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 下半年 下午试卷 案例

（考试时间 150 分钟）

试题一 现准备为某银行开发一个信用卡管理系统 CCMS,该系统的基本功能为：

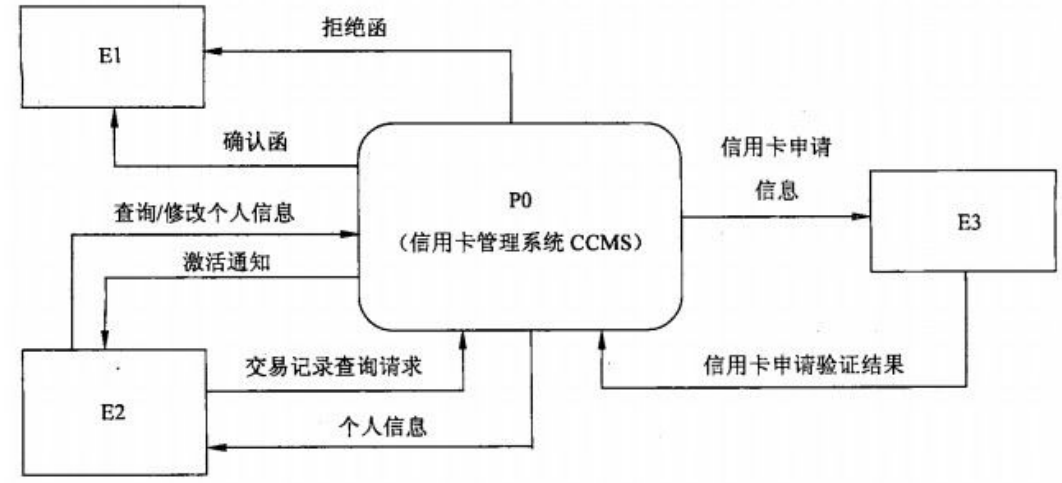
1. 信用卡申请。非信用卡客户填写信用卡申请表，说明所要申请的信用卡类型及申请者的基本信息，提交 CCMS 。如果信用卡申请被银行接受， CCMS 将记录该客户的基本信息，并发送确认函给该客户，告知客户信用卡的有效期及信贷限额；否则该客户 将会收到一封拒绝函。非信用卡客户收到确认函后成为信用卡客户。

2. 信用卡激活。信用卡客户向 CCMS 提交激活请求，用信用卡号和密码激活该信用卡。激活操作结束后， CCMS 将激活通知发送给客户，告知客户其信用卡是否被成功激活。

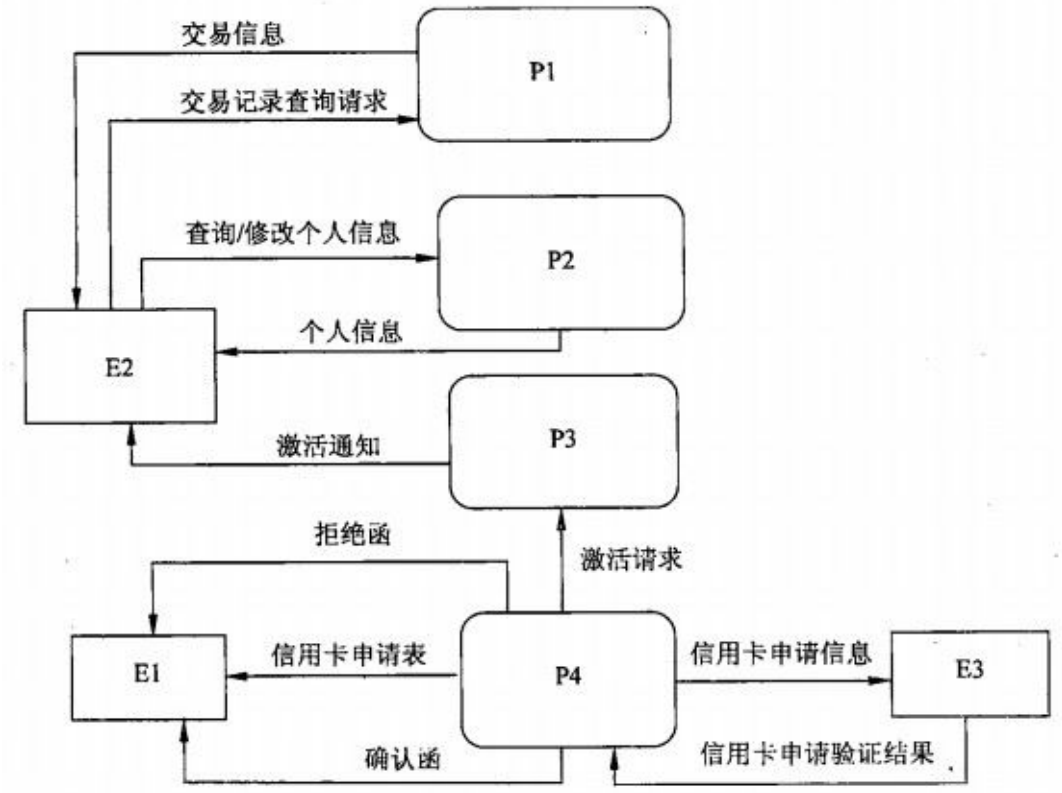
3. 信用卡客户信息管理。信用卡客户的个人信息可以在 CCMS 中进行在线管理。每位信用卡客户可以在线查询和修改个人信息。

4. 交易信息查询。信用卡客户使用信用卡进行的每一笔交易都会记录在 CCMS 中。信用卡客户可以通过 CCMS 查询并核实其交易信息(包括信用卡交易记录及交易额)。

下图(a)和(b)分别给出了该系统的顶层数据流图和 0 层数据流图的初稿。



（a）顶层数据流图



（b）0 层数据流图

问题： 1.1

根据说明，将图(a)中的 E1　　E3 填充完整。

问题： 1.2

图(a)中缺少三条数据流，根据说明，分别指出这三条数据流的起点和终点。(注： 数据流的起点和终点均采用图中的符号和描述)

问题： 1.3

图(b)中有两条数据流是错误的，请指出这两条数据流的名称，并改正。(注：数据流的起点和终点均采用图中的符号和描述)

