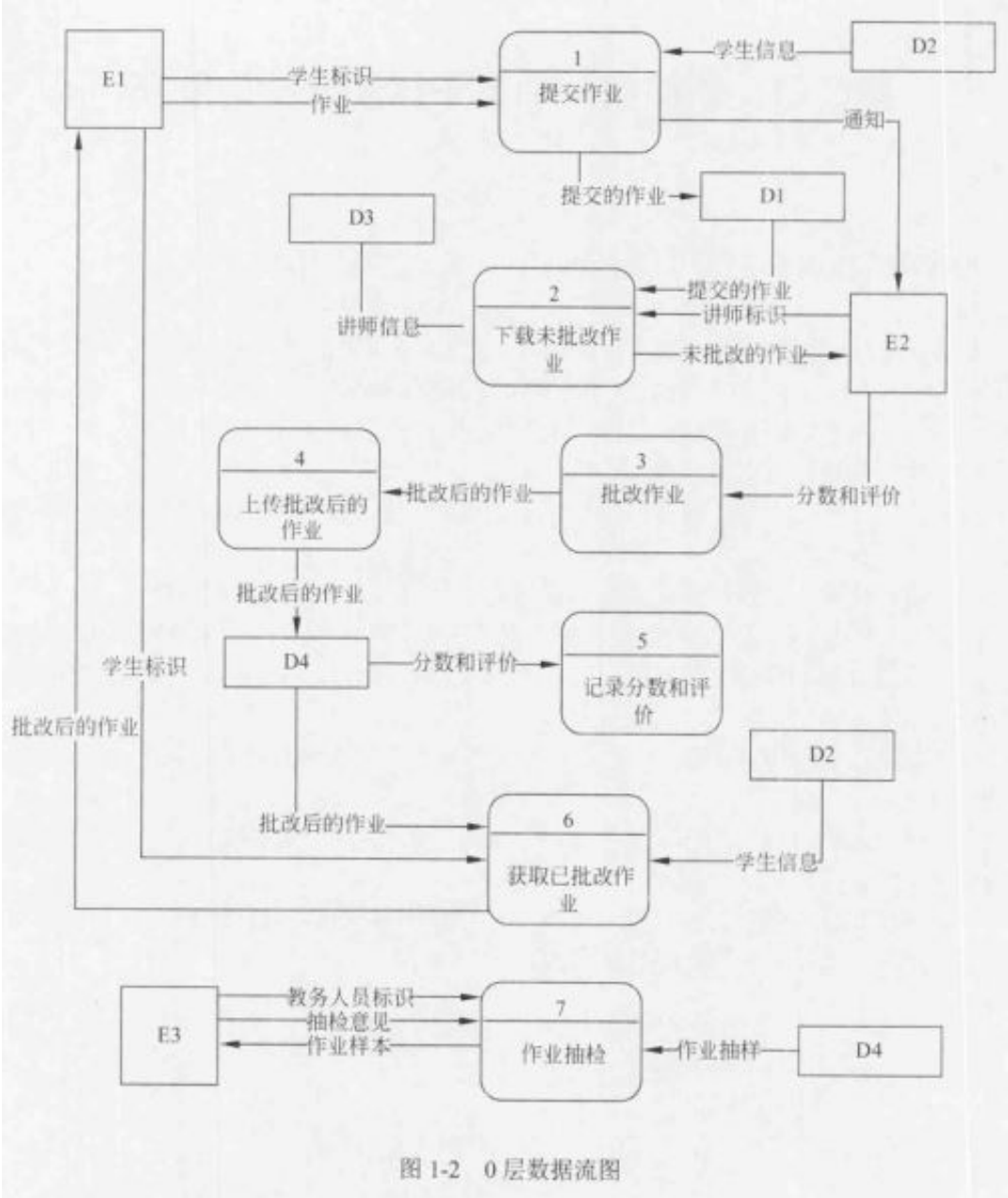


图 1-2　0 层数据流图

问题：1.1

使用说明中的词语，给出图 1-1 中的实体 E1 ～ E3 的名称。

问题：1.2

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：1.3

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：1.4

若发送给学生和讲师的通知是通过第三方 Email 系统进行的，则需要对图 1-1 和图 1-2 进行哪些修改？用 100 字以内文字加以说明。

试题二 【说明】

某企业拟构建一个高效、低成本、符合企业实际发展需要的办公自动化系统。工程师小李主要承担该系统的公告管理和消息管理模块的研发工作。公告管理模块的主要功能包括添加、修改、删除和查看公告。消息管理模块的主要功能是消息群发。

小李根据前期调研和需求分析进行了概念模型设计，具体情况分述如下：

【需求分析结果】

(1) 该企业设有研发部、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，每名员工只属于一个部门，部门信息包括：部门号、名称、部门经理和电话，其中部门号唯一确定部门关系的每一个元组。

(2) 员工信息包括：员工号、姓名、岗位、电话和密码。员工号唯一确定员工关系的每一个元组；岗位主要有经理、部门经理、管理员等，不同岗位具有不同的权限。一名员工只对应一个岗位，但一个岗位可对应多名员工。

(3) 消息信息包括：编号、内容、消息类型、接收人、接收时间、发送时间和发送人。其中(编号，接收人)唯一标识消息关系中的每一个元组。一条消息可以发送给多个接收人，一个接收人可以接收多条消息。

