

软件设计师

2015年下半年试题

本试卷为：**样式1**

样式1：适用于模拟考试，所有答案在最后面。

样式2：适用于复习，每道题的题目和答案在一起。

本试卷由**跨步软考**提供

我们目前提供的免费服务有：

- 手机APP刷题
- 网页版刷题
- 真题pdf版下载
- 视频课程下载
- 其他资料下载

更多免费服务请访问我们的官网：<https://kuabu.xyz>

你也可以关注我们的微信公众号：**跨步软考**

如果您发现试题有错误，您可以通过以下方式联系我们

-
- 客服邮箱：kuabu@outlook.com
- 您也可以在微信公众号后台留言

本文档所有权归**跨步软考**(kuabu.xyz)，您可以传播甚至修改本文档，但是必须标明出自“**跨步软考 (kuabu.xyz)**”

上午综合试卷

第1题: CPU是在 (1) 结束时响应DMA请求的。

- A. 一条指令执行
- B. 一段程序
- C. 一个时钟周期
- D. 一个总线周期

第2题: 虚拟存储体系由 (2) 两级存储器构成。

- A. 主存-辅存
- B. 寄存器-Cache
- C. 寄存器-主存
- D. Cache-主存

第3题: 浮点数能够表示的数的范围是由其 (3) 的位数决定的。

- A. 尾数
- B. 阶码
- C. 数符
- D. 阶符

第4题: 在机器指令的地址字段中, 直接指出操作数本身的寻址方式称为 (4)。

- A. 隐含寻址
- B. 寄存器寻址
- C. 立即寻址
- D. 直接寻址

第5题: 内存按字节编址从B3000H到DABFFH的区域其存储容量为 (5)。

- A. 123KB
- B. 159KB
- C. 163KB

D. 194KB

第6题: CISC是 (6) 的简称。

- A. 复杂指令系统计算机
- B. 超大规模集成电路
- C. 精简指令系统计算机
- D. 超长指令字

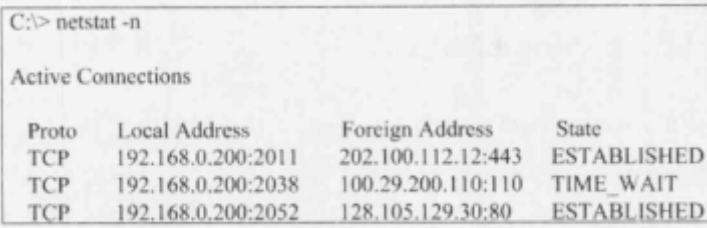
第7题: (7) 不属于主动攻击。

- A. 流量分析
- B. 重放
- C. IP地址欺骗
- D. 拒绝服务

第8题: 防火墙不具备 (8) 功能。

- A. 记录访问过程
- B. 查毒
- C. 包过滤
- D. 代理

第9题: 根据下图所示的输出信息, 可以确定的是: (9)



Active Connections			
Proto	Local Address	Foreign Address	State
TCP	192.168.0.200:2011	202.100.112.12:443	ESTABLISHED
TCP	192.168.0.200:2038	100.29.200.110:110	TIME_WAIT
TCP	192.168.0.200:2052	128.105.129.30:80	ESTABLISHED

- A. 本地主机正在使用的端口号是公共端口号
- B. 192.168.0.200正在与128.105.129.30建立连接
- C. 本地主机与202.100.112.12建立了安全连接
- D. 本地主机正在与100.29.200.110建立连接

第10题: 以下著作权权利中, (10) 的保护期受时间限制。

- A. 署名权
- B. 修改权
- C. 发表权
- D. 保护作品完整权

第11题：王某在其公司独立承担了某综合信息管理系统软件的程序设计工作。该系统交付用户、投入试运行后，王某辞职，并带走了该综合信息管理系统源程序，拒不交还公司。王某认为，综合信息管理系统源程序是他独立完成的：他是综合信息管理系统源程序的软件著作人。王某的行为（11）。

- A. 侵犯了公司的软件著作权
- B. 未侵犯公司的软件著作权
- C. 侵犯了公司的商业秘密权
- D. 不涉及侵犯公司的软件著作权

第12题：声音（音频）信号的一个基本参数是频率；它是指声波每秒钟变化的次数，用Hz表示。人耳能听到的音频信号的频率范围是（12）。

- A. 0Hz ~ 20KHz
- B. 0Hz ~ 200KHz
- C. 20Hz ~ 20KHz
- D. 20Hz ~ 200KHz

第13题：颜色深度是表达图像中单个像素的颜色或灰度所占的位数(bit)。若每个像素具有8位的颜色深度，则可表示（13）种不同的颜色。

- A. 8
- B. 64
- C. 256
- D. 512

第14题：视觉上的颜色可用亮度、色调和饱和度三个特征来描述。其中饱和度是指颜色的（14）。

- A. 种数
- B. 纯度

- C. 感觉
- D. 存储量

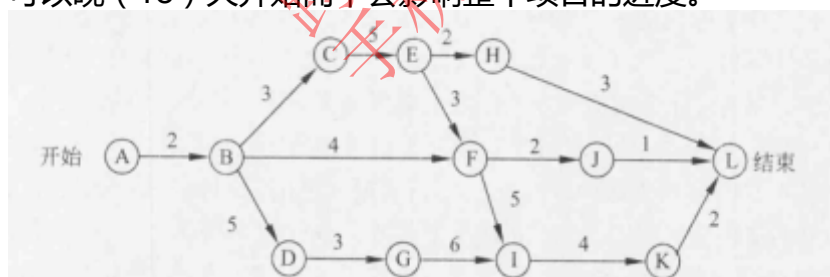
第15题：若用户需求不清晰且经常发生变化，但系统规模不太大且不太复杂，则最适宜采用（15）开发方法，对于数据处理领域的问题，若系统规模不太大且不太复杂，需求变化也不大，则最适宜采用（16）开发方法。

- A. 结构化
- B. Jackson
- C. 原型化
- D. 面向对象

第16题：若用户需求不清晰且经常发生变化，但系统规模不太大且不太复杂，则最适宜采用（15）开发方法，对于数据处理领域的问题，若系统规模不太大且不太复杂，需求变化也不大，则最适宜采用（16）开发方法。

- A. 结构化
- B. Jackson
- C. 原型化
- D. 面向对象

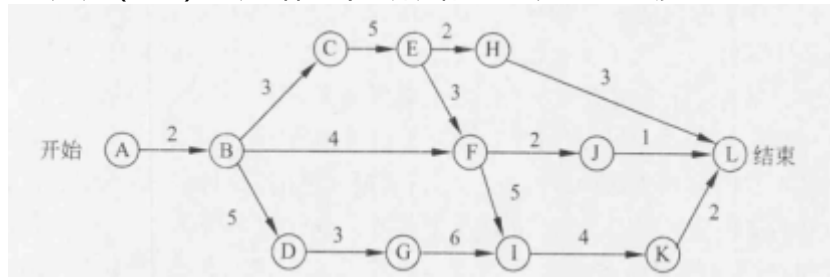
第17题：某软件项目的活动图如下图所示，其中顶点表示项目里程碑，连接顶点的边表示活动，边上的数字表示该活动所需的天数，则完成该项目的最少时间为（17）天。活动BD最多可以晚（18）天开始而不会影响整个项目的进度。



- A. 9
- B. 15
- C. 22
- D. 24

第18题：某软件项目的活动图如下图所示，其中顶点表示项目里程碑，连接顶点的边表示活动，边上的数字表示该活动所需的天数，则完成该项目的最少时间为（17）天。活动BD最多

可以晚 (18) 天开始而不会影响整个项目的进度。



- A. 2
- B. 3
- C. 5
- D. 9

第19题：以下关于软件项目管理中人员管理的叙述，正确的是 (19)。

- A. 项目组成员的工作风格也应该作为组织团队时要考虑的一个要素
- B. 鼓励团队的每个成员充分地参与开发过程的所有阶段
- C. 仅根据开发人员的能力来组织开发团队
- D. 若项目进度滞后于计划，则增加开发人员一定可以加快开发进度

第20题：编译器和解释器是两种基本的高级语言处理程序。编译器对高级语言源程序的处理过程可以划分为词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成等阶段，其中， (20) 并不是每个编译器都必需的，与编译器相比，解释器 (21)。

- A. 词法分析和语法分析
- B. 语义分析和中间代码生成
- C. 中间代码生成和代码优化
- D. 代码优化和目标代码生成

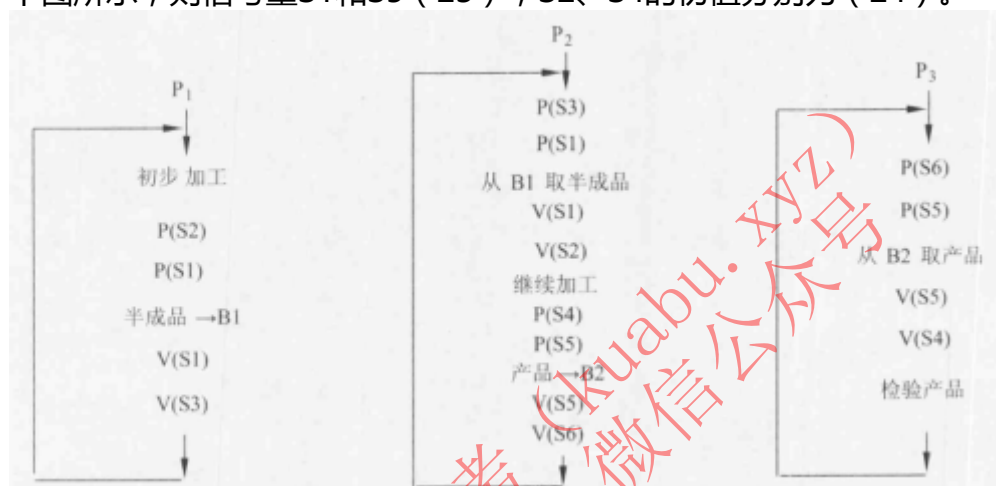
第21题：编译器和解释器是两种基本的高级语言处理程序。编译器对高级语言源程序的处理过程可以划分为词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成等阶段，其中， (20) 并不是每个编译器都必需的，与编译器相比，解释器 (21)。

- A. 不参与运行控制，程序执行的速度慢
- B. 参与运行控制，程序执行的速度慢
- C. 参与运行控制，程序执行的速度快
- D. 不参与运行控制，程序执行的速度快

第22题：表达式采用逆波兰式表示时，利用（22）进行求值。

- A. 栈
- B. 队列
- C. 符号表
- D. 散列表

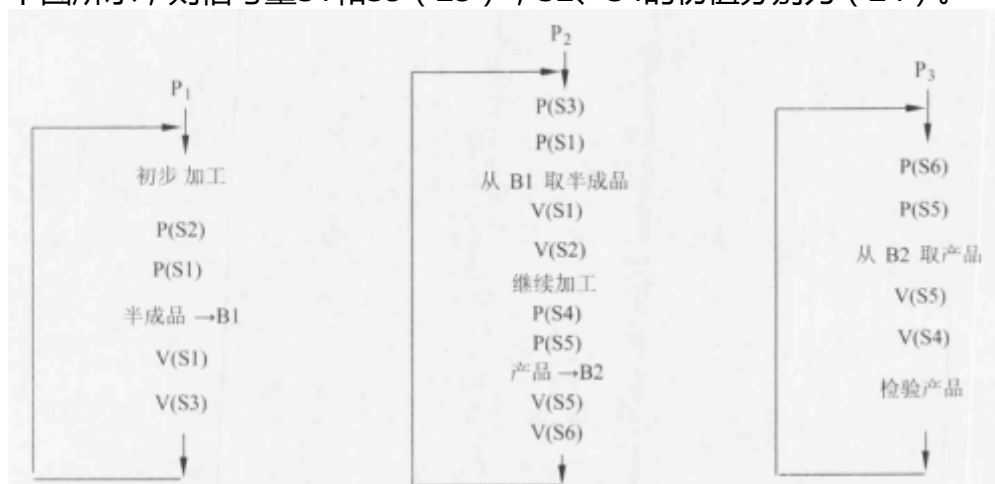
第23题：某企业的生产流水线上有2名工人P1和P2，1名检验员P3。P1将初步加工的半成品放入半成品箱B1；P2从半成品箱B1取出继续加工，加工好的产品放入成品箱B2；P3从成品箱B2去除产品校验。假设B1可存放n件半成品，B2可存放m件产品，并设置6个信号量S1、S2、S3、S4、S5和S6，且S3和S6的初值都为0。采用PV操作实现P1、P2和P3的同步模型如下图所示，则信号量S1和S5（23）；S2、S4的初值分别为（24）。



- A. 分别为同步信号量和互斥信号量，初值分别为0和1
- B. 都是同步信号量，其初值分别为0和0
- C. 都是互斥信号量，其初值分别为1和1
- D. 都是互斥信号量，其初值分别为0和1

第24题：某企业的生产流水线上有2名工人P1和P2，1名检验员P3。P1将初步加工的半成品放入半成品箱B1；P2从半成品箱B1取出继续加工，加工好的产品放入成品箱B2；P3从成品箱B2去除产品校验。假设B1可存放n件半成品，B2可存放m件产品，并设置6个信号量S1、S2、S3、S4、S5和S6，且S3和S6的初值都为0。采用PV操作实现P1、P2和P3的同步模型如

下图所示, 则信号量S1和S5 (23) ; S2、S4的初值分别为 (24) 。



- A. n、0
- B. m、0
- C. m、n
- D. n、m

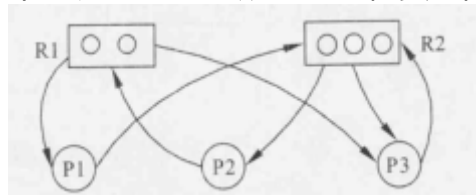
第25题：假设磁盘块与缓冲区大小相同，每个盘块读入缓冲区的时间为 $15\mu s$ ，由缓冲区送至用户区的时间是 $5\mu s$ ，在用户区内系统对每块数据的处理时间为 $1\mu s$ ，若用户需要将大小为10个磁盘块的Doc1文件逐块从磁盘读入缓冲区，并送至用户区进行处理，那么采用单缓冲区需要花费的时间为 (25) μs ；采用双缓冲区需要花费的时间为 (26) μs 。