问题：1.4
根据说明，将图(b)中 P1　P4 的处理名称填充完整。

试题二　某公司拟开发一多用户电子邮件客户端系统，部分功能的初步需求分析结果如下：

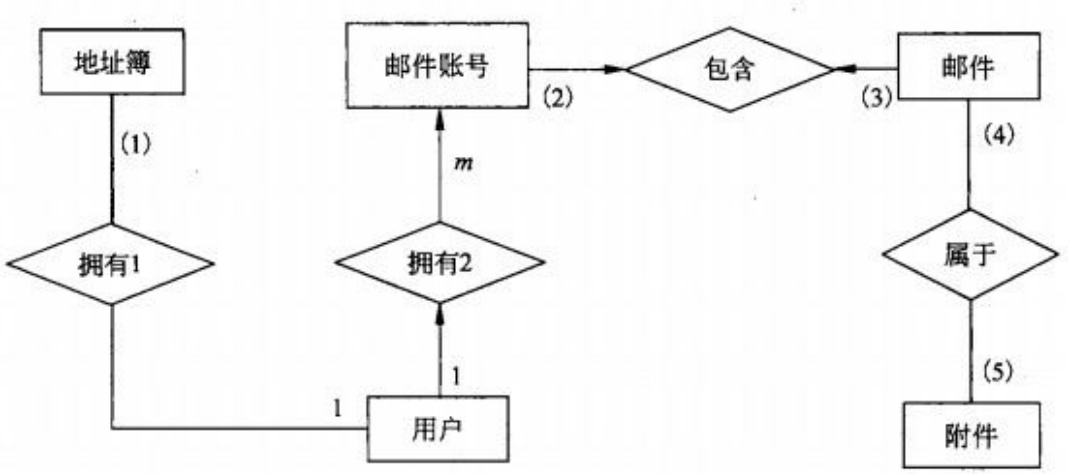
(1）邮件客户端系统支持多个用户，用户信息主要包括用户名和用户密码，且系统 中的用户名不可重复。

(2）邮件账号信息包括邮件地址及其相应的密码，一个用户可以拥有多个邮件地址（如 user1@123.com)。

(3）一个用户可拥有一个地址簿，地址簿信息包括联系人编号、姓名、电话、单位 地址、邮件地址 1 、邮件地址 2 、邮件地址 3 等信息。地址簿中一个联系人只能属于一个用户，且联系人编号唯一标识一个联系人。

(4）一个邮件账号可以含有多封邮件，一封邮件可以含有多个附件。邮件主要包括 邮件号、发件人地址、收件人地址、邮件状态、邮件主题、邮件内容、发送时间、接收时间。其中，邮件号在整个系统内唯一标识一封邮件，邮件状态有已接收、待发送、已发送和已删除 4 种，分别表示邮件是属于收件箱、发件箱、已发送箱和废件箱。一封邮件可以发送给多个用户。附件信息主要包括附件号、附件文件名、附件大小。一个附件 只属于一封邮件，附件号仅在一封邮件内唯一。