(4) 公告信息包括：编号、标题、名称、内容、发布部门、发布时间。其中编号唯一确定公告关系的每二个元组。一份公告对应一个发布部门，但一个部门可以发布多份公告；一份公告可以被多名员工阅读，一名员工可以阅读多份公告。

【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(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)：

部门((a)，部门经理，电话)

员工(员工号，姓名，岗位号，部门号，电话，密码)

岗位(岗位号，名称，权限)

消息((b)，消息类型，接收时间，发送时间，发送人)

公告((c)，名称，内容，发布部门，发布时间)

阅读公告((d)，阅读时间)

图 2-1　实体联系图

问题：2.1

根据问题描述，补充四个联系，完善图 2-1 所示的实体联系图。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)～(d)补充完整。

(2)给出“消息”和“阅读公告”关系模式的主键与外键。

问题：2.3

消息和公告关系中都有“编号”属性，请问它是属于命名冲突吗？用 100 字以内文字说明原因。

试题三　【说明】

某出版社拟开发一个在线销售各种学术出版物的网上商店(ACShop)，其主要的功能需求描述如下：

(1) ACShop 在线销售的学术出版物包括论文、学术报告或讲座资料等。

(2) ACShop 的客户分为两种：未注册客户和注册客户。

(3) 未注册客户可以浏览或检索出版物，将出版物添加到购物车中。未注册客户进行注册操作之后，成为 ACShop 注册客户。

(4) 注册客户登录之后，可将待购买的出版物添加到购物车中，并进行结账操作。结账操作的具体流程描述如下：

①从预先填写的地址列表中选择一个作为本次交易的收货地址。如果没有地址信息，则可以添加新地址。

②选择付款方式。 ACShop 支持信用卡付款和银行转账两种方式。注册客户可以从预先填写的信用卡或银行账号中选择一个付款。若没有付款方式信息，则可以添加新付款方式。