- A. 150
- B. 151
- C. 156
- D. 201

第26题：假设磁盘块与缓冲区大小相同，每个盘块读入缓冲区的时间为 $15\mu s$ ，由缓冲区送至用户区的时间是 $5\mu s$ ，在用户区内系统对每块数据的处理时间为 $1\mu s$ ，若用户需要将大小为10个磁盘块的Doc1文件逐块从磁盘读入缓冲区，并送至用户区进行处理，那么采用单缓冲区需要花费的时间为 (25) μs ；采用双缓冲区需要花费的时间为 (26) μs 。

- A. 150
- B. 151
- C. 156
- D. 201

第27题：在如下所示的进程资源图中，（27）。



- A. P1、P2、P3都是非阻塞节点，该图可以化简，所以是非死锁的
- B. P1、P2、P3都是阻塞节点，该图不可以化简，所以是死锁的
- C. P1、P2是非阻塞节点，P3是阻塞节点，该图不可以化简，所以是死锁的
- D. P2是阻塞节点，P1、P3是非阻塞节点，该图可以化简，所以是非死锁的

第28题：在支持多线程的操作系统中，假设进程P创建了若干个线程，那么（28）是不能被这些线程共享的。

- A. 该进程中打开的文件
- B. 该进程的代码段
- C. 该进程中某线程的栈指针
- D. 该进程的全局变量

第29题：某开发小组欲开发一个超大规模软件：使用通信卫星，在订阅者中提供、监视和控制移动电话通信，则最不宜采用（29）过程模型。

- A. 瀑布
- B. 原型
- C. 螺旋
- D. 喷泉

第30题：（30）开发过程模型以用户需求为动力，以对象为驱动，适合于面向对象的开发方法。

- A. 瀑布
- B. 原型
- C. 螺旋
- D. 喷泉

第31题：在ISO/IEC软件质量模型中，易使用性的子特性不包括（31）。

- A. 易理解性
- B. 易学性
- C. 易操作性
- D. 易分析性

第32题：在进行子系统结构设计时，需要确定划分后的子系统模块结构，并画出模块结构图。该过程不需要考虑（32）。

- A. 每个子系统如何划分成多个模块
- B. 每个子系统采用何种数据结构和核心算法
- C. 如何确定子系统之间、模块之间传送的数据及其调用关系
- D. 如何评价并改进模块结构的质量

第33题：数据流图中某个加工的一组动作依赖于多个逻辑条件的取值，则用（33）能够清楚地表示复杂的条件组合与应做的动作之间的对应关系。

- A. 流程图
- B. NS盒图
- C. 形式语言
- D. 决策树

第34题：根据软件过程活动对软件工具进行分类，则逆向工程工具属于（34）工具。

- A. 软件开发
- B. 软件维护
- C. 软件管理
- D. 软件支持

第35题：若用白盒测试方法测试以下代码，并满足条件覆盖，则至少需要（35）个测试用例。采用McCabe度量法算出该程序的环路复杂性为（36）。

```
int find_max(int i, int j, int k){  
    int max;  
    if (i > j) then  
        if (i > k) then max = i;  
        else max = k;  
    else if (j > k) max = j;  
    else max = k;  
    return max;  
}
```

- A. 3
- B. 4
- C. 5
- D. 6

第36题：若用白盒测试方法测试以下代码，并满足条件覆盖，则至少需要（35）个测试用例。采用McCabe度量法算出该程序的环路复杂性为（36）。

```
int find_max(int i, int j, int k){  
    int max;  
    if (i > j) then  
        if (i > k) then max = i;  
        else max = k;  
    else if (j > k) max = j;  
    else max = k;  
    return max;  
}
```

- A. 1
- B. 2
- C. 3
- D. 4

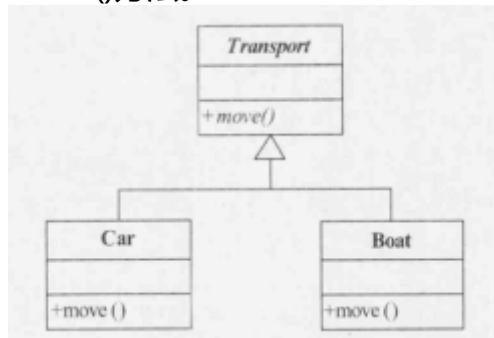
第37题：在面向对象的系统中，对象是运行时实体，其组成部分不包括（37）；一个类定义了一组大体相似的对象，这些对象共享（38）。

- A. 消息
- B. 行为（操作）
- C. 对象名
- D. 状态

第38题：在面向对象的系统中，对象是运行时实体，其组成部分不包括（37）；一个类定义了一组大体相似的对象，这些对象共享（38）。

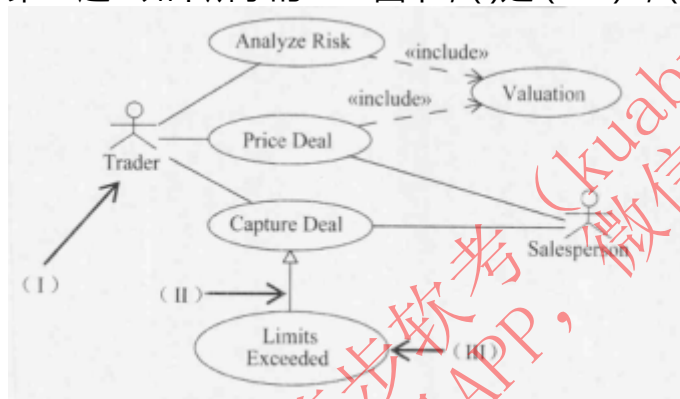
- A. 属性和状态
- B. 对象名和状态
- C. 行为和多重度
- D. 属性和行为

第39题：如下所示的UML类图中，Car和Boat类中的move()方法（39）了Transport类中的move()方法。



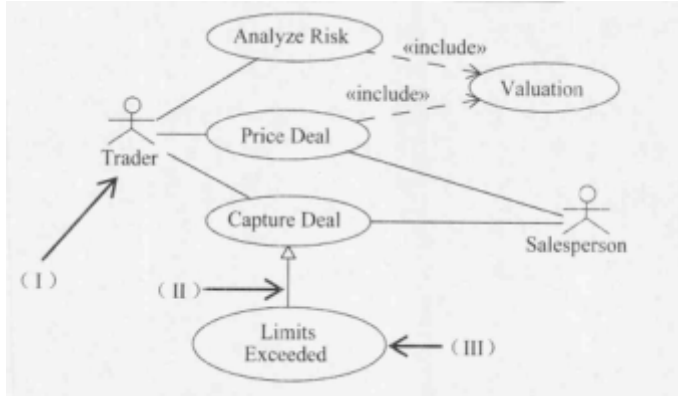
- A. 继承
- B. 覆盖 (重置)
- C. 重载
- D. 聚合

第40题：如下所示的UML图中，(I)是（40），(II)是（41），(III)是（42）。



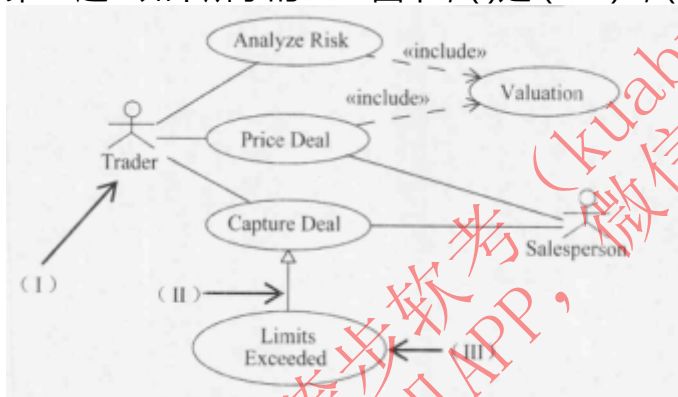
- A. 参与者
- B. 用例
- C. 泛化关系
- D. 包含关系

第41题：如下所示的UML图中，(I)是 (40)，(II)是 (41)，(III)是 (42)。



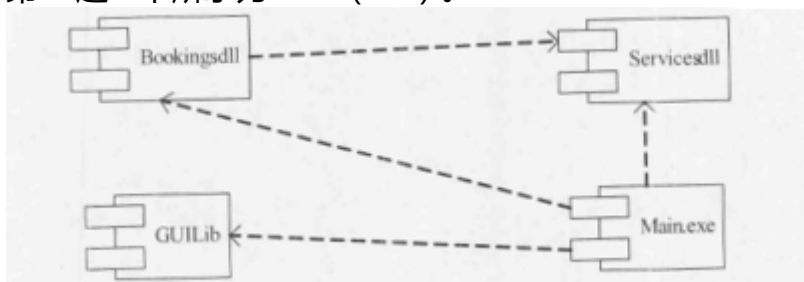
- A. 参与者
- B. 用例
- C. 泛化关系
- D. 包含关系

第42题：如下所示的UML图中，(I)是 (40)，(II)是 (41)，(III)是 (42)。



- A. 参与者
- B. 用例
- C. 泛化关系
- D. 包含关系

第43题：下所示为UML (43)。



- A. 类图

- B. 部署图
- C. 组件图
- D. 网络图

第44题：以下关于Singleton（单例）设计模式的叙述中，不正确的是（44）。

- A. 单例模式是创建型模式
- B. 单例模式保证一个类仅有一个实例
- C. 单例类提供一个访问唯一实例的全局访问点
- D. 单例类提供一个创建一系列相关或相互依赖对象的接口

第45题：（45）设计模式能够动态地给一个对象添加一些额外的职责而无需修改此对象的结构；（46）设计模式定义一个用于创建对象的接口，让子类决定实例化哪一个类；欲使一个后端数据模型能够被多个前端用户界面连接，采用（47）模式最适合。

- A. 组合（Composite）
- B. 外观(Facade)
- C. 享元（Flyweight）
- D. 装饰器(Decorator)

第46题：（45）设计模式能够动态地给一个对象添加一些额外的职责而无需修改此对象的结构；（46）设计模式定义一个用于创建对象的接口，让子类决定实例化哪一个类；欲使一个后端数据模型能够被多个前端用户界面连接，采用（47）模式最适合。

- A. 工厂方法（Factory Method）
- B. 享元（Flyweight）
- C. 观察者（Observer）
- D. 中介者(Mediator)

第47题：（45）设计模式能够动态地给一个对象添加一些额外的职责而无需修改此对象的结构；（46）设计模式定义一个用于创建对象的接口，让子类决定实例化哪一个类；欲使一个后端数据模型能够被多个前端用户界面连接，采用（47）模式最适合。

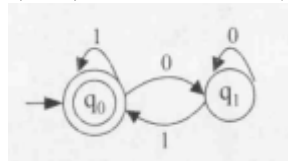
- A. 装饰器(Decorator)
- B. 享元（Flyweight）
- C. 观察者(Observer)

D. 中介者(Mediator)

第48题：某程序运行时陷入死循环，则可能的原因是程序中存在（48）。

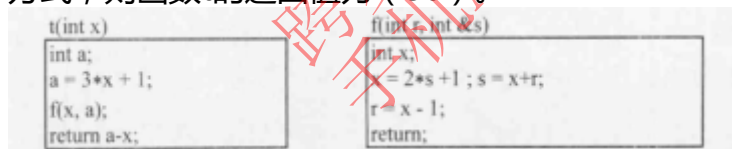
- A. 词法错误
- B. 语法错误
- C. 动态的语义错误
- D. 静态的语义错误

第49题：某非确定的有限自动机(NFA)的状态转换图如下图所示（q0既是初态也是终态）。以下关于该NFA的叙述中，正确的是（49）。



- A. 其可识别的0、1序列的长度为偶数
- B. 其可识别的0、1序列中0与1的个数相同
- C. 其可识别的非空0、1序列中开头和结尾字符都是0
- D. 其可识别的非空0、1序列中结尾字符是1

第50题：函数t()、f()的定义如下所示，若调用函数t时传递给x的值为5，并且调用函数F()时，第一个参数采用传值（call by value）方式，第二个参数采用传引用(call by reference)方式，则函数t的返回值为（50）。



- A. 33
- B. 22
- C. 11
- D. 负数

第51题：数据库系统通常采用三级模式结构：外模式、模式和内模式。这三级模式分别对应数据库的（51）。

- A. 基本表、存储文件和视图
- B. 视图、基本表和存储文件

C. 基本表、视图和存储文件

D. 视图、存储文件和基本表

第52题：在数据库逻辑设计阶段，若实体中存在多值属性，那么将E-R图转换为关系模式时，(52)，得到的关系模式属于4NF。

- A. 将所有多值属性组成一个关系模式
- B. 使多值属性不在关系模式中出现
- C. 将实体的码分别和每个多值属性独立构成一个关系模式
- D. 将多值属性和其它属性一起构成该实体对应的关系模式

第53题：在分布式数据库中有分片透明、复制透明、位置透明和逻辑透明等基本概念，其中：(53)是指局部数据模型透明，即用户或应用程序无需知道局部使用的是哪种数据模型；(54)是指用户或应用程序不需要知道逻辑上访问的表具体是如何分块存储的。

- A. 分片透明
- B. 复制透明
- C. 位置透明
- D. 逻辑透明

第54题：在分布式数据库中有分片透明、复制透明、位置透明和逻辑透明等基本概念，其中：(53)是指局部数据模型透明，即用户或应用程序无需知道局部使用的是哪种数据模型；(54)是指用户或应用程序不需要知道逻辑上访问的表具体是如何分块存储的。

- A. 分片透明
- B. 复制透明
- C. 位置透明
- D. 逻辑透明

第55题：设有关系模式R (A1,A2,A3,A4,A5,A6)，其中：函数依赖集F={A1→A2,A1A3→A4,A5A6→A1,A2A5→A6,A3A5→A6},则(55)是关系模式R的一个主键，R规范化程度最高达到(56)。

- A. A1A4
- B. A2A4
- C. A3A5
- D. A4A5

第56题：设有关系模式R (A1,A2,A3,A4,A5,A6) , 其中：函数依赖集F={A1→A2,A1A3→A4,A5A6→A1,A2A5→A6,A3A5→A6},则 (55) 是关系模式R的一个主键, R规范化程度最高达到 (56) 。

- A. 1NF
- B. 2NF
- C. 3NF
- D. BCNF

第57题：对于一个长度为n(n>1)且元素互异的序列, 每其所有元素依次通过一个初始为空的栈后, 再通过一个初始为空的队列。假设队列和栈的容量都足够大, 且只要栈非空就可以进行出栈操作, 只要队列非空就可以进行出队操作, 那么以下叙述中, 正确的是 (57) 。