问题：2.1
根据以上说明设计的 E-R 图如下图所示，请指出地址簿与用户、电子邮件账号与邮 件、邮件与附件之间的联系类型。



电子邮件客户端系统 E-R 图

问题： 2.2

该邮件客户端系统的主要关系模式如下，请填补（a）～（c）的空缺部分。

用户（用户名，用户密码）

地址簿（ （a） ，联系人编号，姓名，电话，单位地址，邮件地址 1，邮件地址 2，邮件地址 3）

邮件账号（邮件地址，邮件密码，用户名）

邮件（ （b） ，收件人地址，邮件状态，邮件主题，邮件内容，发送时间，接收时间）

附件（ （c） ，附件号，附件文件名，附件大小）

问题： 2.3

(1）请指出问题 2 中给出的地址簿、邮件和附件关系模式的主键，如果关系模式存在外键请指出。（2）附件属于弱实体吗？请用 50 字以内的文字说明原因。

试题三 某企业为了方便员工用餐，为餐厅开发了一个订餐系统(COS: CafeteriaOrdering System)，企业员工可通过企业内联网使用该系统。

企业的任何员工都可以查看菜单和今日特价。

系统的顾客是注册到系统的员工，可以订餐(如果未登录，需先登录)、注册工资支付、预约规律的订餐，在特殊情况下可以覆盖预订。

餐厅员工是特殊顾客，可以进行备餐、生成付费请求和请求送餐，其中对于注册工资支付的顾客生成付费请求并发送给工资系统。

菜单管理员是餐厅特定员工，可以管理菜单。

送餐员可以打印送餐说明，记录送餐信息(如送餐时间)以及记录收费(对于没有注册工资支付的顾客，由送餐员收取现金后记录)。

顾客订餐过程如下：

1. 顾客请求查看菜单；

2. 系统显示菜单和今日特价；

3. 顾客选菜；

4. 系统显示订单和价格；

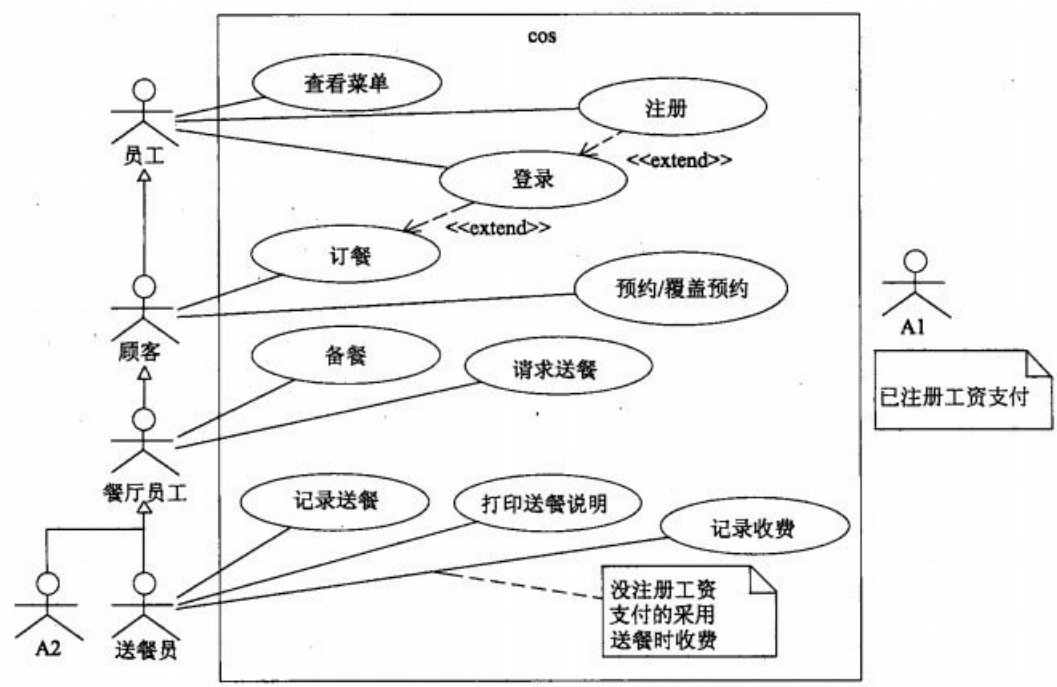
.5. 顾客确认订单；

6. 系统显示可送餐时间；

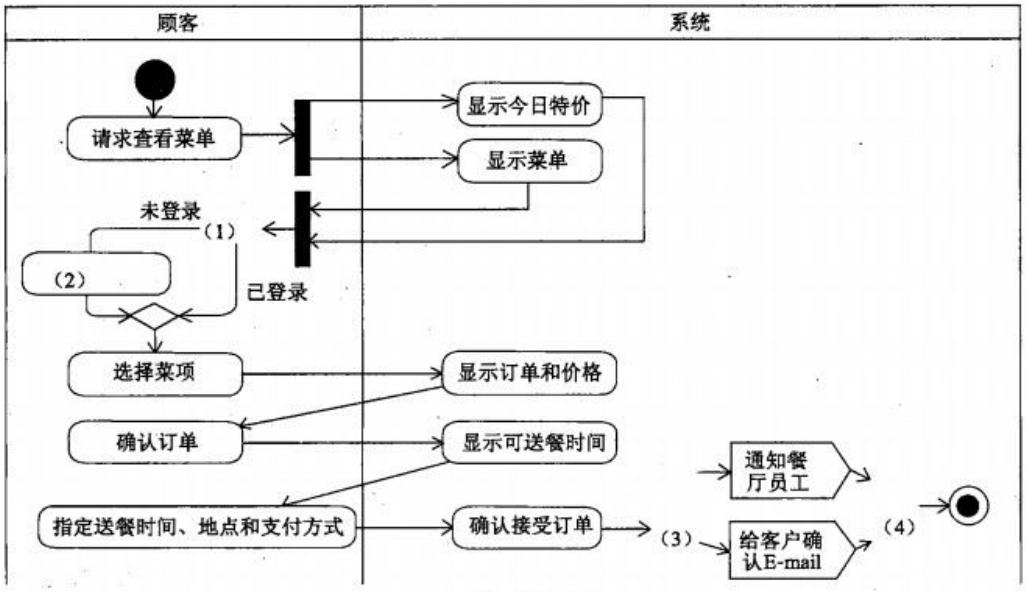
7. 顾客指定送餐时间、地点和支付方式；

8. 系统确认接受订单，然后发送 E-mail 给顾客以确认订餐，同时发送相关订餐信息通知给餐厅员工。

系统采用面向对象方法开发，使用 UML 进行建模。系统的顶层用例图和一次订餐的活动图初稿分别如下图(a)和(b)所示。



（a）COS 系统顶层用例图



（b）一次订餐的活动图

问题： 3.1

根据说明中的描述，给出图（a）中 A1 和 A2 所对应的参与者。

问题： 3.2

根据说明中的描述，给出图（a)中缺少的四个用例及其所对应的参与者。

问题： 3.3

根据说明中的描述，给出图（b）中(1）（ 4)处对应的活动名称或图形符号。

问题： 3.4

指出图(a)中员工和顾客之间是什么关系，并解释该关系的内涵。

试题四

0-1 背包问题可以描述为：有 n 个物品，对 $i=1,2,\cdots,n$，第 i 个物品价值为 v_i，重量为 w_i（v_i 和 w_i 为非负数），背包容量为 W（W 为非负数），选择其中一些物品装入背包，使装入背包物品的总价值最大，即 $\max\sum_{i=1}^{n}v_ix_i$，且总重量不超过背包容量，即 $\sum_{i=1}^{n}w_ix_i \leqslant W$，其中，$x_i \in \{0,1\}$，$x_i$=0 表示第 i 个物品不放入背包，x_i=1 表示第 i 个物品放入背包。

问题：4.1

用回溯法求解此 0-1 背包问题，请填充下面伪代码中(1)～(4)处空缺。 回溯法是一种系统的搜索方法。在确定解空间后，回溯法从根结点开始，按照深度优先策略遍历解空间树，搜索满足约束条件的解。对每一个当前结点，若扩展该结点已经不满足约束条件，则不再继续扩展。为了进一步提高算法的搜索效率，往往需要设计一个限界函数，判断并剪枝那些即使扩展了也不能得到最优解的结点。现在假设已经设计了 BOUND(v,w,k,W)的函数，其中 v、w、k 和 W 分别表示当前已经获得的价值、当前背包的重量、已经确定是否选择的物品数和背包的总容量。对应于搜索树中的某个结点， 该函数值表示确定了部分物品是否选择之后，对剩下的物品在满足约束条件的前提下进行选择可能获得的最大价值，若该价值小于等于当前已经得到的最优解，则该结点无需再扩展。 下面给出 0-1 背包问题的回溯算法伪代码。 函数参数说明如下： W：背包容量； n：物品个数； w:重量数组； v:价值数组； fw:获得最大价值时背包的重量； fp:背包获得的最大价值； X：问题的最优解。变量说明如下： CW：当前的背包重量； cp：当前获#的价值； k：当前考虑的物品编号；Y:当前已获得的部分解。