③确认提交购物车中待购买的出版物后， ACShop 会自动生成与之相对应的订单。

(5) 管理员负责维护在线销售的出版物目录，包括添加新出版物或者更新在售出版物信息等操作。

现采用面向对象方法分析并设计该网上商店 ACShop ，得到如图 3-1 所示的用例图和图 3-2 所示的类图。

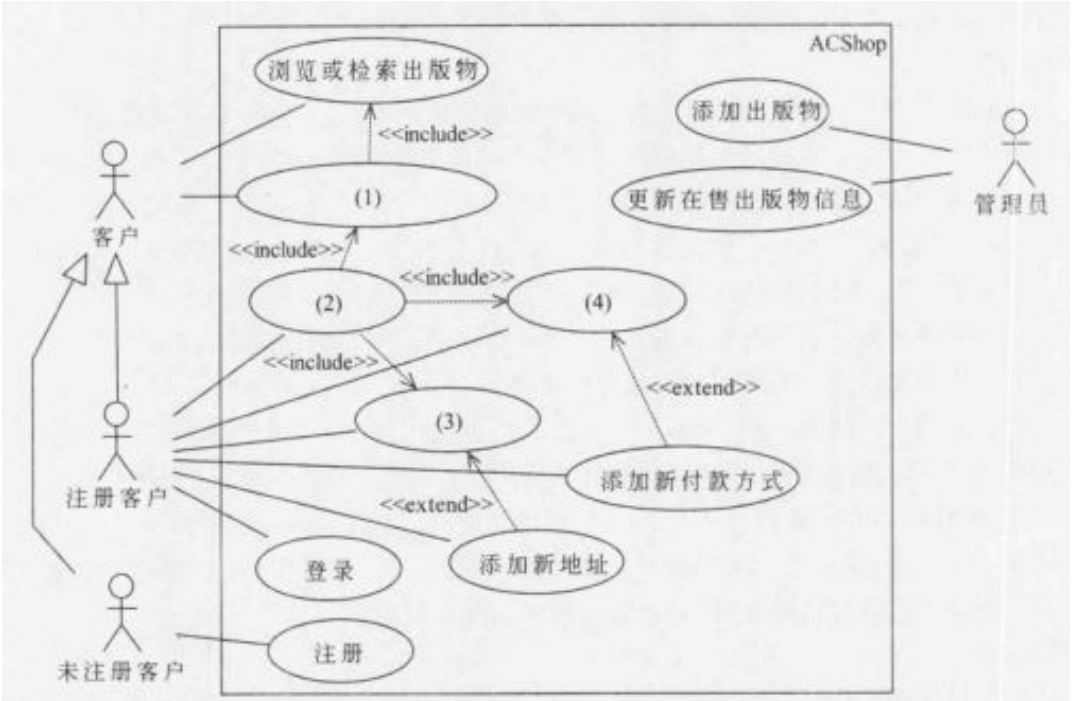


图 3-1　用例图

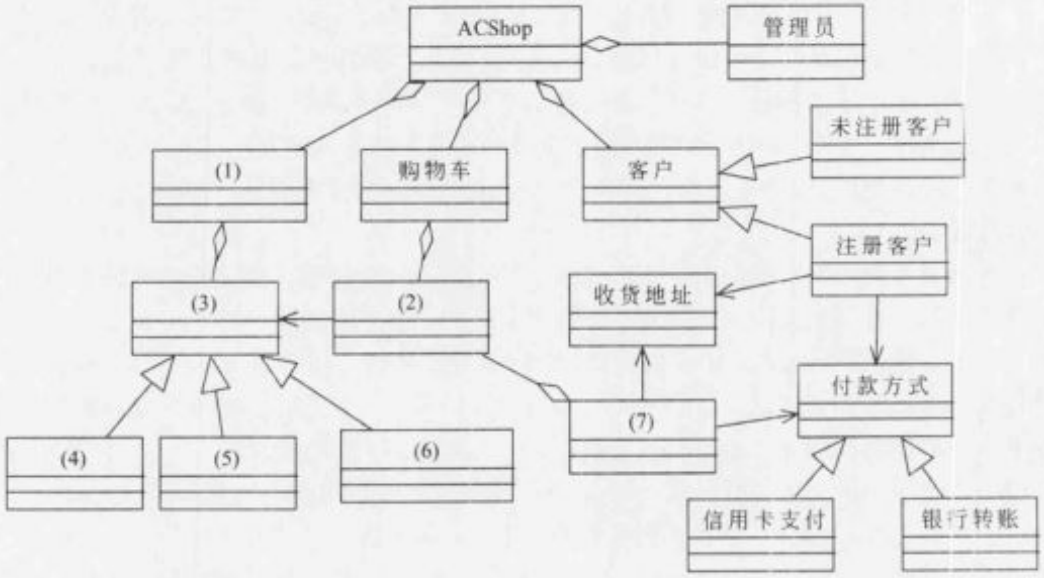


图 3-2　类图

问题：3.1

据说明中的描述，给出图 3-1 中(1)～(4)所对应的用例名。

问题：3.2

根据说明中的描述，分别说明用例“添加新地址”和“添加新付款方式”会在何种情况下由图 3-1 中的用例(3)和(4)扩展而来？

问题：3.3

根据说明中的描述，给出图 3-2 中(1)～(7)所对应的类名。

试题四 【说明】

计算两个字符串 x 和 y 的最长公共子串(LongestCommonSubstring)。

假设字符串 x 和字符串 y 的长度分别为 m 和 n，用数组 c 的元素 c[i][j]记录 x 中前 i 个字符和 y 中前 j 个字符的最长公共子串的长度。

c[i][j]满足最优子结构，其递归定义为：

计算所有 c[i][j](0≤i≤m， 0≤j≤n)的值，值最大的 c[i][j]即为字符串 x 和 y 的最长公共子串的长度。根据该长度即 i 和 j，确定一个最长公共子串。

(1) 常量和变量说明

x， y：长度分别为 m 和 n 的字符串。

c[i][j]：记录 x 中前 i 字符和 y 中前 j 个字符的最长公共子串的长度。

max： x 和 y 的最长公共子串的长度。

maxi， maxj：分别表示 x 和 y 的某个最长公共子串的最后一个字符在 x 和 y 中的位置(序号)。

(2) C 程序

$$c[i][j]=\begin{cases} c[i-1][j-1]+1 & \text{若 } i>0 \text{且} j>0 \text{且} x[i]=y[j] \\ 0 & \text{其他} \end{cases}$$

```
#include <stdio.h>
#include <string.h>

int c[50][50];
int maxi;
int maxj;

int lcs(char *x, int m, char *y, int n) {
    int i, j;
    int max = 0;
    maxi = 0;
    maxj = 0;

    for ( i = 0; i <= m; i++ )    c[i][0] = 0;
    for ( i = 1; i <= n; i++ )    c[0][i] = 0;
    for ( i = 1; i <= m; i++ ) {
      for ( j = 1; j <= n; j++ )  {
        if ( ____(1)____ ) {
          c[i][j] = c[i - 1][j - 1] + 1;
          if( max < c[i][j] ) {
            ____(2)____ ;
            maxi = i;
            maxj = j;
          }
        }
        else ____(3)____ ;
      }
    }
    return max;
}
void printLCS(int max, char *x){
    int i = 0;
    if (max == 0)    return;
    for (____(4)____; i < maxi; i++)
      printf("%c", x[i]);
}
void main() {
char* x = "ABCADAB";
```

```
    char* y = "BDCABA";
    int max = 0;
    int m = strlen(x);
    int n = strlen(y);

    max = lcs(x, m, y, n);
    printLCS(max, x);
}
```

问题： 4.1
根据以上说明和 C 代码，填充 C 代码中的空(1)～(4)。
问题： 4.2
根据题干说明和以上 C 代码，算法采用了(5)设计策略。
分析时间复杂度为(6)(用 O 符号表示)。
问题： 4.3
根据题干说明和以上 C 代码，输入字符串 x="ABCADAB’，’y="BDCABA″,则输出为(7)。