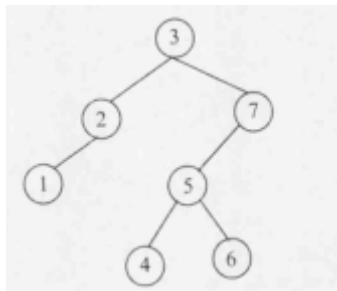
- A. 出队序列和出栈序一定互为逆序
- B. 出队序列和出栈序列一定相同
- C. 入栈序列与入队序列一定相同
- D. 入栈序列与入队序列一定互为逆序

第58题：设某n阶三对角矩阵 $A_{n \times n}$ 的示意图如下图所示。若将该三对角矩阵的非零元素按行存储在一维数组B[k] ($1 \leq k \leq 3 \cdot n - 2$) 中, 则k与i、j的对应关系是 (58) 。

$$A_{n \times n} = \begin{bmatrix} a_{1,1} & a_{1,2} & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & \\ & a_{3,2} & a_{3,3} & a_{3,4} & \\ & & \dots & \dots & \dots \\ 0 & & & \dots & \dots \\ & & & & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

- A. $k=2i+j-2$
- B. $k=2i-j+2$
- C. $k=3i+j-1$
- D. $k=3i-j+2$

第59题：对于非空的二叉树, 设D代表根结点, L代表根结点的左子树R代表根结点的右子树。若对下图所示的二叉树进行遍历后的结点序列为7 6 5 4 3 2 1, 则遍历方式是 (59) 。



- A. LRD
- B. DRL
- C. RLD
- D. RDL

第60题：在55个互异元素构成的有序表A[1..55]中进行折半查找（或二分查找，向下取整）。若需要找的元素等于A[19]，则在查找过程中参与比较的元素依次为（60）、A[19]。

- A. A[28]、A[30]、A[15]、A[20]
- B. A[28]、A[14]、A[21]、A[17]
- C. A[28]、A[15]、A[22]、A[18]
- D. A[28]、A[18]、A[22]、A[20]

第61题：设一个包含n个顶点、e条弧的简单有向图采用邻接矩阵存储结构（即矩阵元素A[i][j]等于1或0，分别表示顶点i与顶点j之间有弧或无弧），则该矩阵的非零元素数目为（61）。

- A. e
- B. 2e
- C. n-e
- D. n+e

第62题：已知算法A的运行时间函数为 $T(n)=8T(n/2)+n^2$ ，其中n表示问题的规模，则该算法的时间复杂度为（62）。另已知算法B的运行时间函数为 $T(n)=XT(n/4)+n^2$ ，其中n表示问题的规模。对充分大的n，若要算法B比算法A快，则X的最大值为（63）。

- A. $\theta(n)$
- B. $\theta(n \lg n)$
- C. $\theta(n^2)$
- D. $\theta(n^3)$

第63题：已知算法A的运行时间函数为 $T(n)=8T(n/2)+n^2$ ，其中n表示问题的规模，则该算法的时间复杂度为（62）。另已知算法B的运行时间函数为 $T(n)=XT(n/4)+n^2$ ，其中n表示问题的规模。对充分大的n，若要算法B比算法A快，则X的最大值为（63）。

- A. 15
- B. 17
- C. 63
- D. 65

第64题：在某应用中，需要先排序一组大规模的记录，其关键字为整数。若这组记录的关键字基本上有序，则适宜采用（64）排序算法。若这组记录的关键字的取值均在0到9之间（含），则适宜采用（65）排序算法。

- A. 插入
- B. 归并
- C. 快速
- D. 计数

第65题：在某应用中，需要先排序一组大规模的记录，其关键字为整数。若这组记录的关键字基本上有序，则适宜采用（64）排序算法。若这组记录的关键字的取值均在0到9之间（含），则适宜采用（65）排序算法。

- A. 插入
- B. 归并
- C. 快速
- D. 基数

第66题：集线器与网桥的区别是：（66）。

- A. 集线器不能检测发送冲突，而网桥可以检测冲突
- B. 集线器是物理层设备，而网桥是数据链路层设备
- C. 网桥只有两个端口，而集线器是一种多端口网桥
- D. 网桥是物理层设备，而集线器是数据链路层设备

第67题：POP3协议采用（67）模式，客户端代理与POP3服务器通过建立TCP连接来传送数据。

- A. Browser/Server

B. Client/Server

C. Peer to Peer

D. Peer to Server

第68题: TCP使用的流量控制协议是 (68)。

A. 固定大小的滑动窗口协议

B. 后退N帧的ARQ协议

C. 可变大小的滑动窗口协议

D. 停等协议

第69题: 以下4种路由中, (69) 路由的子网掩码是255.255.255.255。

A. 远程网络

B. 静态

C. 默认

D. 主机

第70题: 以下关于层次化局域网模型中核心层的叙述, 正确的是 (70)。

A. 为了保障安全性, 对分组要进行有效性检查

B. 将分组从一个区域高速地转发到另一个区域

C. 由多台二、三层交换机组成

D. 提供多条路径来缓解通信瓶颈

第71题: In a world where it seems we already have too much to do, and too many things to think about, it seems the last thing we need is something new that we have to learn.

But use cases do solve a problem with requirements: with (71) declarative requirements it's hard to describe steps and sequences of events.

Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful. As simple as this sounds, this is important. When confronted only with a pile of requirements, it's often (72) to make sense of what the authors of the requirements really wanted the system to do. In the preceding example, use cases reduce the ambiguity of the requirements by specifying exactly when and under what conditions certain behavior occurs; as such, the sequence of the behaviors can be regarded as a requirement. Use cases are particularly well suited to capture approaches. Although this may sound simple, the

fact is that (73) requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the (74) of the system" s behavior. Use cases are a simple yet powerful way to express the behavior of the system in way that all stakeholders can easily understand.

But, like anything, use cases come with their own problems, and as useful as they are, they can be (75) . The result is something that is as bad, if not worse, than the original problem. Therein it" s important to utilize use cases effectively without creating a greater problem than the one you started with.

- A. plenty
- B. loose
- C. extra
- D. strict

第72题 : In a world where it seems we already have too much to do, and too many things to think about, it seems the last thing we need is something new that we have to learn.

But use cases do solve a problem with requirements; with (71) declarative requirements it" s hard to describe steps and sequences of events.

Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful. As simple as this sounds, this is important. When confronted only with a pile of requirements, it" s often (72) to make sense of what the authors of the requirements really wanted the system to do. In the preceding example, use cases reduce the ambiguity of the requirements by specifying exactly when and under what conditions certain behavior occurs; as such, the sequence of the behaviors can be regarded as a requirement. Use cases are particularly well suited to capture approaches. Although this may sound simple, the fact is that (73) requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the (74) of the system" s behavior. Use cases are a simple yet powerful way to express the behavior of the system in way that all stakeholders can easily understand.

But, like anything, use cases come with their own problems, and as useful as they are, they can be (75) . The result is something that is as bad, if not worse, than the original problem. Therein it" s important to utilize use cases effectively without creating a greater problem than the one you started with.

- A. impossible
- B. possible
- C. sensible
- D. practical

第73题: In a world where it seems we already have too much to do, and too many things to think about, it seems the last thing we need is something new that we have to learn.

But use cases do solve a problem with requirements: with (71) declarative requirements it's hard to describe steps and sequences of events.

Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful. As simple as this sounds, this is important. When confronted only with a pile of requirements, it's often (72) to make sense of what the authors of the requirements really wanted the system to do. In the preceding example, use cases reduce the ambiguity of the requirements by specifying exactly when and under what conditions certain behavior occurs; as such, the sequence of the behaviors can be regarded as a requirement. Use cases are particularly well suited to capture approaches. Although this may sound simple, the fact is that (73) requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the (74) of the system's behavior. Use cases are a simple yet powerful way to express the behavior of the system in way that all stakeholders can easily understand.

But, like anything, use cases come with their own problems, and as useful as they are, they can be (75). The result is something that is as bad, if not worse, than the original problem. Therein it's important to utilize use cases effectively without creating a greater problem than the one you started with.

- A. modern
- B. conventional
- C. different
- D. formal

第74题: In a world where it seems we already have too much to do, and too many things to think about, it seems the last thing we need is something new that we have to learn.

But use cases do solve a problem with requirements: with (71) declarative requirements it's hard to describe steps and sequences of events.

Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful. As simple as this sounds, this is important. When confronted only with a pile of requirements, it's often (72) to make sense of what the authors of the requirements really wanted the system to do. In the preceding example, use cases reduce the ambiguity of the requirements by specifying exactly when and under what conditions certain behavior occurs; as such, the sequence of the behaviors can be regarded as a requirement. Use cases are particularly well suited to capture approaches. Although this may sound simple, the fact is that (73) requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the (74) of the system's behavior. Use cases are a simple yet powerful way to express the behavior of the system in way that all stakeholders can easily

understand.

But,like anything, use cases come with their own problems, and as useful as they are,they can be (75). The result is something that is as bad, if not worse, that the original problem.Therein it" s important to utilize use cases effectively without creating a greater problem than the one you started with.

- A. statics
- B. nature
- C. dynamics
- D. originals

第75题 : In a world where it seems we already have too much to do,and too many things to think about, it seems the last thing we need is something new that we have to learn.

But use cases do solve a problem with requirements:with (71) declarative requirements it" s hard to describe steps and sequences of events.

Use cases,stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful.As simple as this sounds,this is important. When confronted only with a pile of requiements, it" s often (72) to make sense of what the authors of the requirements really wanted the system to do.In the preceding example, use cases reduce the ambiguity of the requirements by specifying exactly when and under what conditions certain behavior occurs;as such, the sequence of the behaviors can be regarded as a requirement. Use cases are particularly well suited to capture approaches. Although this may sound simple, the fact is that (73) requirement capture approaches, with their emphasis on declarative requirements and "shall" statements,completely fail to capture fail to capture the (74) of the system" s behavior. Use cases are a simple yet powerful way to express the behavior of the system in way that all stakeholders can easily understand.

But,like anything, use cases come with their own problems, and as useful as they are,they can be (75). The result is something that is as bad, if not worse, that the original problem.Therein it" s important to utilize use cases effectively without creating a greater problem than the one you started with.

- A. misapplied
- B. applied
- C. used
- D. powerful

下午案例分析

第1题：【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。

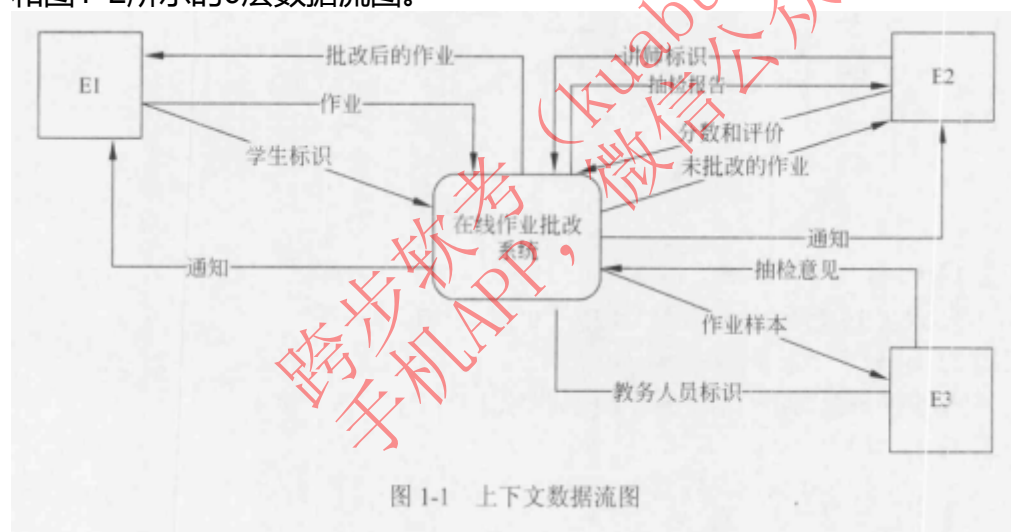
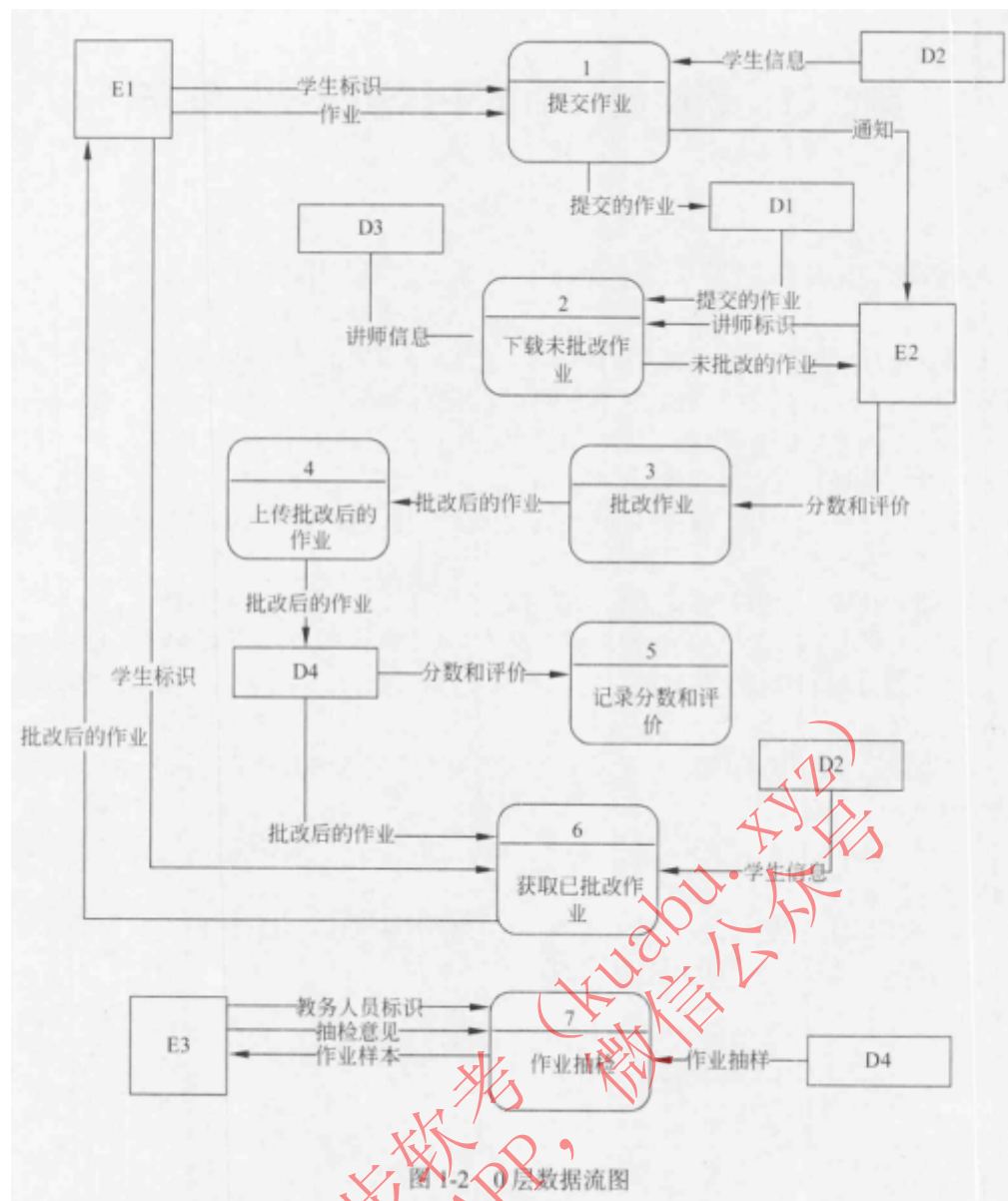