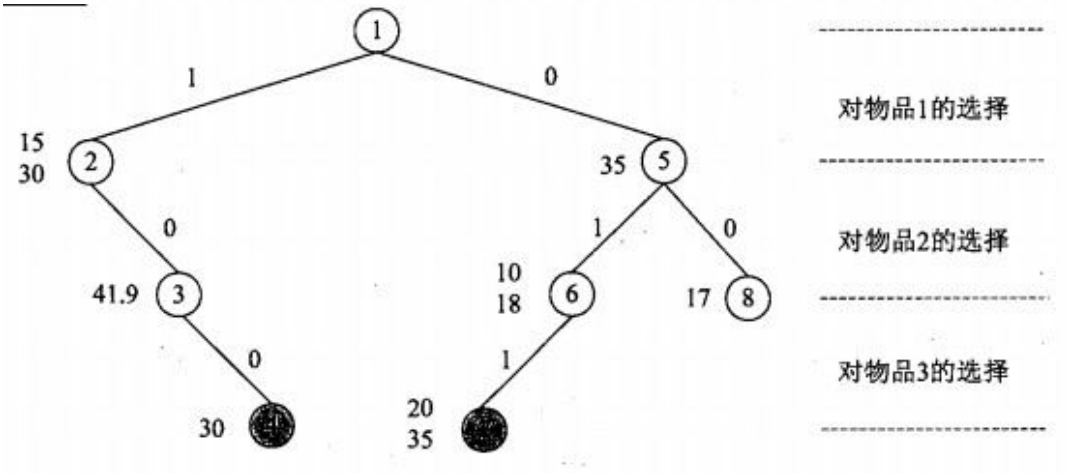
```
BKNAP( W,n,w,v,fw,fp,X )
1  cw ← cp ← 0
2     (1)
3  fp ← -1
4  while true
5      while k ≤ n and cw + w[k] ≤ W do
6            (2)
7          cp ← cp + v[k]
8          Y[k] ← 1
9          k ← k + 1
10     if k > n then
11             if fp < cp then
12                     fp ← cp
13                     fw ← cw
14                     k ← n
15                     X ← Y
16      else Y(k) ← 0
17      while BOUND(cp,cw,k,W) ≤ fp do
18          while k ≠ 0 and Y(k) ≠ 1 do
19                (3)
20          if k = 0 then return
21          Y[k] ← 0
22          cw ← cw - w[k]
23          cp ← cp - v[k]
24        (4)
```

问题：4.2

考虑下表所示的实例，假设有 3 个物品，背包容量为 22 。 下图是根据上述算法构造的搜索树，其中结点的编号表示了搜索树生成的顺序，边上的数字 1/0 分别表示选择/不选择对应物品。除了根结点之外，每个左孩子结点旁边的上下两个数字分别表示当前背包的重量和已获得的价值，右孩子结点旁边的数字表示扩展了该结点后最多可能获得的价值。为获得最优解，应该选择物品(5),获得的价值 为(6)。. 对于上述实例，若采用穷举法搜索骜个解空间，则搜索树的结点数为(7)，而用了上述回溯法，搜索树的结点数为(8)。