试题五 【说明】

某大型购物中心欲开发一套收银软件，要求其能够支持购物中心在不同时期推出的各

种促销活动，如打折、返利(例如，满 300 返 100)等等。现采用策略(Strategy)模式实现该要求，得到如图 5-1 所示的类图。

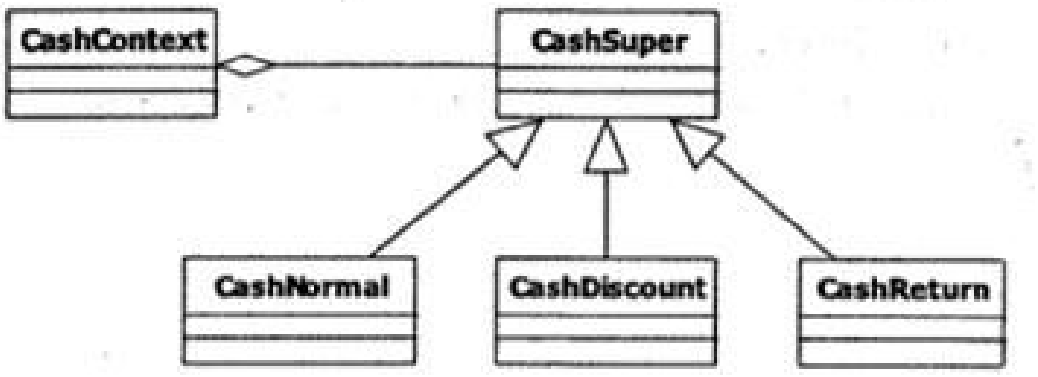


图 5-1 策略模式类图

问题： 5.1

【C++代码】

```
#include <iostream>
using namespace std;
enum TYPE{NORMAL, CASH_DISCOUNT, CASH_RETURN};
class CashSuper{
public:
   (1) ;
};
class CashNormal : public CashSuper {     //正常收费子类
public:
    double acceptCash(double money) {     retum money;    }
};
class CashDiscount : public CashSuper {
private:
    double moneyDiscount;     //  折扣率
public:
   CashDiscount(double discount) {    moneyDiscount= discount;    }
   double acceptCash(double money) {    retum money * moneyDiscount;    }
};
class CashRetum : public CashSuper {      // 满额返利
private:
    double moneyCondition;      // 满额数额
    double moneyReturn;       // 返利数额
public:
    CashRetnm(double motieyCondition, double moneyReturn) {
       this->moneyCondition=moneyCondition;
       this->moneyReturn=moneyRetum;
    }
    double acceptCash(double money) {
       double result = money;
       if(money>=moneyCondition)
           result=money-(int)(money/moneyCondition ) * moneyRetum;
       return  result ;
    }
};
class CashContext {
private:
   CashSuper *cs;
public:
   CashContext(int type) {
     switch(type) {
      case NORMAL:            //正常收费
           ________(2)_______ ;
           break;
      case CASH_RETURN:       //满300返100
           ________(3)________;
           break;
      case CASH_DISCOUNT:     //打八折
           ________(4)_______ ;
           break;
   }
}
double GetResult(double money) {
       _____(5)_____;
    }
};
//此处略去main()函数
```

试题六 【说明】

某大型购物中心欲开发一套收银软件，要求其能够支持购物中心在不同时期推出的各种促销活动，如打折、返利(例如，满 300 返 100)等等。现采用策略(Strategy)模式实现该要求，得到如图 6-1 所示的类图。

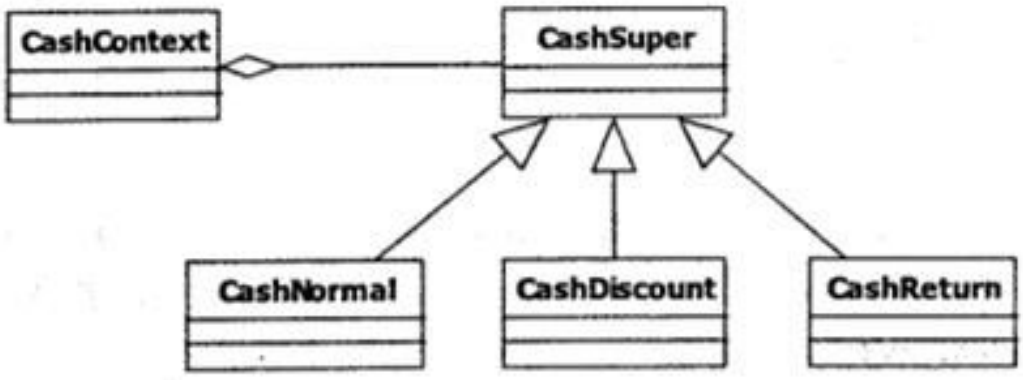


图 6-1 策略模式类图

问题：6.1

【Java代码】

```
import java.util.*;
enum TYPE { NORMAL, CASH_DISCOUNT, CASH_RETURN};
interface CashSuper {
   public    (1)   ;
}
class CashNormal implements CashSuper{    // 正常收费子类
   public double accptCash(double money){
      return money;
   }
}
class CashDiscount implements CashSuper {
   private double moneyDiscount;                    // 折扣率
   public CashDiscount(double moneyDiscount) {
      this moneyDiscount = moneyDiscount;
   }
   public double acceptCash(double money) {
      return money* moneyDiscount;
   }
}
class CashReturn implements CashSuper {              // 满额返利
   private double moneyCondition;
   private double moneyReturn;
   public CashReturn(double moneyCondition, double moneyReturn) {
      this.moneyCondition =moneyCondition;     // 满额数额
      this.moneyReturn =moneyReturn;           // 返利数额
   }
   public double acceptCash(double money) {
      double result = money;
      if(money >= moneyCondition )
         result=money-Math.floor(money/moneyCondition ) * moneyReturn;
      return result;
   }
}
class CashContext_{
   private CashSuper cs;
   private TYPE  t;
   public CashContext(TYPE t)  {
      switch(t){
         case NORMAL:     // 正常收费
               (2)  ;
            break;
         case CASH_DISCOUNT:      // 满300返100
               (3)  ;
            break;
         case CASH_RETURN:      // 打8折
               (4)  ;
            break;
      }
   }
   public double GetResult(double money) {
          (5)    ;
   }
   //此处略去main()函数
}
```