图 1-1 上下文数据流图



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。

第2题：【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一个元组。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)

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。

第3题：【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。

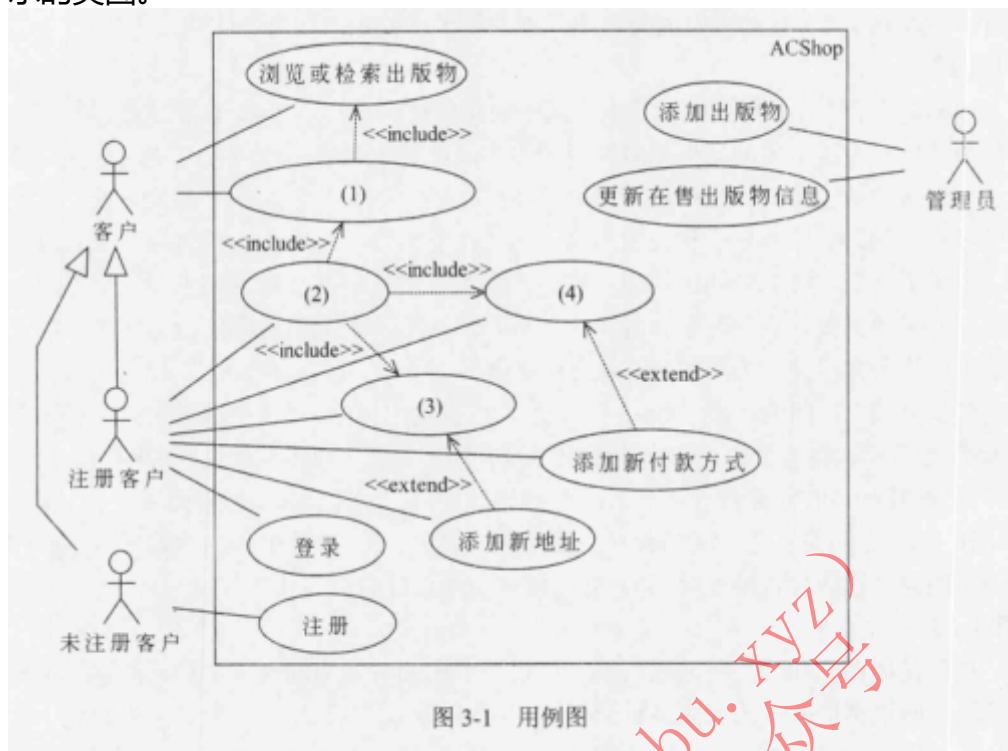


图 3-1 用例图

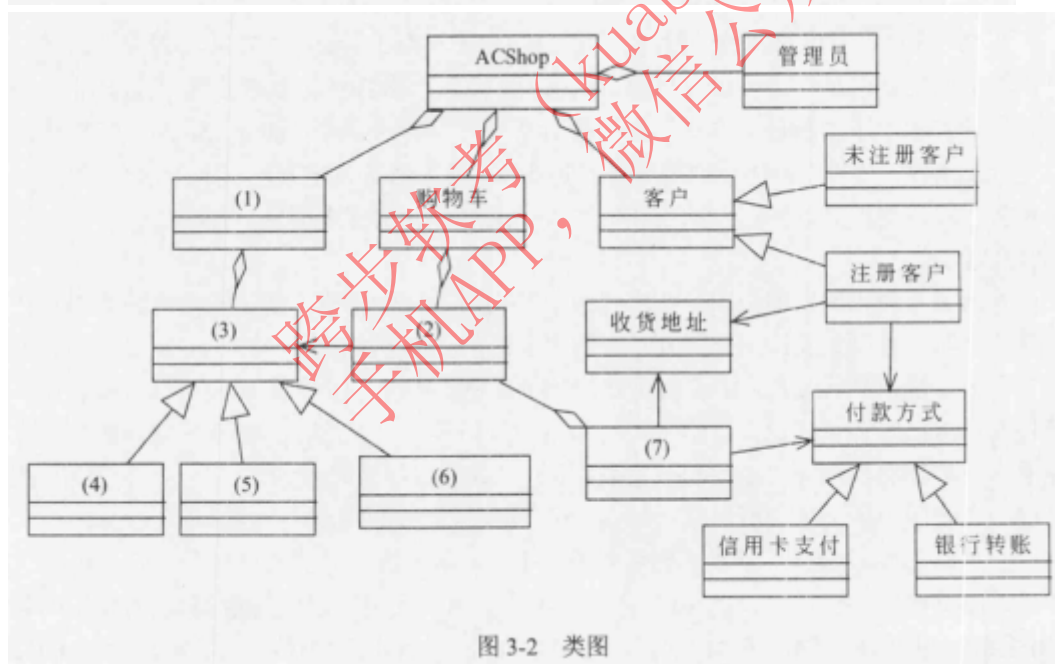


图 3-2 类图

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“添加新支付方式”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。

第4题：【说明】

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

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:

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

计算所有 $c[i][j]$ ($0 \leq i \leq m, 0 \leq j \leq n$) 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。

$\max i, \max j$: 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

```
#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)。

第5题：【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。

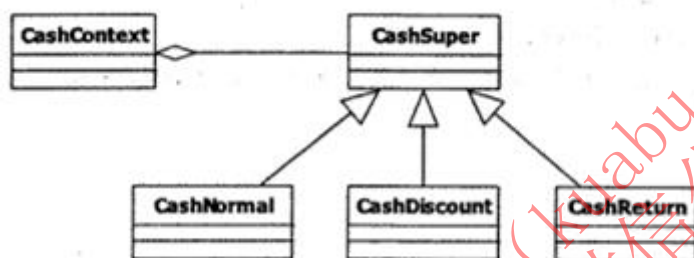


图 5-1 策略模式类图


```
【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) {    return money;    }
};
class CashDiscount : public CashSuper {
private:
    double moneyDiscount;    // 折扣率
public:
    CashDiscount(double discount) {    moneyDiscount= discount;    }
    double acceptCash(double money) {    return money * moneyDiscount;    }
};
class CashRetum : public CashSuper {    // 满额返利
private:
    double moneyCondition;    // 满额数额
    double moneyReturn;    // 返利数额
public:
    CashRetum(double motieyCondition, double moneyReturn) {
        this->moneyCondition=moneyCondition;
        this->moneyReturn=moneyReturn;
    }
    double acceptCash(double money) {
        double result = money;
        if(money>=moneyCondition)
            result=money-(int)(money/moneyCondition ) * moneyReturn;
        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
```

问题：5.1

第6题：【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。

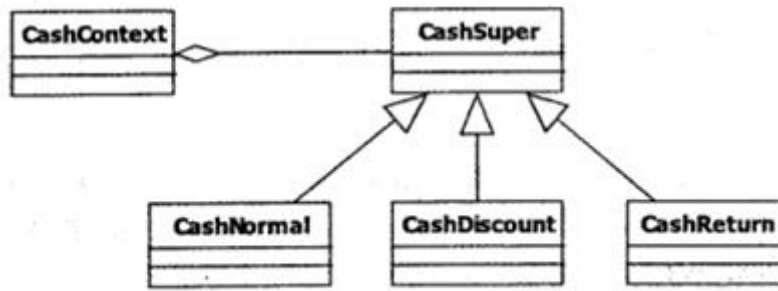


图 6-1 策略模式类图

问题：6.1

跨步软考 (kuabu.xyz)
手机APP，微信公众号

【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
}
```

参考答案与解析

上午综合试卷答案与解析

第1题, 参考答案: D

解析:

本题考查计算机组成基础知识。

DMA控制器在需要的时候代替CPU作为总线主设备, 在不受CPU干预的情况下, 控制I/O设备与系统主存之间的直接数据传输。DMA操作占用的资源是系统总线, 而CPU并非在整个指令执行期间即指令周期内都会使用总线, 故DMA请求的检测点设置在每个机器周期也即总线周期结束时执行, 这样使得总线利用率最高。

第2题, 参考答案: A

解析:

本题考查计算机组成基础知识。

计算机中不同容量、不同速度、不同访问形式、不同用途的各种存储器形成的是一种层次结构的存储系统。所有的存储器设备按照一定的层次逻辑关系通过软硬件连接起来, 并进行有效的管理, 就形成了存储体系。不同层次上的存储器发挥着不同的作用。一般计算机系统中主要有两种存储体系: Cache存储体系由Cache和主存储器构成, 主要目的是提高存储器速度, 对系统程序员以上均透明; 虚拟存储体系由主存储器和在线磁盘存储器等辅存构成, 主要目的是扩大存储器容量, 对应用程序员透明。

第3题, 参考答案: B

解析:

本题考查计算机组成基础知识。

在计算机中使用了类似于十进制科学计数法的方法来表示二进制实数, 因其表示不同的数时小数点位置的浮动不固定而取名浮点数表示法。浮点数编码由两部分组成: 阶码(即指数, 为带符号定点整数, 常用移码表示, 也有用补码的)和尾数(是定点纯小数, 常用补码表示, 或原码表示)。因此可以知道, 浮点数的精度由尾数的位数决定, 表示范围的大小则主要由阶码的位数决定。

第4题, 参考答案: C

解析：

本题考查计算机组成基础知识。

随着主存增加，指令本身很难保证直接反映操作数的值或其地址，必须通过某种映射方式实现对所需操作数的获取。指令系统中将这种映射方式称为寻址方式，即指令按什么方式寻找(或访问)到所需的操作数或信息(例如转移地址信息等)。可以被指令访问到的数据和信息包括通用寄存器、主存、堆栈及外设端口寄存器等。

指令中地址码字段直接给出操作数本身，而不是其访存地址，不需要访问任何地址的寻址方式被称为立即寻址。

第5题，参考答案：B

解析：

本题考查计算机组成基础知识。

直接计算16进制地址包含的存储单元个数即可。

$DABFFH - B3000H + 1 = 27C00H = 162816 = 159k$ ，按字节编址，故此区域的存储容量为159kB。

第6题，参考答案：A

解析：

本题考查计算机组成与结构基础知识。

计算机技术发展使得机器性能提高，随着高级语言的发展，程序员需要更强大的命令，指令集往往结合应用需要不断扩展，推动了指令集越来越复杂，形成了CISC，即Complex Instruction Set Computer，就是使用复杂指令集系统的计算机。与其对应的是RISC，即Reduced Instruction Set Computer，精简指令集系统的计算机。

第7题，参考答案：A

解析：

本题考查的是网络攻击的基础知识。

网络攻击有主动攻击和被动攻击两类。其中主动攻击是指通过一系列的方法，主动向被攻击对象实施破坏的一种攻击方式，例如重放攻击、IP地址欺骗、拒绝服务攻击等均属于攻击者主动向攻击对象发起破坏性攻击的方式。流量分析攻击是通过持续检测现有网络中的流量变化或者变化趋势，而得到相应信息的一种被动攻击方式。

第8题，参考答案：B

解析：

本题考查的是防火墙基础知识。

防火墙是一种放置在网络边界上，用于保护内部网络安全的网络设备。它通过对流经数据流进行分析和检查，可实现对数据包的过滤、保存用户访问网络的记录和服务器代理功能。防火墙不具备检查病毒的功能。

第9题, 参考答案: C

解析:

本题考查网管命令netstat-n的含义以及端口的作用。

从netstat -n的输出信息中可以看出, 本地主机192.168.0.200使用的端口号2011、2038、2052都不是公共端口号。

根据状态提示信息, 其中已经与主机128.105.129.30的80端口建立了普通连接, 与主机100.29.200.110正在等待建立连接, 与主机202.100.112.12的443端口建立连接, 由于443端口主要用于HTTPS服务, 是提供加密和通过安全端口传输的另一种HTTP协议, 所以是建立了安全连接。

第10题, 参考答案: C

解析:

我国著作权法在第10条对权利内容作了较为详尽而具体的规定, 指明著作权的内容包括人身权利和财产权利。著作人身权是指作者享有的与其作品有关的以人格利益为内容的权利, 也称为精神权利, 包括发表权、署名权、修改权和保护作品完整权。著作人身权与作者的身份紧密联系, 永远属于作者本人, 即使作者死亡, 其他任何人不能再拥有它。所以, 我国著作权法第20条规定“作者的署名权、修改权、保护作品完整权的保护期不受限制。”

发表权是属于人身权利, 但发表权是一次性权利, 即发表权行使一次后, 不再享有发表权。发表权是指决定作品是否公之于众的权利, 作品一经发表, 就处于公知状态, 对处于公知状态的作品, 作者不再享有发表权, 以后再次使用作品与发表权无关, 而是行使作品的使用权。

第11题, 参考答案: A

解析:

王某的行为侵犯了公司的软件著作权。因为王某作为公司的职员, 完成的某一综合信息管理系统软件是针对其本职工作中明确指定的开发目标而开发的软件。该软件应为职务作品, 并属于特殊职务作品。公司对该软件享有除署名权外的软件著作权的其他权利, 而王某只享有署名权。王某持有该软件源程序不归还公司的行为, 妨碍了公司正常行使软件著作权, 构成对公司软件著作权的侵犯, 应承担停止侵权法律责任, 交还软件源程序。