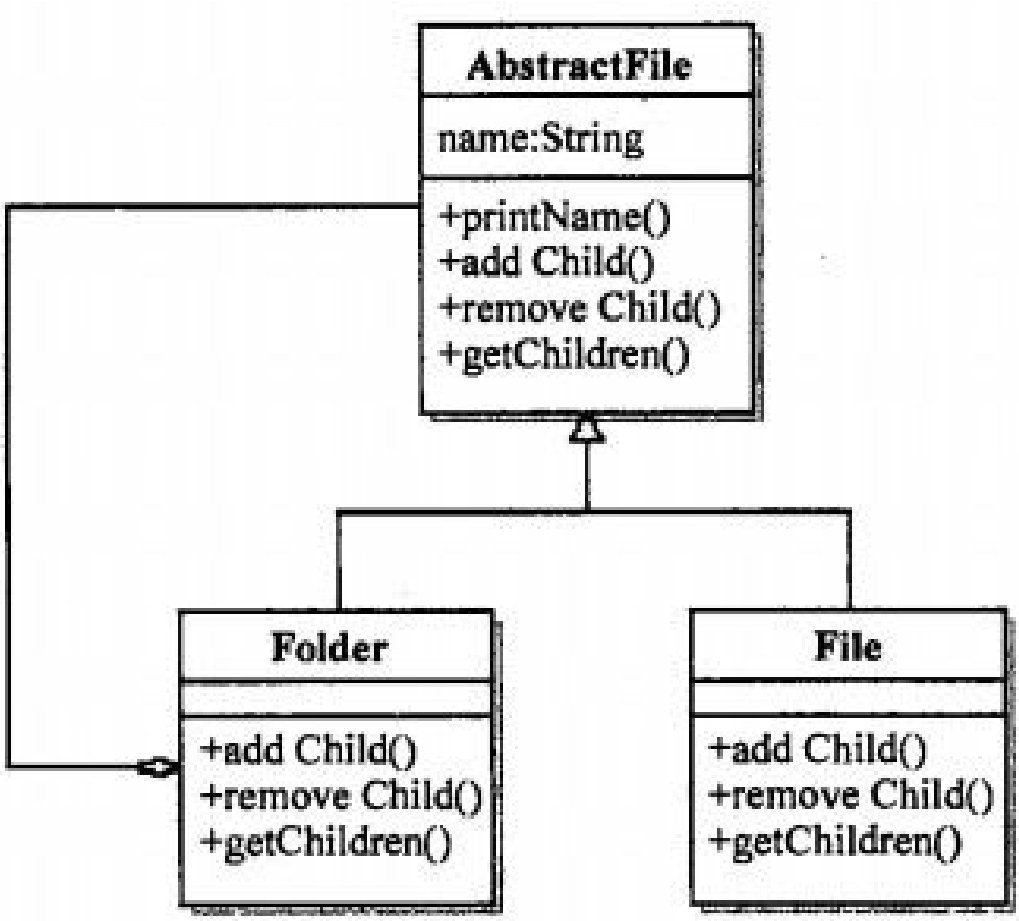
0-1 背包问题实例

	物品 1	物品 2	物品 3
重量	15	10	10
价值	30	18	17
单位价值	2	1.8	1.7



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。

现欲构造一文件/目录树，采用组合(Composite)设计模式来设计，得到的类图如下图所示：



问题： 5.1

【C++代码】

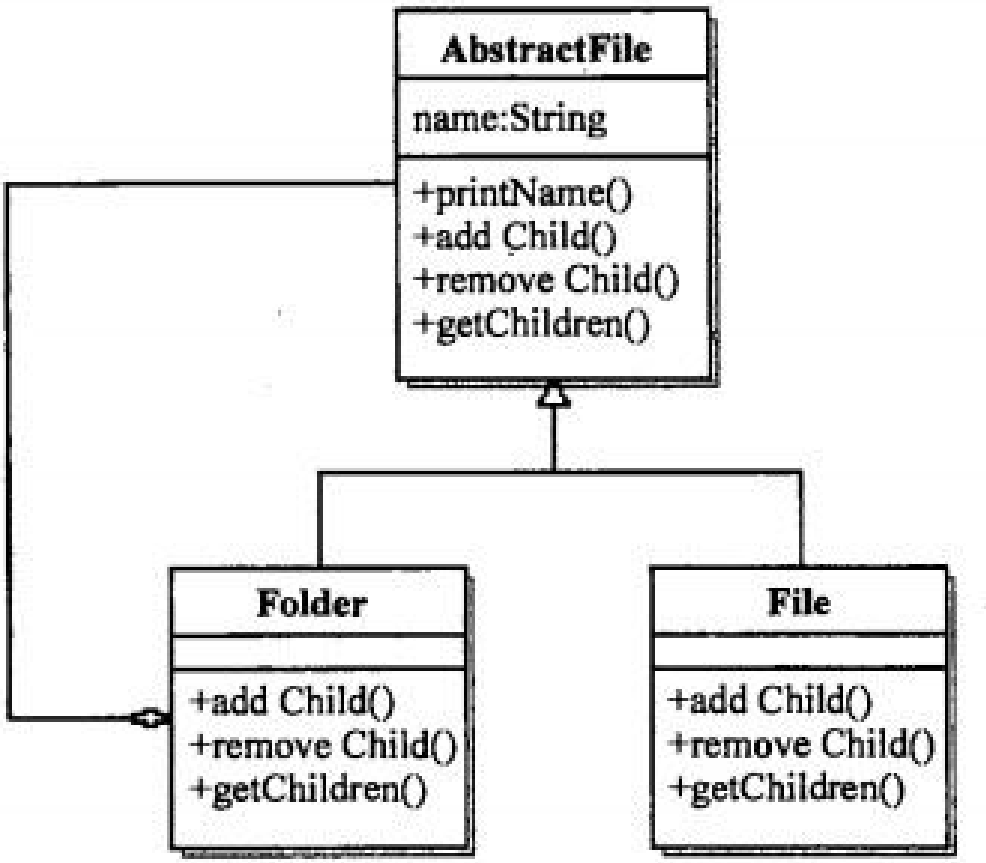
```
#include <list>
#include <iostream>
#include <string>
using namespace std;

class AbstractFile {
protected :
    string name;  // 文件或目录名称
public:
    void printName(){cout << name;}          // 打印文件或目录名称
    virtual void addChild(AbstractFile  *file)=0;
                                             // 给一个目录增加子目录或文件
    virtual void removeChild(AbstractFile *file)=0;
                                             // 删除一个目录的子目录或文件
    virtual list<AbstractFile*> *getChildren()=0;
                                             // 获得一个目录的子目录或文件
};
```

```
class File : public AbstractFile {
public :
    File(string name) {  (1)  = name; }
    void addChild(AbstractFile  *file) { return ; }
    void removeChild(AbstractFile *file) { return ; }
     (2)  getChildren() { return  (3) ; }
};
class Folder :public AbstractFile {
private :
    list <AbstractFile*> childList;  // 存储子目录或文件
public :
    Folder(string name) {  (4)  = name; }
    void addChild(AbstractFile *file) { childList.push_back(file); }
    void removeChild(AbstractFile *file) { childList.remove(file);}
    list<AbstractFile*> *getChildren() { return  (5) ; }
};
void main() {
  // 构造一个树形的文件/目录结构
  AbstractFile *rootFolder = new Folder("c:\\");
  AbstractFile *compositeFolder = new Folder("composite");
  AbstractFile *windowsFolder = new Folder("windows");
  AbstractFile *file = new File("TestComposite.java");
  rootFolder->addChild(compositeFolder);
  rootFolder->addChild(windowsFolder);
  compositeFolder->addChild(file);
}
```

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】

现欲构造一文件/目录树，采用组合(Composite)设计模式来设计，得到的类图如下图所示：


AbstractFile
name:String
+printName()
+add Child()
+remove Child()
+getChildren()
Folder
+add Child()
+remove Child()
+getChildren()
File
+add Child()
+remove Child()
+getChildren()

问题： **6.1**

【Java 代码】

```
import java.util.ArrayList;
import java.util.List;
 (1)  class AbstractFile {
    protected String name;
    public void printName(){System.out.println(name);}
    public abstract boolean addChild(AbstractFile file);
    public abstract boolean removeChild(AbstractFile file);
    public abstract List<AbstractFile> getChildren();
}
class File extends AbstractFile {
    public File(String name) { this.name = name; }
    public boolean addChild(AbstractFile file) { return false; }
    public boolean removeChild(AbstractFile file) { return false; }
    public List<AbstractFile> getChildren() { return  (2) ; }
}
class Folder extends AbstractFile {
    private List <AbstractFile> childList;
    public Folder(String name) {
        this.name = name;
        this.childList = new ArrayList<AbstractFile>();
    }
    public boolean removeChild(AbstractFile file) { return childList.
    remove(file); }
    public  (3) <AbstractFile> getChildren() { return  (4) ; }
    }
    public class Client {
        public static void main(String[] args) {
            // 构造一个树形的文件/目录结构
        AbstractFile rootFolder = new Folder("c:\\");
        AbstractFile compositeFolder = new Folder("composite");
        AbstractFile windowsFolder = new Folder("windows");
        AbstractFile file = new File("TestComposite.java");
            rootFolder.addChild(compositeFolder);
            rootFolder.addChild(windowsFolder);
            compositeFolder.addChild(file);
```

```
        // 打印目录文件树
        printTree(rootFolder);
    }
    private static void printTree(AbstractFile ifile) {
     ifile.printName();
        List <AbstractFile> children = ifile.getChildren();
        if(children == null)   return;
        for (AbstractFile file:children) {
          (5) ;
        }
    }
}
```