试题一 答案：　解析：　E1：学生 E2：讲师 E3：教务人员

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，建模出的上下文数据流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，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的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。通过考查系统的主要功能不难发现，系统中涉及到学生、讲师和教务人员，没有提到其他与系统交互的外部实体。根据描述(1) 中“学生将电子作业通过在线的方式提交”，(2) 中“讲师从系统中下载学生提交的作业”，(7) 中“根据教务人员标识抽取批改后的作业样本，给出抽检意见”等信息，从而即可确定 E1 为“学生”实体， E2 为“讲师”实体， E3 为“教务人员”实体。

D1：提交的作业表

D2：学生表

D3：讲师表

D4：批改后的作业表

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，说明(1)中“验证学生标识后，学生将电子作业通过在线的方式提交，并进行存储”，说明(2) 中“讲师从系统中下载学生提交的作业”，可知 D1 为提交的作业表；说明(2) 中“验证讲师标识后”，可知 D3 为讲师表；说明(4) 中“将批改后的作业(包括分数和评价)返回给系统，进行存储”，可知 D4 为批改后的作业表。

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“通知”和“抽检报告”，而图 1-2 中缺少了这几条数据流，所以需要确定这几条数据流或者其分解的数据流的起点 或终点。下面考查说明中的功能。先考查“通知”，功能(1) 中“系统给学生发送通知表明提交成功”，对照图 1-2，加工 1 没有到实体 E1 学生的“通知”数据流；功能(5) 中“并通知学生作业已批改”，对照图 1-2，加工 5 没有到实体 E1 学生的数据流“通知”。进一步加以区别，加工 1 到实体 E1 学生的数据流为“提交成功通知”，加工 5 到实体 E1 学生缺少的数据流应为“作业已批改通知”。这两条数据流是上下文数据流图中对数据流“通知”的分解。再根据功能(7) 中“然后形成抽检报告给讲师”，对照图 1-2 中加工 7 应该有数据流“抽检报告”，终点为 E2 讲师实体。

下面再仔细核对说明和图 1-2 之间是否还有遗失的数据流。不难发现，功能(3) 中“将批改后的作业的分数和评价记录在学生信息中”，而图 1-2 中加工 5 从 D4 批改后的作业表中读取了分数和评价，并没有存入学生表，所以，此处遗失了数据流“分数与评价”，起点是加工 5，终点是 D2 学生表。

将 Email 系统作为外部实体，并将通知的终点全部改为 Email 系统。

DFD 中，外部实体可以是用户，也可以是其他交互的系统。如果某功能交互的是外部系统，本题中是通过第三方 Email 系统，即系统需要将发送给学生和教师的通知相关信息发送给第三方 Email 系统。然后由第三方 Email 系统给学生和教师发送邮件，此时第三方 Email 系统即为外部实体，而非本系统内部加工，因此需要对图 1-1 和图 1-2 进行修改，添加外部实体“Email 系统”，并将数据流通知的终点都改为 Email 系统。在图 1-1 中将唯一加工到 E1 和 E2 的通知数据流终点改为“Email 系统”。在图 1-2 中，除了将加工 1 到 E2 的数据流通知的终点改为“Email 系统”，还需要将【问题 3】补充“提交成功通知”和“作业已批改通知”的终点也改为“Email 系统”。

数 据 流	起 点	终 点
提交成功通知	1 或 提交作业	E1 或 学生
作业已批改通知	5 或 记录分数和评价	E1 或 学生
分数和评价	5 或 记录分数和评价	D2 或 学生表
抽检报告	7 或 作业抽检	E2 或 讲师