第12题, 参考答案: C

解析:

声音是通过空气传播的一种连续的波, 称为声波。声波在时间和幅度上都是连续的模拟信号, 通常称为模拟声音(音频)信号。人们对声音的感觉主要有音量、音调和音色。音量又称音强或响度, 取决于声音波形的幅度, 也就是说, 振幅的大小表明声音的响亮程度或强弱程度。音调与声音的频率有关, 频率高则声音高昂, 频率低则声音低沉。而音色是由混入基音的泛音所决定的, 每个基音都有其固有的频率和不同音强的泛音, 从而使得声音具有其特殊的音色效果。人耳能听得到的音频信号的频率范围是20Hz~20kHz, 包括: 语音

(300Hz~3400Hz)、音乐(20Hz~20kHz)、其他声音(如风声、雨声、鸟叫声、汽车鸣笛声等, 其带宽范围也是20Hz~20kHz), 频率小于20Hz声波信号称为亚音信号(次音信号), 高于20kHz的信号称为超音频信号(超声波)。

第13题, 参考答案: C

解析:

颜色深度是表达图像中单个像素的颜色或灰度所占的位数(bit), 它决定了彩色图像中可出现的最多颜色数, 或者灰度图像中的最大灰度等级数。8位的颜色深度, 表示每个像素有8位颜色位, 可表示 $2^8=256$ 种不同的颜色或灰度等级。表示一个像素颜色的位数越多, 它能表达的颜色数或灰度等级就越多, 其深度越深。

图像深度是指存储每个像素(颜色或灰度)所用的位数(bit), 它也是用来度量图像的分辨率的。像素深度确定彩色图像的每个像素可能有的颜色数, 或者确定灰度图像的每个像素可能有的灰度级数。如一幅图像的图像深度为b位, 则该图像的最多颜色数或灰度级为 2^b 种。显然, 表示一个像素颜色的位数越多, 它能表达的颜色数或灰度级就越多。例如, 只有1个分量的单色图像(黑白图像), 若每个像素有8位, 则最大灰度数目为 $2^8=256$;一幅彩色图像的每个像素用R、G、B三个分量表示, 若3个分量的像素位数分别为4、4、2, 贝撮大颜色数目为 $2^{4+4+2}=2^{10}=1024$, 就是说像素的深度为10位, 每个像素可以是 2^{10} 种颜色中的一种。本题给出8位的颜色深度, 则表示该图像具有 $2^8=256$ 种不同的颜色或灰度等级。

第14题, 参考答案: B

解析:

饱和度是指颜色的纯度, 即颜色的深浅, 或者说掺入白光的程度, 对于同一色调的彩色光, 饱和度越深颜色越纯。当红色加入白光之后冲淡为粉红色, 其基本色调仍然是红色, 但饱和度降低。也就是说, 饱和度与亮度有关, 若在饱和的彩色光中增加白光的成分, 即增加了光能, 而变得更亮了, 但是其饱和度却降低了。对于同一色调的彩色光, 饱和度越高, 颜色越纯。如果在某色调的彩色光中, 掺入其他彩色光, 将引起色调的变化, 而改变白光的成分只引起饱和度的变化。高饱和度的深色光可掺入白色光被冲淡, 降为低饱和度的浅色光。例如, 一束高饱和度的蓝色光投射到屏幕上会被看成深蓝色光, 若再将一束白色光也投射到屏幕上并与深蓝色重叠, 则深蓝色变成淡蓝色, 而且投射的白色光越强, 颜色越淡, 即饱和度越低。相反, 由于在彩色电视的屏幕上的亮度过高, 则饱和度降低, 颜色被冲淡, 这时可以降低亮度(白光)而使饱和度增大, 颜色加深。

当彩色的饱和度降低时, 其固有色彩特性也被降低和发生变化。例如, 红色与绿色配置在一起, 往往具有一种对比效果, 但只有当红色与绿色都呈现饱和状态时, 其对比效果才比较强烈。如果红色与绿色的饱和度都降低, 红色变成浅红或暗红, 绿色变成浅绿或深绿, 再把它们配置在一起时相互的对比特征就会减弱, 而趋于和谐。另外饱和度高的色彩容易让人感到单调刺眼。饱和度低, 色感比较柔和协调, 但混色太杂又容易让人感觉浑浊, 色调显得灰暗。

第15题, 参考答案: C

解析：

本题考查软件开发方法的基础知识。

要求考生掌握典型的软件开发方法的基本概念和应用场合。需求不清晰且规模不太大时采用原型化方法最合适，而数据处理领域的不太复杂的软件，适于用结构化方法进行开发。

第16题，参考答案：A

解析：

本题考查软件开发方法的基础知识。

要求考生掌握典型的软件开发方法的基本概念和应用场合。需求不清晰且规模不太大时采用原型化方法最合适，而数据处理领域的不太复杂的软件，适于用结构化方法进行开发。

第17题，参考答案：D

解析：

本题考查软件项目管理的基础知识。

根据上图计算出关键路径为A-B-C-E-F-I-K-L，其长度为24，关键路径上的活动均为关键活动。

活动BD不在关键路径上，包含该活动的最长路径为A-B-D-G-I-K-L，其长度为22，因此松弛时间为2。

第18题，参考答案：A

解析：

本题考查软件项目管理的基础知识。

根据上图计算出关键路径为A-B-C-E-F-I-K-L，其长度为24，关键路径上的活动均为关键活动。活动BD不在关键路径上，包含该活动的最长路径为A-B-D-G-I-K-L，其长度为22，因此松弛时间为2。

第19题，参考答案：A

解析：

本题考查软件项目管理的基础知识。

人员管理是软件项目管理的一个重要部分，在组织开发团队时，应该考虑开发人员的工作能力、知识背景、工作风格、兴趣爱好等多方面的因素。每个成员的工作任务分配清楚，不应该参与所有阶段的工作。当项目进度滞后于项目计划时，增加开发人员不一定可以加快开发进度。

第20题，参考答案：C

解析：

本题考查程序语言基础知识。

解释程序也称为解释器, 它可以直接解释执行源程序, 或者将源程序翻译成某种中间表示形式后再加以执行; 而编译程序(编译器)则首先将源程序翻译成目标语言程序, 然后在计算机上运行目标程序。这两种语言处理程序的根本区别是: 在编译方式下, 机器上运行的是与源程序等价的目标程序, 源程序和编译程序都不再参与目标程序的执行过程; 而在解释方式下, 解释程序和源程序(或其某种等价表示)要参与到程序的运行过程中, 运行程序的控制权在解释程序。解释器翻译源程序时不产生独立的目标程序, 而编译器则需将源程序翻译成独立的目标程序。

分阶段编译器的工作过程如下图所示。其中, 中间代码生成和代码优化不是必须的。



第21题, 参考答案: B

解析:

本题考查程序语言基础知识。

解释程序也称为解释器, 它可以直接解释执行源程序, 或者将源程序翻译成某种中间表示形式后再加以执行; 而编译程序(编译器)则首先将源程序翻译成目标语言程序, 然后在计算机上运行目标程序。这两种语言处理程序的根本区别是: 在编译方式下, 机器上运行的是与源程序等价的目标程序, 源程序和编译程序都不再参与目标程序的执行过程; 而在解释方式下, 解释程序和源程序(或其某种等价表示)要参与到程序的运行过程中, 运行程序的控制权在解释程序。解释器翻译源程序时不产生独立的目标程序, 而编译器则需将源程序翻译成独立的目标程序。

分阶段编译器的工作过程如下图所示。其中, 中间代码生成和代码优化不是必须的。



第22题, 参考答案: A

解析:

本题考查程序语言基础知识。

后缀式(逆波兰式)是波兰逻辑学家卢卡西维奇发明的一种表示表达式的方法。这种表示方式把运算符写在运算对象的后面, 例如, 把 $a+b$ 写成 $ab+$, 所以也称为后缀式。

借助栈可以方便地对后缀式进行求值。方法为: 先创建一个初始为空的栈, 用来存放运算数。对后缀表达式求值时, 从左至右扫描表达式, 若遇到运算数, 就将其入栈, 若遇到运算符, 就从栈顶弹出需要的运算数并进行运算, 然后将结果压入栈顶, 如此重复, 直到表达式结束。若表达式无错误, 则最后的运算结果就存放在栈顶并且是栈中唯一的元素。

第23题, 参考答案: C

解析:

试题(23)的正确的答案为C。因为信号量S1是一个互斥信号量, 表示半成品箱B1当前有无工人(生产者)使用, 所以初值为1。信号量S5也是一个互斥信号量, 表示成品箱B2当前有无工人或检验员使用, 所以初值为1。

第24题, 参考答案: D

解析:

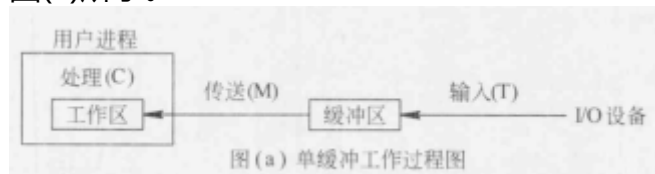
试题(24)的正确的答案为D。信号量S2表示半成品箱B1的容量, 故S2的初值为n。当工人P1不断地将其工序上加工的半成品放入半成品箱B1时, 应该先测试半成品箱是否有空位, 故工人P1使用P(S2), 当工人P2从半成品箱取一件半成品时, 半成品箱B1就空出一个空位, 故工人P2使用V(S2)释放空间。

同理, 信号量S4表示成品箱B2的容量, 故S4的初值为m。当工人P2完成一件产品放入成品箱B2时, 应该先测试成品箱是否有空位, 故工人P2使用P(S4), 当检验员P3从成品箱取一件产品检验时, 成品箱B2就空出一个空位, 故检验员P3使用V(S4)释放空间。

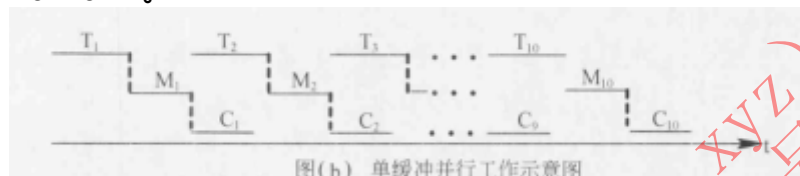
第25题, 参考答案: D

解析:

试题(25)的正确的答案为D。在块设备输入时, 假定从磁盘把一块数据输入到缓冲区的时间为 T , 缓冲区中的数据传送到用户工作区的时间为 M , 而系统处理(计算)的时间为 C , 如图(a)所示。



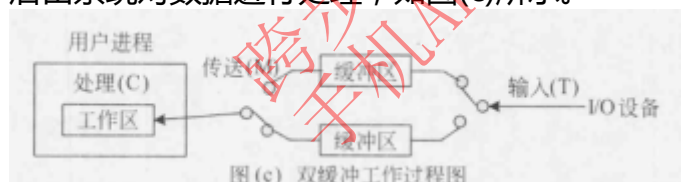
当第一块数据送入用户工作区后, 缓冲区是空闲的就可以传送第二块数据。这样第一块数据的处理 C_1 与第二块数据的输入 T_2 是可以并行的, 依次类推, 如图(b)所示。系统对每一块数据的处理时间为: $\text{Max}(C, T) + M$ 。因为, 当 $T > C$ 时, 处理时间为 $M + T$; 当 $T < C$ 时, 处理时间为 $M + C$ 。本题每一块数据的处理时间为 $15 + 5 = 20$, Doc1文, 牛的处理时间为 $20 * 10 + 1$ 。



第26题, 参考答案: C

解析:

试题(26)的正确的答案为C。双缓冲工作方式基本方法是在设备输入时, 先将数据输入到缓冲区1, 装满后便转向缓冲区2。此时系统可以从缓冲区1中提取数据传送到用户区, 最后由系统对数据进行处理, 如图(c)所示。



双缓冲可以实现对缓冲区中数据的输入 T 和提取 M , 与CPU的计算 C , 三者并行工作, 如图(d)所示。从图中可以看出, 双缓冲进一步加快了I/O的速度, 提高了设备的利用率。在双缓冲时, 系统处理一块数据的时间可以粗略地认为是 $\text{Max}(C, T)$ 。如果 $C < T$, 可使块设备连续输入; 如果 $C > T$, 则可使系统不必等待设备输入。本题每一块数据的处理时间为10, 采用双缓冲需要花费的时间为 $15 * 10 + 5 + 1 = 156$ 。



第27题, 参考答案: D

解析:

R2资源有3个, 已分配2个, P3申请1个R2资源可以得到满足, 故进程P3可以运行完毕释放其占有的资源。这样可以使得P1、P2都变为非阻塞节点, 得到所需资源运行完毕, 因此, 该进程资源图是可化简的。

第28题, 参考答案: C

解析:

在同一进程中的各个线程都可以共享该进程所拥有的资源, 如访问进程地址空间中的每一个虚地址; 访问进程所拥有的已打开文件、定时器、信号量机构等, 但是不能共享进程中某线程的栈指针。

第29题, 参考答案: B

解析:

本题考查软件开发过程模型的基础知识。

瀑布模型将开发阶段描述为从一个阶段瀑布般地转换到另一个阶段的过程。

原型模型中, 开发人员快速地构造整个系统或者系统的一部分以理解或澄清问题。螺旋模型将开发活动和风险管理结合起来, 以减小风险。

喷泉模型开发过程模型以用户需求为动力, 以对象为驱动, 适合于面向对象的开发方法。在这几种开发过程模型中, 原型模型不适宜大规模软件的开发。

第30题, 参考答案: D

解析:

本题考查软件开发过程模型的基础知识。

根据题干描述, 合适的开发过程模型为喷泉模型。

第31题, 参考答案: D

解析:

本题考查软件质量的基础知识。

ISO/IEC软件质量模型由三个层次组成: 第一层是质量特性, 第二层是质量子特性, 第三层是度量指标。易使用性是指与为使用所需的努力和由一组规定或隐含的用户对这样使用所做的个别评价有关的一组属性。其子特性包括易理解性、易学性、易操作性。

第32题, 参考答案: B

解析:

本题考查软件设计的基础知识。

子系统结构设计中, 重点关注如何划分模块, 子系统之间以及模块之间的数据和调用关系, 模块结构质量等这些粗粒度的问题; 而对每个模块内部进行设计时, 才需要考虑采用的数据结构以及处理的算法。