该程序运行后输出结果为：

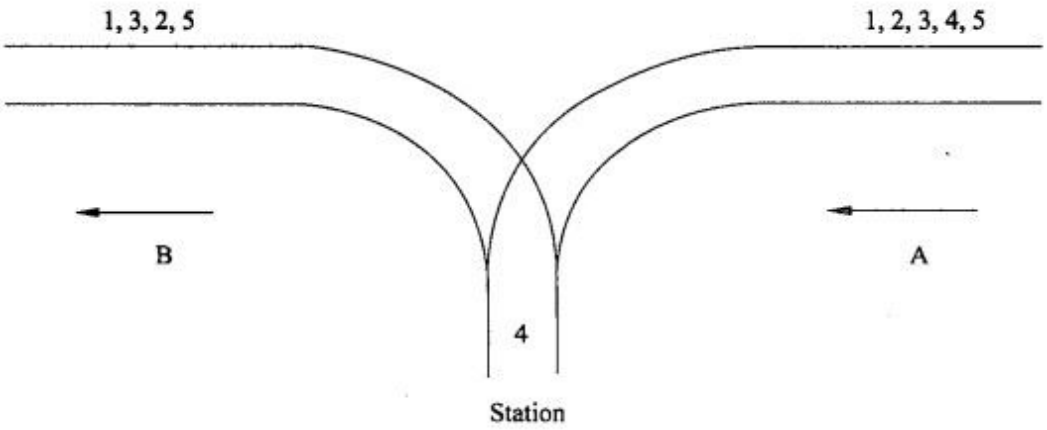
```
c:\
composite
TestComposite.java
Windows
```

试题七 阅读以下说明和C程序，将应填入(n)处的字句写在答题纸的对应栏内。

【说明】

现有n(n<1000)节火车车厢，顺序编号为1，2，3,…，n，按编号连续依次从A方向的铁轨驶入，从B方向铁轨驶出，一旦车厢进入车站(Station)就不能再回到A方向的铁轨上；一旦车厢驶入B方向铁轨就不能再回到车站，如下图所示，其中Station为栈结构，初始为空且最多能停放1000节车厢。

下面的C程序判断能否从B方向驶出预先指定的车厢序列，程序中使用了栈类型STACK，关于栈基本操作的函数原型说明如下：




```
void InitStack(STACK *s)：初始化栈
void Push(STACK *s,int e)：将一个整数压栈，栈中元素数目增 1
void Pop(STACK *s)：栈顶元素出栈，栈中元素数目减 1
int Top(STACK s)：返回非空栈的栈顶元素值，栈中元素数目不变
int IsEmpty(STACK s)：若是空栈则返回 1，否则返回 0
```

问题： **7.1**

【C 程序】

```
#include<stdio.h>

/*此处为栈类型及其基本操作的定义，省略*/

int main( ){
    STACK station;
     int state[1000];
    int n;                                  /*车厢数*/
     int begin, i, j, maxNo;                /*maxNo 为 A 端正待入栈的车厢编号*/
    printf("请输入车厢数： ");
     scanf("%d",&n);
    printf("请输入需要判断的车厢编号序列（以空格分隔）： ");
     if (n < 1)  return -1;
     for (i = 0; i<n; i++)   /* 读入需要驶出的车厢编号序列，存入数组 state[] */
         scanf("%d",&state[i]);
       (1) ; /*初始化栈*/
       maxNo = 1;
        for(i = 0; i < n;){/*检查输出序列中的每个车厢号 state[i]是否能从栈中获取*/
           if ( (2) ){/*当栈不为空时*/
   if (state[i] == Top(station)){ /*栈顶车厢号等于被检查车厢号*/
                printf("%d ",Top(station));
                Pop(&station);   i++;
             }
             else
                if ( (3) ){
                     printf("error\n");
                    return 1;
                 }
                else {
                      begin = (4) ;
                      for(j = begin+1; j<=state[i]; j++) {
                           Push(&station, j);
                      }
                     }
          }
           else {  /*当栈为空时*/
                 begin = maxNo;
                   for(j = begin; j<=state[i]; j++){
                        Push(&station, j);
                   }
                 maxNo = (5) ;
           }
     }
     printf("OK");
         return 0;
    }
```

试题一 答案：　解析：　根据题目中的说明，可以很容易找到与 CCMS 系统进行信息交互的角色有非信用卡客户、信用卡客户以及银行。下面要做的事情是在上图(a)中找到对应的位置。

根据图(a)给出的输入和输出数据流，可知 E1 表示非信用卡客户； E2 表示信用卡客;E3 表示银行。

这道题目主要考查父图与子图的平衡问题。对照上图(a)和(b)可以发现，数据流“信用卡申请表”、“激活请求”、“信用卡交易信息”出现在图(b)中，却没有出现在图(a)中。下一步只要正确地标出这三条数据流的起点和终点就可以了。

数据流的错误主要有与错误的加工相连接、没有经过任何的加工、数据流方向错误等。在图(b)中，并没有出现任何的数据流没有经过加工，那错误就在于与数据流相连接的加工有问题或者数据流方向错误。

这样，可以找两条有错误的数据流“激活请求”和“信用卡申请表”。从图(a)中可知，“激活请求”是从系统流向外部实体 E2 的，而在图(b)中，“激活请求”却出现在两个加工之间。数据流“信用卡申请表”是在问题 2 中补充找到的数辑滩，它应该从外部实体 E1 流向 CCMS 系统。

这道题要求将图(b)中的加工补充完整。加工的名称在说明中 B 经明确给出了：信用卡申请、信用卡激活、信用卡客户信息管理以及交易信息查询。下一步需要根据图 (b)中给出的数据流关系将这 4 个加工对号入座即可。这样可以得到 P1 表示交易信息查询： P2 表示信用卡客户信息管理； P3 表示信用卡激活： P4 表示信用卡申请。

起　点	终　点	名　称
E1	P0	信用卡申请表
E2	P0	激活请求
P0 或 信用卡管理系统 CCMS	E2	信用卡交易信息

注：每条数据流的起点和终点全部答对方可给 1 分

数据流名称	改正后数据流起点	改正后数据流终点
激活请求	E2	P3
信用卡申请表	E1	P4

试题二 答案：　解析：

(a)用户名
(b)邮件号， 发件人地址
(c)邮件号

空(a)分析：根据题意可知邮件客户端系统支持多个用户，用户信息主要包括用户名和用户密码，且系统中的用户名不可重复，“用户名”可以作为用户关系模式主键。地址簿关系模式中与用户关系模式是一个 1 ： 1 的联系，必须将任一方的主键加入另一方， 以建立它们之间的联系，故空(a)处应填写“用户名”。
空(b)分析：根据题意可知邮件号在整个系统内唯一标识一封邮件，故邮件关系模式必须有属性“邮件号”，另外一封邮件需要填写“发件人地址”，故空(b)处应填写“邮件号，发件人地址”。
空(c)分析：根据题意可知邮件和附件是一个 1 :m 的联系，按照 E-R 模型向关系模型的转换规则对于 1 ： m 的联系应将 1 端的主键并入多端，故空(c)处应填写“邮件号”。