试题二 答案： **解析：** 联系名称可不做要求，但不能出现重名。

本题考查数据库概念结构设计及概念结构向逻辑结构转换的过程。

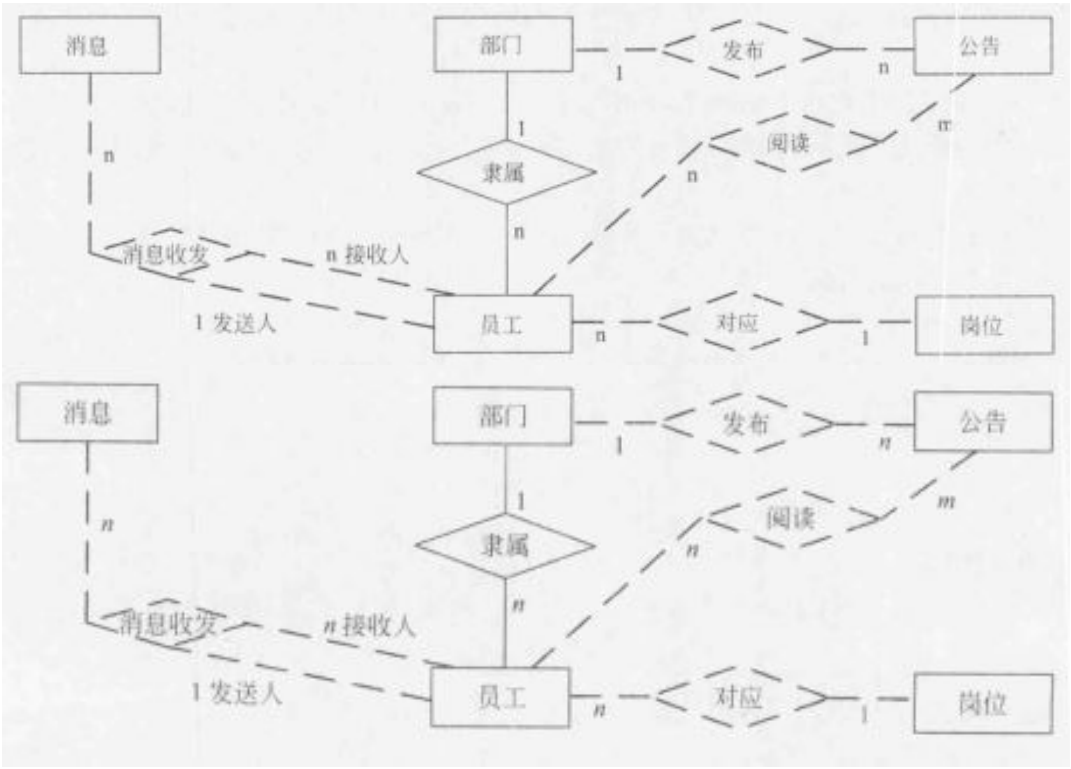
此类题目要求考生认真阅读题目对现实问题的描述，经过分类、聚集、概括等方法，从中确定实体及其联系。题目已经给出了 4 个实体，需要根据需求描述，给出实体间的联系。
根据题意“一个员工只对应一个岗位，但一个岗位可对应多名员工”，可以得出员工与岗位之间的对应联系类型为 n:1 。
由“一条消息可以发送给多个接收人，一个接收人可以接收多条消息”，可以得出员工与消息之间的收发联系类型为 1:n:m 。
由“一份公告对应一个发布部门，但一个部门可以发布多份公告”可以得出部门与公告间的所属联系类型为 1:n 。
由“一份公告可以有多个员工阅读，一个员工可以阅读多份公告”，可以得出，公告与员工之间的阅读联系类型为 n:m 。
完整的 ER 图如下：

(1) (a)部门号，名称
(b)编号，内容，接收人
(c)编号，标题
(d)公告编号，员工号(注：编号，员工号也正确)
(2) 消息关系模式的主键：编号，接收人
外键：接收人、发送人
阅读公告关系模式的主键：公告编号，员工号
外键：公告编号，员工号

(1) 根据题意，完整的关系模式如下：
部门(部门号，名称，部门经理，电话)
员工(员工号，姓名，岗位号，部门号，电话，密码)
岗位(岗位号，名称，权限)
消息(编号，内容，接收人，消息类型，接收时间，发送时间，发送人)
公告(编号，标题，名称，内容，发布部门，发布时间)
阅读公告(公告编号，员工号，阅读时间)
(2) 消息关系模式和阅读公告关系模式的主键和外键的分析如下：
根据题意，消息关系模式的主键为(编号，接收人)。由于接收人、发送人都应参考员工关系的员工号，因此接收人、发送人为消息关系的外键。
根据题意，阅读公告关系模式的主键为(公告编号，员工号)。外键为公告编号、员工号，因为公告编号应参考公告关系的编号，而编号是公告关系的主键，所以公告编号是阅读公告关系的外键；又因为员工号应参考员工关系的员工号，而员工号是员工关系的主键，所

以公告关系的员工号为外键。
不属于命名冲突。因为这两个属性分别属于两个不同的关系模式，可以通过“关系名.属性名”区别，即可以用“消息.编号”和“公告.编号”来区别。

消息和公告关系中都有“编号”属性，但是它们不属于命名冲突。因为这两个属性分别属于两个不同的关系模式，可以通过“关系名.属性名”区别’即可以用“消息.编号”和“公告.编号”来区别。



试题三 答案：　解析：　(1）将(待购买)出版物添加到购物车
(2）结账
(3）选择收货地址
(4）选择付款方式

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。在建模方面，本题中涉及到了 UML 的用例图与类图。本题属于比较经典的考题，难度不大。
本问题考查 UML 用例图，要求将图中缺失的用例(1) ~(4）补充完整。解答此类题目的时候，根据给出的用例图对照说明中的功能需求描述，就可以完成。
首先(1）处的用例与参与者“客户”相关，而“客户”又分为“注册客户”和“未注册客

户”，那么(1）处所代表的用例，是“注册客户”和“未注册客户”都具有的行为。由说明可知，(1）处的用例为“将(待购买)出版物添加到购物车”。
(2）~(3）处的用例与参与者“注册客户”相关，对照说明确定没有在用例图上表示出来的注册客户的行为即可，同时应注意用例(3）与“添加新地址”、用例(4）与“添加新付款方式”之间的扩展(extend)关系。根据说明可知，“注册客户”一个很重要的行为是“结账”，而这个行为在用例图恰好没有表示出来。再者，由说明中给出的结账操作的具体流程可知，结账操作中包含了选择地址和选择付款方式，与用例图中(2）和(3）、
(2）和(4）之间的包含(include)关系对应，因此(2）处的用例为“结账”；而(3）处的用例为“选择收货地址”、(4）处的用例为“选择付款方式”。