第33题, 参考答案: D

解析:

本题考查结构化分析方面的基础知识。

在结构化分析中, 用数据流图对软件功能建模, 加工是数据流的一个重要要素, 可以用多种方式描述, 如流程图、NS盒图等, 其中决策树和决策表适于用来表示加工中涉及多个逻辑条件的情况。

第34题, 参考答案: B

解析:

本题考查软件工程过程及软件工具的基础知识。

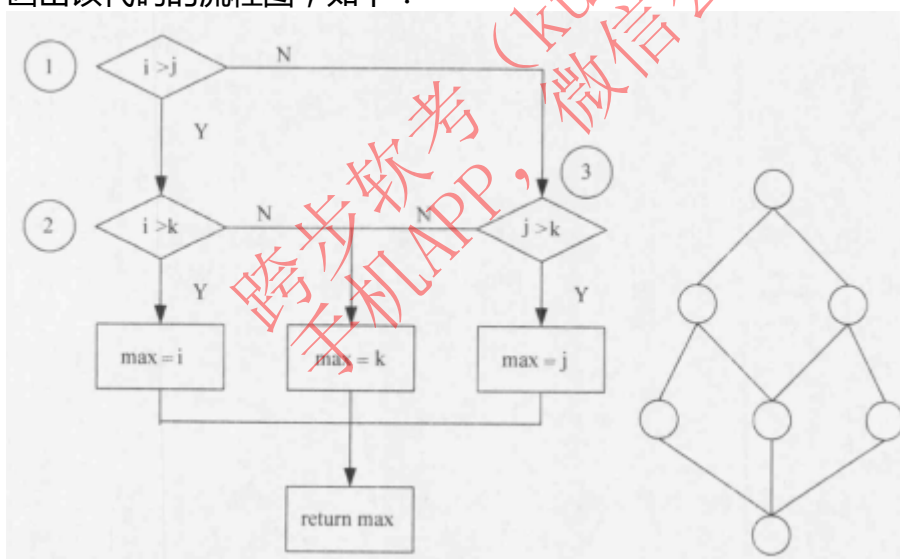
逆向工程从源代码得到软件系统的规格说明和设计信息, 属于软件维护阶段行为, 因此逆向工程工具属于软件维护工具。

第35题, 参考答案: B

解析:

本题考查软件测试的基础知识。

画出该代码的流程图, 如下:



要满足条件覆盖, 要求三个判断框的Y和N至少要经过一次, 即1Y2Y; 1Y2N; 1N3Y; 1N3N, 至少需要4个测试用例。

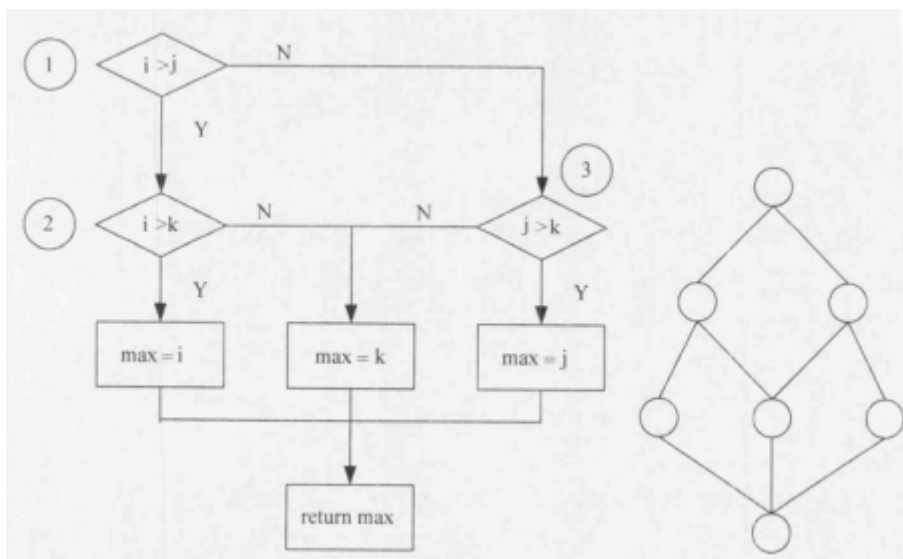
McCabe度量法是一种基于程序控制流的复杂性度量方法, 环路复杂性为 $V(G)=m-n+2$, 图中 $m=9$, $n=7$, $V(G)=9-7+2=4$ 。

第36题, 参考答案: D

解析:

本题考查软件测试的基础知识。

画出该代码的流程图, 如下:



要满足条件覆盖, 要求三个判断框的Y和N至少要经过一次, 即1Y2Y; 1Y2N; 1N3Y; 1N3N, 至少需要4个测试用例。

McCabe度量法是一种基于程序控制流的复杂性度量方法, 环路复杂性为 $V(G)=m-n+2$, 图中 $m=9$, $n=7$, $V(G)=9-7+2=4$ 。

第37题, 参考答案: A

解析:

本题考查面向对象的基本知识。

在面向对象系统中, 对象是基本的运行时的实体, 它既包括数据(属性), 也包括作用于数据的操作(行为)。所以, 一个对象把属性和行为封装为一个整体。封装是一种信息隐蔽技术, 它的目的是使对象的使用者和生产者分离, 使对象的定义和实现分开。从程序设计者来看, 对象是一个程序模块; 从用户来看, 对象为他们提供了所希望的行为。在对象内的操作通常叫做方法。一个对象通常可由对象名、属性和方法三部分组成。

一个类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的共同行为和属性, 这些对象共享这些行为和属性。

第38题, 参考答案: D

解析:

本题考查面向对象的基本知识。

在面向对象系统中, 对象是基本的运行时的实体, 它既包括数据(属性), 也包括作用于数据的操作(行为)。所以, 一个对象把属性和行为封装为一个整体。封装是一种信息隐蔽技术, 它的目的是使对象的使用者和生产者分离, 使对象的定义和实现分开。从程序设计者来看, 对象是一个程序模块; 从用户来看, 对象为他们提供了所希望的行为。在对象内的操作通常叫做方法。一个对象通常可由对象名、属性和方法三部分组成。

一个类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的共同行为和属性, 这些对象共享这些行为和属性。

第39题, 参考答案: B

解析：

本题考查面向对象和统一建模语言(UML)的基本知识。

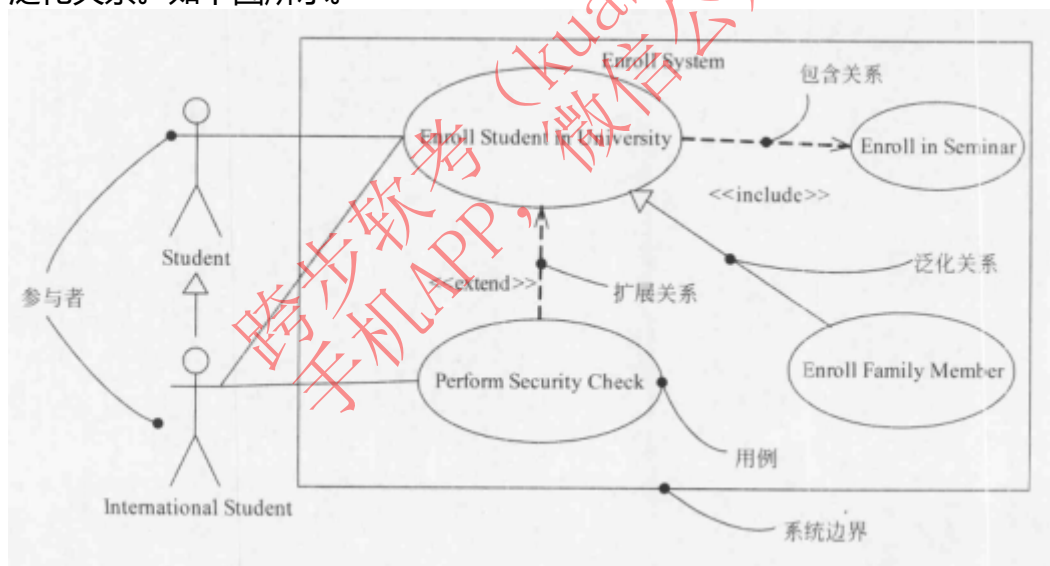
一个类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的行为和属性。有些类之间存在一般和特殊关系, 即一些类是某个类的特殊情况, 某个类是一些类的一般情况, 即继承关系。继承是父类和子类之间共享数据和方法的机制。父类描述了这些子类的公共属性和方法。一个子类可以继承它的父类(或祖先类)中的属性和方法, 这些属性和操作在子类中不必定义, 子类中还可以定义自己的属性和方法, 也可以重新定义父类中已经定义的方法, 即重置或覆盖(overriding)。UML类图中, 如果父类中已有方法名在子类中不出现, 表示子类继承父类中的方法; 如果父类中已有方法名在子类中出现了, 就表示子类在继承父类接口定义的前提下, 用适合于自己要求的实现去替换父类中的相应实现, 即覆盖了父类中的方法。

第40题, 参考答案:A

解析：

本题考查统一建模语言(UML)的基本知识。

用例图(use case diagram)展现了一组用例、参与者(Actor)以及它们之间的关系。用例图通常包括用例、参与者, 以及用例之间的扩展关系(<<extend>>)和包含关系(<<include>>), 参与者和用例之间的关联关系, 用例与用例以及参与者与参与者之间的泛化关系。如下图所示。



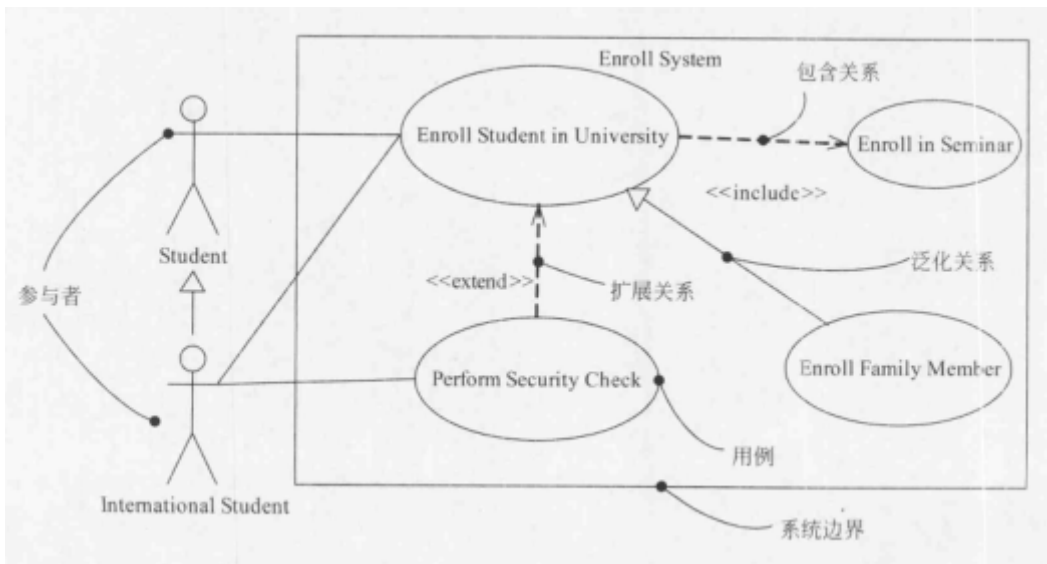
用例图用于对系统的静态用例视图进行建模, 主要支持系统的行为, 即该系统在它的周边环境的语境中所提供的外部可见服务。

第41题, 参考答案:C

解析：

本题考查统一建模语言(UML)的基本知识。

用例图(use case diagram)展现了一组用例、参与者(Actor)以及它们之间的关系。用例图通常包括用例、参与者, 以及用例之间的扩展关系(<<extend>>)和包含关系(<<include>>), 参与者和用例之间的关联关系, 用例与用例以及参与者与参与者之间的泛化关系。如下图所示。



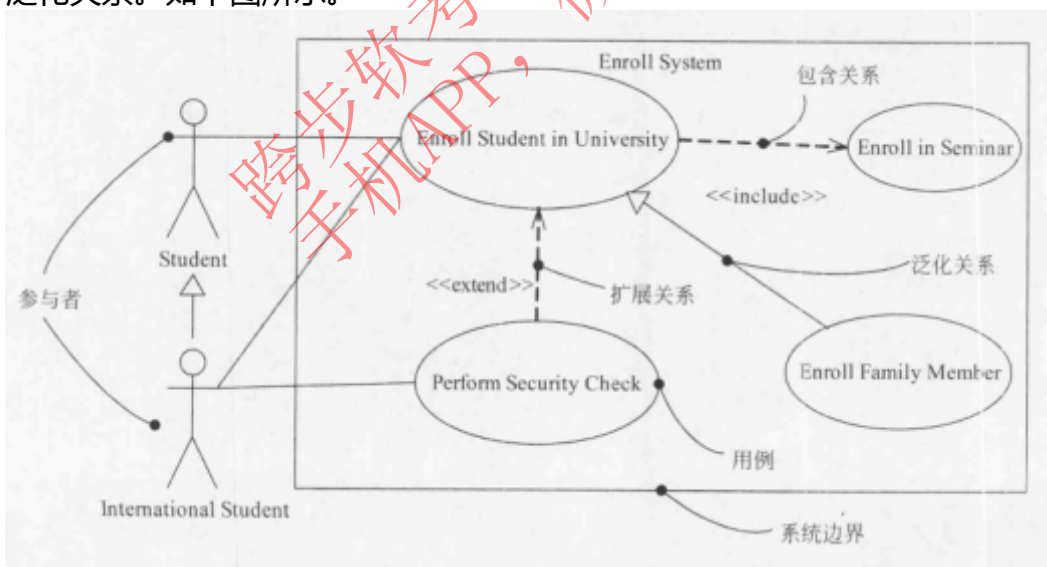
用例图用于对系统的静态用例视图进行建模，主要支持系统的行为，即该系统在它的周边环境的语境中所提供的外部可见服务。

第42题，参考答案：B

解析：

本题考查统一建模语言(UML)的基本知识。

用例图(use case diagram)展现了一组用例、参与者(Actor)以及它们之间的关系。用例图通常包括用例、参与者，以及用例之间的扩展关系(<<extend>>)和包含关系(<<include>>)，参与者和用例之间的关联关系，用例与用例以及参与者与参与者之间的泛化关系。如下图所示。



用例图用于对系统的静态用例视图进行建模，主要支持系统的行为，即该系统在它的周边环境的语境中所提供的外部可见服务。

第43题，参考答案：C

解析：

本题考查统一建模语言(UML)的基本知识。

UML中提供了多种建模系统的图, 体现系统的静态方面和动态方面。类图(class diagram)展现了一组对象、接口、协作和它们之间的关系。在面向对象系统的建模中所建立的最常见的图就是类图。类图给出系统的静态设计视图。部署图(deployment diagram)是用来对面向对象系统的物理方面建模的方法, 展现了运行时处理结点以及其中构件(制品)的配置。部署图对系统的静态部署视图进行建模, 它与组件图(构件图)相关。组件图或构件图(component diagram)展现了一组构件之间的组织和依赖, 如题中的图所示。组件图或构件图专注于系统的静态实现视图。它与类图相关, 通常把构件映射为一个或多个类、接口或协作。UML部署图经常被认为是一个网络图。

第44题, 参考答案: D

解析:

本题考查设计模式的基本概念。?