(1) 地址簿关系模式的主键为“联系人编号”，外键为“用户名”，因为“用户名” 是参考用户关系模式的“用户名”主键。邮件关系模式的主键为“邮件号”，外键为“发件人地址”或“收件人地址”，因为当用户向其他人发邮件的时候，“发件人地址”是参考邮件账号关系模式的“邮件地址”的主键；当用户收邮件的时候，“收件人地址”是参考邮件账号关系模式的“邮件地址”的主键。附件关系模式的主键为“邮件号，附件号”，外键为“邮件号”，因为该“邮件号”参考邮件关系模式的“邮件号”的主键。
(2) 附件属于弱实体，因为如果没有邮件，附件也就不存在。

（1）1

（2）1

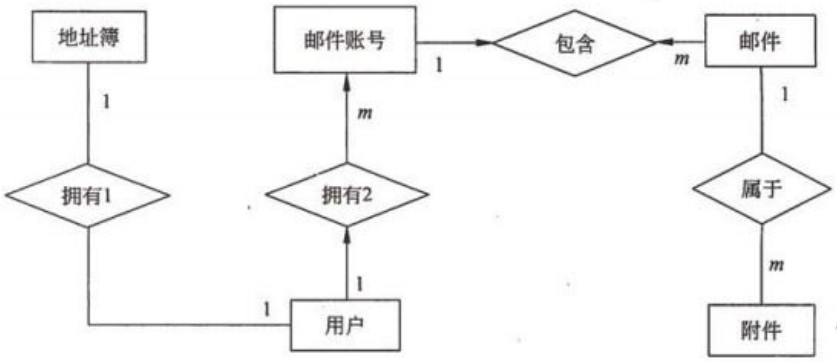
（3）m　　或n或 *

（4）1

（5）m　　或n或 *

两个实体模型之间的联系可以分为三类：一对一联系（1∶1）、一对多联系（1∶n）和多对多联系（m∶n）。

根据题意，地址簿与用户之间应该是一个 1∶1 的联系，空（1）应填 1。电子邮件账号与邮件之间应该是一个 1∶m 的联系，故空（2）和空（3）应分别填写 1 和 m。邮件与附件之间应该是一个 1∶m 的联系，故空（4）和空（5）应分别填写 1 和 m。得到的 E-R 图如下图所示。



（1）

关 系 模 式	主　　键	外　　键
地址簿	联系人编号	用户名
邮件	邮件号	发件人地址或收件人地址
附件	邮件号，附件号	邮件号

（2）附件属于弱实体，因为附件的存在必须以邮件的存在为前提，即附件总是依附于某邮件。

试题三 答案：　解析：　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员工、连册到系统的员工(顾客)、餐厅员工、菜单管理员、送餐员以及工资系统。

由“菜单管理员是餐厅特定员工”以及图中 A2 和图中餐厅员工之间的“是一种”关系可知， A2 为菜单管理员；图中还缺少描述中与工资系统的交互，由“……并发送给工资系统”可知， A1 为工资系统。

考查用例及其和参与者之间的关系时，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。

本题中，由“任何员工都可以查看菜单和今日特价”可知，图中缺少用例查看今日特价，对应参与者是员工；由“系统的顾客是……，注册工资支付、……”可知，图中缺少用例注册工资支付，对应参与者是顾客和工资系统；由“餐厅员工是……，可以进行备餐、生成付费请求……发送给工资系统”可知，图中缺少用例“生成付费请求”，对应的参与者是餐厅员工和工资系统；由“菜单管理员是餐厅特定员工，可以管理菜单”可知，图中缺少用例管理菜单，对应的参与者是菜单管理员。

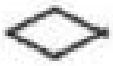
需要注意的是，在注册工资支付所对应的参与者中，虽然没有明确说明要和工资系统交互，但是由“对于注册工资支付的顾客生成付费请求并发送给工资系统”可知，工资支付是由工资系统控制，所以注册也需要和工资系统交互。

泛化关系(一般/特殊关系、继承关系)。泛化关系描述了一个参与者可以完成另一个参与者同样的任务，并可补充额外的角色功能。

参与者之间的关系表示子类型“是一种”父类型，即泛化关系。其中父类型通常是一个抽象泛化的参与者，可以完成子类型可完成的共同行为，每个具体的子类型继承它， 可以完成父类型参与者同样的任务，并可以补充额外的角色功能。

用 例 名	参 与 者
查看当日特价	员工
注册工资支付	顾客和工资系统（或顾客和 A1）
生成付费请求	餐厅员工和工资系统（或餐厅员工和 A1）
管理菜单	菜单管理员（或 A2）

（注：4 行的顺序可以不同，但是每行必须对应）

（1）  　　（2）登录

（3）[redacted] 或 [redacted]　　（4）[redacted] 或 [redacted]

在顾客订餐过程的描述中，在“顾客选菜”之前，图中缺少符号和活动。由说明中顾客“可以订餐（如果未登录，需先登录）”可以判断，在系统“显示菜单和今日特价”之后“顾客选菜”之前，需要判断（判定符号）当前用户身份是否为顾客，如果不是，需先登录；由“……发送 E-mail 给顾客以确认订餐，同时发送相关订餐信息通知给餐厅员工”可知，发送 E-mail 和通知餐厅员工为并行活动，需要在前后有同步条[redacted]（或纵向）。

试题四 答案：　解析：

(5）物品 2 和物品 3

(6）35

(7）15

(8）8

根据问题 1 中给出的伪代码运行该实例，可以很容易得到此 0-1 背包问题的最优解，应该选择物品 2 和物品 3，此时背包的重量为 10+10=20, 获得的价值为 17+18=35 。

若采用穷举法搜索整个解空间，即要构造一颗完全二叉树，此时搜索树的结点数应为 2^4-1= 15, 而采用了上述回溯法，搜索树的结点数仅为 8 个，如上图所示。

（1）k ← 1　　或其等价形式　　（2）cw ← cw + w[k]　或其等价形式

（3）k ← k–1　　或其等价形式　　（4）k ← k+1　　或其等价形式

本题考查的是用回溯法求解 0-1 背包问题。回溯法有两类算法框架：非递归形式和递归形式，本题采用非递归形式表示。理解回溯法的基本思想和这两类算法框架是正确解答本题的根本要求。