“添加新地址”的扩展条件：地址信息为空或没有地址信息。
“添加新付款方式”的扩展条件：付款方式信息为空或没有付款方式信息。

扩展是用例之间的一种关联关系。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，这样使描述可能更加清晰。
用例(3）和(4）在结账操作的流程中给出了详细的描述：“如果没有地址信息，可以添加新地址信息”、“若没有付款方式信息，则可以添加新付款方式”。所以用例“添加新地址”和“添加新付款方式”分别是用例(3）和(4）的一种分支情况，其扩展点就是分支条件。所以“添加新地址”的扩展条件：地址信息为空或没有地址信息；“添加新付款方式”的扩展条件：付款方式信息为空或没有付款方式信息。
(1）目录或出版物目录
(2）待购买的出版物
(3）出版物
(4）论文
(5）学术报告
(6）讲座资料
(7）订单
注：(4）（ 6）答案次序可以互换。

本问题考查 UML 的类图，要求将图中缺失的类补充完整，是比较传统的考法。在解答此题时，可以先关注一下需要填写的类之间的关系。由类图可知，主要是两大类关系：聚集关系和继承关系。由说明可知，在题目中存在着 3 组继承关系：“ACShop 在线销售的学术出版物包括论文、学术报告或讲座资料等”；“ACShop 的客户分为两种：未注册客户和注册

客户”；“ACShop 支持信用卡付款或银行转账两种方式”。后 2 组继承关系已经在类图中给出了，所以空(3) （ 6）处要表达的就是第 1 组继承关系。由此可知，空(3）处应填入“(学术)出版物”，(4) ~(6）处分别是“论文”、“学术报告”和“讲座资料”。类(3)和类(1）之间是聚集关系，而现在已经知道类(3）表示的是“出版物”。由说明可知，与“出版物”之间具有聚集关系的应该是“出版物目录”，因此(1）处应填入“出版物目录”。

类(2）与类“购物车”之间具有聚集关系，购物车中包含的是“待购买的出版物”，因此(2）处应填入“待购买的出版物”。由此也可以确定(7）处应该填入的类是“订单”。

试题四 答案： 解析： (1) x[i-1]==y[j-1]

(2) max=c[i][j]

(3) c[i][j]=0

(4) i=maxi-max

本题考查算法设计与分析和 C 语言实现算法的相关技术。

此类题目要求考生认真阅读题目，首先理解问题以及求解问题的算法思路。

根据题干说明，给出的问题具有最优子结构，考生应该能想到该题用动态规划或者贪心求解。一般在给出递归定义最优解时，已经比较清楚地给出要用动态规划方法，并且根据给出的 C 程序，可知以自底向上的方式进行计算，即先求小规模问题的，再求规模更大的问题的解。进入到 C 程序内部，函数 lcs 是计算 c 数组，并确定其最大的元素。在两重循环内，应该是递归公式的迭代求解过程，因此空(1）处填入“x[i-1]==y[j-1]”；若当前的最大长度小于 c[i][j]，则应该更新当前最大长度，即空(2）处填入“max=c[i][j]”；空(3）前面是 else 与 if 对应，即是 x[i-1]≠y[j-1]的情况，根据递归式此处填入“c[i][j]=0”；函数 printLCS 是根据函数 lcs 计算的结果输出最长公共子串，长度为 max，在串 x 中的最后位置是 maxi，而在串 y 中的最后位置是 maxj，因此，空(4）填入“i=maxi-max”。

(5）动态规划

(6) O(m×n)或 O(mn)

根据【问题 1 】中的分析，已知算法采用动态规划技术，算法的时间复杂度分析过程为：

(1）函数 lcs 中，有两个一重循环和一个两重循环，时间复杂度为 m+n+mn;

(2) 函数 printLCS 中，有一个一重循环，时间复杂度为 m(或 n)。

故算法的时间复杂度为 O(mn)。

(7) AB

根据题干和 C 代码，计算出下表的值。

最大值为 2 。在计算过程中，我们记录第一个最大值，即表中阴影部分元素，因此得到最长公共子串为 AB 。

表 1 二维数组 c

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
A	1	0	0	0	0	1	0	1
B	2	0	1	0	0	0	2	0
C	3	0	0	0	1	0	0	0
A	4	0	0	0	0	2	0	1
D	5	0	0	1	0	0	0	0
A	6	0	0	0	0	1	0	1
B	7	0	1	0	0	0	2	0

试题五 答案： **解析：** (1) virtual double acceptCash(double money)=0

(2) cs = new CashNormal()

(3) cs = new CashReturn(300,100)

(4) cs = new CashDiscount(0.8)

(5) return cs->acceptCash(money)

本题考查策略(Strategy)模式的基本概念和应用。

Strategy 模式的设计意图是，定义一系列的算法，把它们一个个封装起来，并且使它们可以相互替换。此模式使得算法可以独立于使用它们的客户而变化，其结构图如下图所示。