Singleton(单例)设计模式是一种创建型模式, 其意图是保证一个类仅有一个实例, 并提供一个访问这个唯一实例的全局访问点。单例模式适用于当类只能有一个实例而且客户可以从一个众所周知的访问点访问它时, 以及当这个唯一实例应该是通过子类化可扩展的, 并且客户应该无需更改代码就能使用一个扩展的实例时。

第45题, 参考答案: D

解析:

本题考查设计模式的基本概念。每种设计模式都有特定的意图, 描述一个在我们周围不断重复发生的问题, 以及该问题的解决方案的核心, 使该方案能够重用而不必做重复劳动。组合(Composite)模式将对象组合成树形结构以表示“部分-整体”的层次结构, 使得用户对单个对象和组合对象的使用具有一致性。适用于: 想表示对象的部分-整体层次结构; 希望用户忽略组合对象与单个对象的不同, 用户将统一地使用组合结构中的所有对象。

外观(Facade)模式为子系统的一组接口提供一个一致的界面, Facade模式定义了一个高层接口, 这个接口使得这一子系统更加容易使用。适用于: 要为一个复杂子系统提供一个简单接口时, 子系统往往因为不断演化而变得越来越复杂; 客户程序与抽象类的实现部分之间存在着很大的依赖性; 当需要构建一个层次结构的子系统时, 使用Facade模式定义子系统中每层的入口点。

享元(Flyweight)模式运用共享技术有效地支持大量细粒度的对象。适用于: 一个应用程序使用了大量的对象; 完全由于使用大量的对象, 造成很大的存储开销; 对象的大多数状态都可变为外部状态; 如果删除对象的外部状态, 那么可以用相对较少的共享对象取代很多组对象: 应用程序不依赖于对象标识。

装饰器(Decorator)模式描述了以透明围栏来支持修饰的类和对象的关系, 动态地给一个对象添加一些额外的职责, 从增加功能的角度来看, 装饰器模式相比生成子类更加灵活。适用于: 在不影响其他对象的情况下, 以动态、透明的方式给单个对象添加职责; 处理那些可以撤销的职责; 当不能采用生成子类的方式进行扩充时。

工厂方法(Factory Method)定义一个用于创建对象的接口, 让子类决定将哪一个类实例化, 使一个类的实例化延迟到其子类。适用于: 当一个类不知道它所必须创建的对象的时候; 当一个类希望由它的子类来指定它所创建的对象的时候; 当类将创建对象的职责委托给多个帮助子类中的某一个, 并且希望将哪一个帮助子类是代理者这一信息局部化的时候。

观察者(Observer)模式定义对象间的一种一对多的依赖关系, 当一个对象的状态发生改变

时, 所有依赖于它的对象都得到通知并被自动更新。适用于: 当一个抽象模型有两个方面, 其中一个方面依赖于另一个方面, 将这两者封装在独立的对象中以使它们可以各自独立地改变和复用; 当对一个对象的改变需要同时改变其他对象, 而不知道具体有多少对象有待改变时; 当一个对象必须通知其他对象, 而它又不能假定其他对象是谁, 即不希望这些对象是紧耦合的。

中介者(Mediator)用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式地相互引用, 从而使其耦合松散, 而且可以独立地改变它们之间的交互。适用于: 一组对象以定义良好但是复杂的方式进行通信, 产生的相互依赖关系结构混乱且难以理解; 一个对象引用其他很多对象并且直接与这些对象通信, 导致难以复用该对象; 想定制一个分布在多个类中的行为, 而又不想生成太多的子类。欲使一个后端数据模型能够被多个前端用户界面连接, 采用中介者模式最合适。

第46题, 参考答案: A

解析:

本题考查设计模式的基本概念。每种设计模式都有特定的意图, 描述一个在我们周围不断重复发生的问题, 以及该问题的解决方案的核心, 使该方案能够重用而不必做重复劳动。组合(Composite)模式将对象组合成树形结构以表示“部分-整体”的层次结构, 使得用户对单个对象和组合对象的使用具有一致性。适用于: 想表示对象的部分-整体层次结构; 希望用户忽略组合对象与单个对象的不同, 用户将统一地使用组合结构中的所有对象。

外观(Facade)模式为子系统的一组接口提供一个一致的界面, Facade模式定义了一个高层接口, 这个接口使得这一子系统更加容易使用。适用于: 要为一个复杂子系统提供一个简单接口时, 子系统往往因为不断演化而变得越来越复杂; 客户程序与抽象类的实现部分之间存在着很大的依赖性; 当需要构建一个层次结构的子系统时, 使用Facade模式定义子系统中每层的入口点。

享元(Flyweight)模式运用共享技术有效地支持大量细粒度的对象。适用于: 一个应用程序使用了大量的对象; 完全由于使用大量的对象, 造成很大的存储开销; 对象的大多数状态都可变为外部状态; 如果删除对象的外部状态, 那么可以用相对较少的共享对象取代很多组对象: 应用程序不依赖于对象标识。

装饰器(Decorator)模式描述了以透明围栏来支持修饰的类和对象的关系, 动态地给一个对象添加一些额外的职责, 从增加功能的角度来看, 装饰器模式相比生成子类更加灵活。适用于: 在不影响其他对象的情况下, 以动态、透明的方式给单个对象添加职责; 处理那些可以撤销的职责; 当不能采用生成子类的方式进行扩充时。

工厂方法(Factory Method)定义一个用于创建对象的接口, 让子类决定将哪一个类实例化, 使一个类的实例化延迟到其子类。适用于: 当一个类不知道它所必须创建的对象的类的时候; 当一个类希望由它的子类来指定它所创建的对象的时候; 当类将创建对象的职责委托给多个帮助子类中的某一个, 并且希望将哪一个帮助子类是代理者这一信息局部化的时候。

观察者(Observer)模式定义对象间的一种一对多的依赖关系, 当一个对象的状态发生改变时, 所有依赖于它的对象都得到通知并被自动更新。适用于: 当一个抽象模型有两个方面, 其中一个方面依赖于另一个方面, 将这两者封装在独立的对象中以使它们可以各自独立地改变和复用; 当对一个对象的改变需要同时改变其他对象, 而不知道具体有多少对象有待改变时; 当一个对象必须通知其他对象, 而它又不能假定其他对象是谁, 即不希望这些对象是紧耦合的。

中介者(Mediator)用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式地相互引用, 从而使其耦合松散, 而且可以独立地改变它们之间的交互。适用于: 一组对

象以定义良好但是复杂的方式进行通信, 产生的相互依赖关系结构混乱且难以理解; 一个对象引用其他很多对象并且直接与这些对象通信, 导致难以复用该对象; 想定制一个分布在多个类中的行为, 而又不想生成太多的子类。欲使一个后端数据模型能够被多个前端用户界面连接, 采用中介者模式最合适。

第47题, 参考答案: D

解析:

本题考查设计模式的基本概念。每种设计模式都有特定的意图, 描述一个在我们周围不断重复发生的问题, 以及该问题的解决方案的核心, 使该方案能够重用而不必做重复劳动。组合(Composite)模式将对象组合成树形结构以表示“部分-整体”的层次结构, 使得用户对单个对象和组合对象的使用具有一致性。适用于: 想表示对象的部分-整体层次结构; 希望用户忽略组合对象与单个对象的不同, 用户将统一地使用组合结构中的所有对象。

外观(Facade)模式为子系统的一组接口提供一个一致的界面, Facade模式定义了一个高层接口, 这个接口使得这一子系统更加容易使用。适用于: 要为一个复杂子系统提供一个简单接口时, 子系统往往因为不断演化而变得越来越复杂; 客户程序与抽象类的实现部分之间存在着很大的依赖性; 当需要构建一个层次结构的子系统时, 使用Facade模式定义子系统中每层的入口点。

享元(Flyweight)模式运用共享技术有效地支持大量细粒度的对象。适用于: 一个应用程序使用了大量的对象; 完全由于使用大量的对象, 造成很大的存储开销; 对象的大多数状态都可变为外部状态; 如果删除对象的外部状态, 那么可以用相对较少的共享对象取代很多组对象: 应用程序不依赖于对象标识。

装饰器(Decorator)模式描述了以透明围栏来支持修饰的类和对象的关系, 动态地给一个对象添加一些额外的职责, 从增加功能的角度来看, 装饰器模式相比生成子类更加灵活。适用于: 在不影响其他对象的情况下, 以动态、透明的方式给单个对象添加职责; 处理那些可以撤销的职责; 当不能采用生成子类的方式进行扩充时。

工厂方法(Factory Method)定义一个用于创建对象的接口, 让子类决定将哪一个类实例化, 使一个类的实例化延迟到其子类。适用于: 当一个类不知道它所必须创建的对象的类的时候; 当一个类希望由它的子类来指定它所创建的对象的时候; 当类将创建对象的职责委托给多个帮助子类中的某一个, 并且希望将哪一个帮助子类是代理者这一信息局部化的时候。

观察者(Observer)模式定义对象间的一种一对多的依赖关系, 当一个对象的状态发生改变时, 所有依赖于它的对象都得到通知并被自动更新。适用于: 当一个抽象模型有两个方面, 其中一个方面依赖于另一个方面, 将这两者封装在独立的对象中以使它们可以各自独立地改变和复用; 当对一个对象的改变需要同时改变其他对象, 而不知道具体有多少对象有待改变时; 当一个对象必须通知其他对象, 而它又不能假定其他对象是谁, 即不希望这些对象是紧耦合的。

中介者(Mediator)用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式地相互引用, 从而使其耦合松散, 而且可以独立地改变它们之间的交互。适用于: 一组对象以定义良好但是复杂的方式进行通信, 产生的相互依赖关系结构混乱且难以理解; 一个对象引用其他很多对象并且直接与这些对象通信, 导致难以复用该对象; 想定制一个分布在多个类中的行为, 而又不想生成太多的子类。欲使一个后端数据模型能够被多个前端用户界面连接, 采用中介者模式最合适。

第48题, 参考答案: C

解析：

本题考查程序语目基础知识。

程序已经开始运行, 说明编译时无错误, 因此不是语法错误和词法错误, 编译时发现的语义错误称为静态的语义错误。运行时陷入死循环属于动态语义错误。

第49题, 参考答案：D

解析：

本题考查程序语言基础知识。

若存在一条从初态到某一终止状态的路径, 且这条路径上所有弧的标记符连接成的字符串等于 ω , 则称 ω 可由NFA识别(接受或读出)。

对于题中给出的NFA, 其初态为 q_0 , q_0 上的自回路表示识别零个或多个1, 接下来识别出一个0时进入状态 q_1 , q_1 上的自回路表示识别零个或多个0, 接下来识别出1个1之后再回到 q_0 。

例如, 该自动机可识别空串(因为 q_0 既是初态, 也是终态), 01、00001、101、1、11、111、1111等。

01的识别路径为 $q_0 \rightarrow q_1 \rightarrow q_0$

00001的识别路径为 $q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_0$

101的识别路径为 $q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_0$

1的识别路径为 $q_0 \rightarrow q_0$

11的识别路径为 $q_0 \rightarrow q_0 \rightarrow q_0$

111的识别路径为 $q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$

1111的识别路径为 $q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$

识别字符串时必须从初始状态 q_0 出发, 并回到状态 q_0 , 因此对于仅由1构成的任意长度的串, 在识别过程中不会离开 q_0 。当识别出一个0而离开 q_0 后就进入 q_1 , 此后的字符若全部为0, 则会一直在 q_1 , 直到识别出一个1而回到 q_0 , 因此除了空串, 该NFA识别的字符串必须以1结尾。

第50题, 参考答案：A

解析：

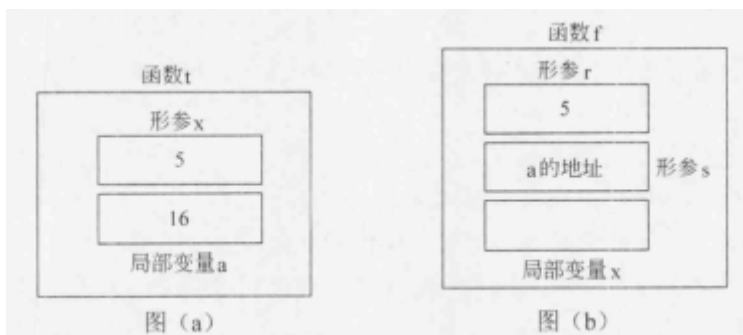
本题考查程序语言基础知识。

若函数调用时采用传值方式, 则是将实参的值传给形参, 再执行被调用的函数, 对形参的修改不影响实参。若采用传引用方式, 则是将实参的地址传递给形参, 本质上是通过间接访问的方式修改实参, 也可以简化理解为: 在被调用函数中对形参的修改等于是对实参进行修改。

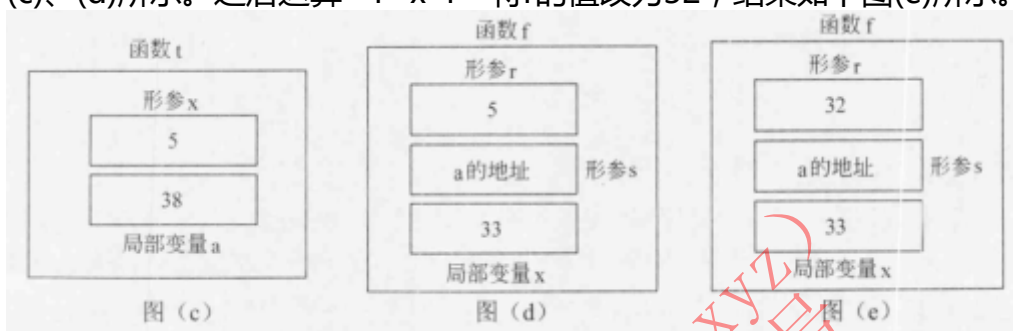
当函数 t 和 f 运行时, 其每个形参和局部变量都有各自的存储单元, 下面图中矩形框表示存储单元。