回溯法从第一项物品开始考虑是否应该装入背包中，因此当前考虑的物品编号 k 从 1 开始，即 k ← 1。然后逐项往后检查，若能全部放入背包则将该项放入背包，此时背包的重量应该是当前的重量加上当前考虑物品的重量，即 cw ← cw + w[k]，当然背包中物品的价值也为当前的价值加上当前考虑物品的价值。若已经考虑完了所有的物品，则得到一个解，判断该解是否为当前最优，若为最优，则将该解的信息放入变量 fp、fw 和 X 中。若还没有考虑完所有的物品，意味着有些物品不能放入背包，此时先判断若不将当前的物品放入背包中，则其余物品放入背包是否可能得到比当前最优解更优的解，若得不到则回溯；否则继续考虑其余的物品。

试题五 答案：　解析：

(1) this->name

(2) list*

(3) NULL

(4) this->name

(5) &childList

本题考查基本面向对象设计中设计模式的运用能力。

组合设计模式主要是表达整体和部分的关系，并且对整体和部分对象的使用无差别。题目中 AbstractFile 是 File 类和 Folder 类的父类，它抽象了两个类的共有属性和行为，在后续 main 方法的使用中，不论是 File 对象还是 Folder 对象，都可被当作 AbstractFile 对象来使用。另外，由于 Folder 对象可以聚合其他的 Folder 对象和 File 对象，等价于 Folder 对象可以聚合另一个 AbstractFile 对象。

在类 File 和类 Folder 的构造函数中都需要记录文件或目录的名称，因此空(1) 和 空(4) 处主要是设置对象的名称。因为 File 对象不再聚合其他的对象，所以 File 对象没有孩子节点，因此，空(3) 处应该返回 NULL 。 getChildren() 方法继承自 AbstractFile 类，因此其返回类型也应保持一致。对于空(5) ，要求返回 Folder 对象的孩子对象，因此返回其成员 childList 的地址。

试题六 答案：　解析：　(1) abstract

(2) null

(3) List

(4) childList

(5) printTree(file)

本题考查基本面向对象设计中设计模式的运用能力。

组合设计模式主要是表达整体和部分的关系，并且对整体和部分对象的使用无差别。题目中 AbstractFile 是 File 类和 Folder 类的父类，它抽象了两个类的共有属性和行为，在后续 main 方法的使用中，不论是 File 对象还是 Folder 对象，都可被当作 AbstractFile 对象来使用。另外，由于 Folder 对象可以聚合其他的 Folder 对象和 File 对象，等价于 Folder 对象可以聚合另一个 AbstractFile 对象。

题目中 AbstractFile 类应该为抽象类，因此其修饰符应该包括 abstract，空(2) 处返回 File 类对象的孩子，但 File 类对象没有孩子节点，因此其返回值应该为 NULL。

getChildren 方法是继承自抽象父类 AbstractFile,所以其返回类型应该和父类的定义保持一致，空(4)处返回存储孩子节点的集合对象 childList。该程序的运行能够打印出文件目录树，因此空(5) 处应该为打印方法的调用。

试题七 答案：　解析：　(1) InitStack(&station)

(2) !IsEmpty(station)

(3) state[i]

(4) Top(station)

(5) j，或 state[i]+l

本题考查栈数据结构的应用和C程序设计基本能力。

栈的运算特点是后进先出。在本题中，入栈序列为1、2、…、n–1、n，出栈序列保存在state[]数组中，state[0]记录出栈序列的第1个元素，state[1]记录出栈序列的第2个元素，依此类推。程序采用模拟元素入栈和出栈的操作过程来判断出栈序列是否恰当。需要注意的是，对于栈，应用时不一定是所有元素先入栈后，再逐个进行出栈操作，也不一定是进入一个元素紧接着就出来一个元素，而是栈不满且输入序列还有元素待进入就可以进栈，只要栈不空，栈顶元素就可以出栈，从而使得唯一的一个入栈序列可以得到多个出栈序列。当然，在栈中有多个元素时，只能让栈顶的元素先出栈，栈中其他的元素才能从顶到底逐个出栈。本题中入栈序列和出栈序列的元素为车厢号。

空（1）处对栈进行初始化，根据题干中关于栈基本操作的说明，调用InitStack初始化栈，由于形参是指针参数，因此实参应为地址量，即应填入“InitStack(&station)”。

当栈不空时，就可以令栈顶车厢出栈，空（2）处应填入“!IsEmpty(station)”。

栈顶车厢号以Top(station)表示，若栈顶车厢号等于出栈序列的当前车厢号state[i]，说明截至到目前为止，出栈子序列state[0]～state[i]可经过栈运算获得。由于进栈时小编号的车厢先于大编号的车厢进入栈中，因此若栈顶车厢号大于出栈序列的当前车厢号state[i]，则对于state[i]记录的车厢，若它还在栈中，则此时无法出栈，因为它不在栈顶，所以出错；若它已先于当前的栈顶车厢出栈，则与目前的出栈序列不匹配，仍然出错，因此空（3）处应填入“state[i] < Top(station)”。

若栈顶车厢号小于出栈序列的当前车厢号state[i]，则说明state[i]记录的车厢还没有进入栈中，因此从入栈序列（A端）正待进入的车厢（即比栈顶车厢号正好大1）开始，直到state[i]记录的车厢号为止，这些车厢应依次进入栈中。程序中用以下代码实现此操作：

```
for(j = begin+1; j<=state[i]; j++) {
            Push(&station, j);
}
```

显然，begin应获取栈顶车厢号的值，即空（4）处应填入“Top(station)”。

还有一种情况，就是待考查的出栈序列还没有结束而栈空了，则说明需要处理入栈序列，使其车厢入栈。程序中用maxNo表示A端正待入栈的车厢编号，相应的处理如下面代码所示：

```
begin = maxNo;
            for(j = begin; j<=state[i]; j++){
                Push(&station, j);
            }
```

接下来，A端正待入栈的车厢编号等于j或state[i]+1，即空（5）处应填入j或“state[i]+1”。

如果驶出的车厢编号序列是经由栈获得的，则程序运行时输出该序列及字符串“OK”，否则输出“error”而结束。