•需要使用一个算法的不同变体。例如，定义一些反应不同空间的空间/时间权衡的算法。当这些变体实现为一个算法的类层次时，可以使用策略模式。

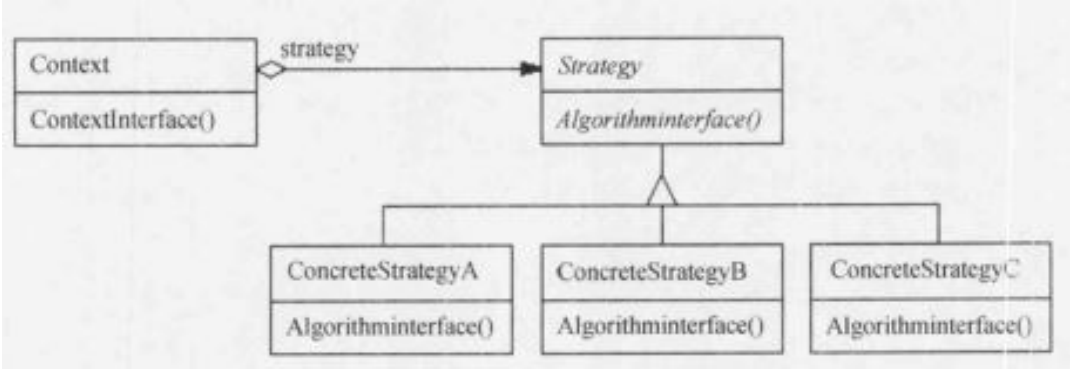
•算法使用客户不应该知道的数据。可使用策略模式以避免暴露复杂的、与算法相关的数据结构。

一个类定义了多种行为，并且这些行为在这个类的操作中以多个条件语句的形式出现，将相关的条件分支移入它们各自的 Strategy 类中，以代替这些条件语句。

本题中类 CashSuper 对应于上图中的类 Strategy，类 CashNormal 、 CashDiscout 和 CashRetum 分别代表 3 种不同的具休促销策略。 CashSuper 类提供其 3 个子类的公共操作接口，由子类给出 3 种不同促销策略的具体实现。在 C++语言中，可以采用继承+(纯)虚拟函数来实现。从 3 个子类 CashNormal 、 CashDiscout 和 CashRetum 的代码可以看出，公共操作接口为 doubleacceptCash(doublemoney)。由于不需要父类 CashSuper 提供任何 的促销实现方式，所以这里采用纯虚拟函数。应填入空(1) 处的语句是“virtual doubleacceptCash(doublemoney)=0”。

空(2) （ 4）都出现在类 CashContext 中，该类对应于上图中的类 Context，其作用是依据策略对象来调用不同的策略算法。因此空(2) （ 4）是根据不同的 case 分支来创建不同的策略对象。由此可知空(2) 4)分别应填入“cs=newCashNormal() ”、“cs=new CashRetum(300， 100)”和“cs=newCashDiscount(0_8)”。

方法 GetResult 是对接口的调用，从而计算出采用不同促销策略之后应付的费用，这里需要通过 CashSuper 的对象 cs 来调用公共操作接口，因此第(5) 空应填入“returncs-> acceptCash(money)”。



试题六 答案： **解析：** (1) doubleacceptCash(double money)

(2) cs=new CashNormal()

(3) cs=new CashReturn(300,100)

(4) cs=new CashDiscount(0.8)

(5) return cs.acceptCash(money)

本题考查策略(Strategy)模式的基本概念和应用。

Strategy 模式的设计意图是，定义一系列的算法，把它们一个个封装起来，并且使它们可以相互替换。此模式使得算法可以独立于使用它们的客户而变化，其结构图如下图所示。

Strategy(策略)定义所有支持的算法的公共接口。 Context 使用这个接口来调用某 ConcreteStrategy 定义的算法。

•ConcreteStrategy(具体策略)以 Strategy 接口实现某具体算法。

•Context(上下文)用一个 ConcreteStrategy 对象来配置；维护一个对 Strategy 对象的引用；可定义一个接口来让 Strategy 访问它的数据。

Strategy 模式适用于：

•许多相关的类仅仅是行为有异。“策略”提供了一种用多个行为中的一个行为来配置一个类的方法。

•需要使用一个算法的不同变体。例如，定义一些反应不同空间的空间/时间权衡的算法。当这些变体实现为一个算法的类层次时，可以使用策略模式。

•算法使用客户不应该知道的数据。可使用策略模式以避免暴露复杂的、与算法相关的数据结构。

一个类定义了多种行为，并且这些行为在这个类的操作中以多个条件语句的形式出现，将相关的条件分支移入它们各自的 Strategy 类中，以代替这些条件语句。

本题中类 CashSuper 对应于上图中的类 Strategy，类 CashNormal 、 CashDiscout 和 CashRetum 分别代表 3 种不同的具体促销策略。 CashSuper 类提供其 3 个子类的公共操作接口，由子类给出 3 种不同促销策略的具体实现。这里采用了 Java 中的接口(Interface)来实现。从 3 个子类 CashNormal 、 CashDiscout 和 CashRetum 的代码可以看出，公共操作接口为 doubleacceptCash(doublemoney)，因此应填入空(1) 处的语句是“ doubleacceptCash(doublemoney)”。

空(2) （ 4）都出现在类 CashContext 中，该类对应于上图中的类 Comext，其作用是依据策略对象来调用不同的策略算法。因此空(2) （ 4）是根据不同的 case 分支来创建不同的策略对象。由此可知空(2) （ 4）分别应填入“cs=newCashNormal()”、“cs=new CashDiscount(0.8)”和“cs=newCashRetum(300， 100)”。

方法 GetResult 是对接口的调用，从而计算出采用不同促销策略之后应付的费用，这里需要通过 CashSuper 的对象 cs 来调用公共操作接口，因此第(5) 空应填入“returncs. acceptCash(money)”。