如题中所述, 调用 t 时传递给其形参 x 的值为5。因此函数 t 被调用而执行时, 在执行函数调用 $f(x, a)$ 之前, 其形参 x 和局部变量 a 的值如下图(a)所示。

执行函数调用 $f(x, a)$ 时, t 中 x 的值传给 f 的形参 r , a 的地址传给 f 的形参 s , 如下图(b)所示。



在 f 执行时，其局部变量 x 的值由运算 “ $X=2*s+1$ ” 改为 33，其中运算时可理解为 s 提供了 t 的局部变量 a 的值(是由间接访问机制实现的)。接下来的运算为 “ $S=x+r$ ”，也就是将 x 的值(即 33)与 r 的值(即 5)相加得到 38，然后(通过间接访问)存入 t 的局部变量 a，结果如下图 (c)、(d) 所示。之后运算 “ $r=x-1$ ” 将 r 的值改为 32，结果如下图 (e) 所示。



当函数 f 运行结束并返回函数 t 后，函数 f 的运行空间将由系统撤销，接下来运算 “ $a-x$ ” 产生的值为 33(即 $38-5$)，因此函数 t 的返回值为 33。

第51题，参考答案：B

解析：

本题考查数据库的基本概念。

数据库通常采用三级模式结构，其中，视图对应外模式、基本表对应模式、存储文件对应内模式。

第52题，参考答案：C

解析：

本题考查对数据库应用系统设计中逻辑结构设计的掌握。

在数据库设计中，将 E-R 图转换为关系模式是逻辑设计的主要内容。转换中将实体转换为关系模式，对实体中的派生属性不予考虑，组合属性只取各组合分量，若含多值属性，通常一个实体对应一个关系模式。对实体中的多值属性，取实体的码和多值属性构成新增的关系模式，且该新增关系模式中，实体的码多值决定多值属性，属于平凡的多值依赖，关系属于 4NF。

第53题，参考答案：D

解析：

本题考查分布式数据库基本概念。

分片透明是指用户或应用程序不需要知道逻辑上访问的表具体是怎么分块存储的, 复制透明是指采用复制技术的分布方法, 用户不需要知道数据是复制到哪些节点, 如何复制的。位置透明是指用户无须知道数据存放的物理位置, 逻辑透明, 即局部数据模型透明, 是指用户或应用程序无须知道局部场地使用的是哪种数据模型。

第54题, 参考答案: A

解析:

本题考查分布式数据库基本概念。

分片透明是指用户或应用程序不需要知道逻辑上访问的表具体是怎么分块存储的, 复制透明是指采用复制技术的分布方法, 用户不需要知道数据是复制到哪些节点, 如何复制的。位置透明是指用户无须知道数据存放的物理位置, 逻辑透明, 即局部数据模型透明, 是指用户或应用程序无须知道局部场地使用的是哪种数据模型。

第55题, 参考答案: C

解析:

本题主要考核关系模式规范化方面的相关知识。

试题(55)的正确答案为C。因为根据函数依赖集F可知属性 A_3 和 A_5 只出现在函数依赖的左部, 故必为候选关键字属性, 又因为 A_3A_5 可以决定关系R中的全部属性, 故关系模式R的一个主键是 A_3A_5 。

第56题, 参考答案: B

解析:

试题(56)的正确答案为B。因为根据函数依赖集F可知, R中的每个非主属性完全函数依赖于 A_3A_5 , 但该函数依赖集中存在传递依赖, 所以R是2NF。

第57题, 参考答案: B

解析:

本题考查数据结构基础知识。

栈和队列都是线性的数据结构。栈的操作要求是入栈和出栈都在表尾进行, 即在栈中有多个元素时, 后进去的元素先出来, 特点是后进先出, 元素入栈的顺序与出栈的顺序可以相同也可以不同。而队列的修改要求是在队尾加入元素, 在队头删除元素, 特点是先进先出, 元素的入队顺序与出队顺序一定相同。

将一个栈和队列连接后, 进出队列的元素顺序是相同的, 而进入队列的元素顺序正是从栈中出来的元素顺序, 因此, 正确的叙述为出队序列与出栈序列一定相同。

第58题, 参考答案: A

解析：

本题考查数据结构基础知识。

解答该问题需先计算排列在 A_{ij} 之前的元素个数。

在按行存储方式下, 存储在 A_{ij} 之前的元素分为 $i-1$ 行, 除第1行外, 每行3个元素。在第 i 行上, A_{ij} 之前的元素个数分为三种情况: $i > j$ 时为0个, $i = j$ 时有1个, $i < j$ 时为2个, 概括为 $j-i+1$ 个。

综上, 排列在 A_{ij} 之前的元素个数为 $(i-1) \times 3 - 1 + j - i + 1$, 即 $2i + j - 3$ 。

由于数组B的下标从1开始, 所以 $k = 2i + j - 3 + 1$ 。

第59题, 参考答案: D

解析：

本题考查数据结构基础知识。

由于序列的第一个元素是结点7, 最后一个元素是结点1, 因此, 左右子树的遍历顺序是先右后左。观察结点7的左子树, 遍历顺序为654, 因此是中序遍历过程。所以答案为RDL。

第60题, 参考答案: B

解析：

本题考查数据结构基础知识。

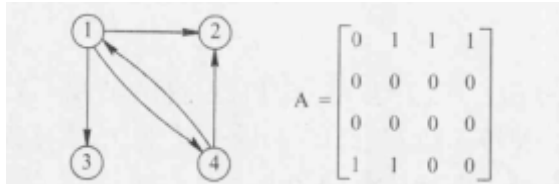
对55个元素构成的有序表进行折半查找时, 可用判定树描述查找过程。由于 $A[19]$ 小于中间元素 $A[28]$, 所以判定树的左分支如下图所示。从中可知, 查找过程中参与比较的元素分别为 $A[28]$ 、 $A[14]$ 、 $A[21]$ 、 $A[17]$ 、 $A[19]$ 。

第61题, 参考答案: A

解析：

本题考查数据结构基础知识。

通过一个例子说明。某有向图及其邻接矩阵如下图所示。



邻接矩阵中的每个非零元素都表示一条弧, 所以非零元素数目为弧的个数 e 。

第62题, 参考答案: D

解析：

本题考查算法分析的基础知识。

根据主方法, 先计算算法A的时间复杂度, $a=8$, $b=2$, $\log_b a = \log_2 8 = 3$, 而 $f(n) = n^2$, 因

此时间复杂度为 $\theta(n^3)$ 。然后计算算法B的时间复杂度, $a=X$, $b=4$, $\log_b a = \log_4 X$, 而 $f(n)=n^2$, 若算法B和算法A的效率一样, 则X应该为64($\log_4 64=3$), 而现在要使得B比A快, 则X应该比64小, 因此最大的整数应该为63。

第63题, 参考答案: C

解析:

本题考查算法分析的基础知识。

根据主方法, 先计算算法A的时间复杂度, $a=8$, $b=2$, $\log_b a = \log_2 8 = 3$, 而 $f(n)=n^2$, 因此时间复杂度为 $\theta(n^3)$ 。然后计算算法B的时间复杂度, $a=X$, $b=4$, $\log_b a = \log_4 X$, 而 $f(n)=n^2$, 若算法B和算法A的效率一样, 则X应该为64($\log_4 64=3$), 而现在要使得B比A快, 则X应该比64小, 因此最大的整数应该为63。

第64题, 参考答案: A

解析:

本题考查算法设计和排序的基础知识。

排序是一类最基本的操作, 因此要求考生熟悉一些典型的排序算法, 包括其算法思想、时空复杂度以及应用场合。若数据基本有序, 插入排序应该是最佳选择, 输入数据是否有序对归并和计数排序算法并没有影响。对传统的快速排序算法, 输入数据有序反而使其效率最低。若关键字取值范围较小, 则计数排序是最佳选择, 因为在该情况下, 该算法的时间复杂度为线性时间。

第65题, 参考答案: D

解析:

本题考查算法设计和排序的基础知识。

排序是一类最基本的操作, 因此要求考生熟悉一些典型的排序算法, 包括其算法思想、时空复杂度以及应用场合。若数据基本有序, 插入排序应该是最佳选择, 输入数据是否有序对归并和计数排序算法并没有影响。对传统的快速排序算法, 输入数据有序反而使其效率最低。若关键字取值范围较小, 则计数排序是最佳选择, 因为在该情况下, 该算法的时间复杂度为线性时间。

第66题, 参考答案: B

解析:

集线器是物理层设备, 相当于在10BASE2局域网中把连接工作站的同轴电缆收拢在一个盒子里, 这个盒子只起到接收和发送的功能, 可以检测发送冲突, 但不能识别数据链路层的帧。网桥是数据链路层设备, 它可以识别数据链路层MAC地址, 有选择地把帧发送到输出端口, 网桥也可以有多个端口, 如果网桥端口很多, 并配置了加快转发的硬件, 就成为局域网交换机。

第67题, 参考答案: B

解析:

本题考查POP3协议及POP3服务器方面的基础知识。
POP3协议是TCP/IP协议簇中用于邮件接收的协议。邮件客户端通过与服务器之间建立TCP连接, 采用Client/Server计算模式来传送邮件。

第68题, 参考答案: C

解析:

TCP的流量控制采用了可变大小的滑动窗口协议, 由接收方指明接收缓冲区的大小(字节数), 发送方发送了规定的字节数后等待接收方的下一次请求。固定大小的滑动窗口协议用在数据链路层的HDLC中。可变大小的滑动窗口协议可以应付长距离通信过程中线路延迟不确定的情况, 而固定大小的滑动窗口协议则适合链路两端点之间通信延迟固定的情况。

第69题, 参考答案: D

解析:

主机路由的子网掩码是255.255.255.255。网络路由要指明一个子网, 所以不可能为全1, 默认路由是访问默认网关, 而默认网关与本地主机属于同一个子网, 其子网掩码也应该与网络路由相同, 对静态路由也是同样的道理。

第70题, 参考答案: B

解析:

在层次化局域网模型中, 核心层的主要功能是将分组从一个区域高速地转发到另一个区域。核心层是因特网络的高速骨干, 由于其重要性, 因此在设计中应该采用冗余组件设计, 使其具备高可靠性, 能快速适应变化。在设计核心层设备的功能时, 应尽量避免使用数据包过滤、策略路由等降低数据包转发处理的特性, 以优化核心层获得低延迟和良好的可管理性。

汇聚层是核心层和接入层的分界点, 应尽量将资源访问控制、核心层流量的控制等都在汇聚层实施。汇聚层应向核心层隐藏接入层的详细信息, 汇聚层向核心层路由器进行路由宣告时, 仅宣告多个子网地址汇聚而形成的一个网络。另外, 汇聚层也会对接入层屏蔽网络其他部分的信息, 汇聚层路由器可以不向接入路由器宣告其他网络部分的路由, 而仅仅向接入设备宣告自己为默认路由。

接入层为用户提供了在本地网段访问应用系统的能力, 接入层要解决相邻用户之间的互访需要, 并且为这些访问提供足够的带宽。接入层还应该适当负责一些用户管理功能, 包括地址认证、用户认证和计费管理等内容。接入层还负责一些用户信息收集工作, 例如用户的IP地址、MAC地址和访问日志等信息。

第71题, 参考答案: D

解析：

在这个世界上，似乎我们有太多的事情要去做，有太多的事情要去思考，那么需要做的最后一件事就是必须学习新事物。

而用例恰恰可以解决带有需求的问题：如果具有严格声明的需求，则很难描述事件的步骤和序列。

简单地说，用例可以将事件序列的说明放在一起，引导系统完成有用的任务。正如听起来一样简单——这很重要。在面对很多需求的时候，通常不太可能理解需求的作者真正想要系统做什么。在前面的例子中，通过指定特定行为发生的时间和条件，用例减少了需求的不确定性。这样的话，行为的顺序就可以当作是一种需求。用例特别适用于捕捉这类需求。尽管听起来可能很简单，但事实情况是由于常规的需求捕捉方法所侧重的是声明需求和“应该怎么样”的陈述，因此完全无法捕捉系统行为的动态方面。用例是一种简单而有效的表达系统行为的方式，使用这种方式所有参与者都很容易理解。

但是与任何事物一样，用例也存在自己的问题——在用例非常有用的同时，人们也可能误用它，结果就产生了比原来更为糟糕的问题。因此重点在于：如何有效地使用用例，而又不会产生出比原来更严重的问题。

第72题，参考答案：A

解析：

在这个世界上，似乎我们有太多的事情要去做，有太多的事情要去思考，那么需要做的最后一件事就是必须学习新事物。

而用例恰恰可以解决带有需求的问题：如果具有严格声明的需求，则很难描述事件的步骤和序列。

简单地说，用例可以将事件序列的说明放在一起，引导系统完成有用的任务。正如听起来一样简单——这很重要。在面对很多需求的时候，通常不太可能理解需求的作者真正想要系统做什么。在前面的例子中，通过指定特定行为发生的时间和条件，用例减少了需求的不确定性。这样的话，行为的顺序就可以当作是一种需求。用例特别适用于捕捉这类需求。尽管听起来可能很简单，但事实情况是由于常规的需求捕捉方法所侧重的是声明需求和“应该怎么样”的陈述，因此完全无法捕捉系统行为的动态方面。用例是一种简单而有效的表达系统行为的方式，使用这种方式所有参与者都很容易理解。

但是与任何事物一样，用例也存在自己的问题——在用例非常有用的同时，人们也可能误用它，结果就产生了比原来更为糟糕的问题。因此重点在于：如何有效地使用用例，而又不会产生出比原来更严重的问题。

第73题，参考答案：B

解析：

在这个世界上，似乎我们有太多的事情要去做，有太多的事情要去思考，那么需要做的最后一件事就是必须学习新事物。

而用例恰恰可以解决带有需求的问题：如果具有严格声明的需求，则很难描述事件的步骤和序列。

简单地说，用例可以将事件序列的说明放在一起，引导系统完成有用的任务。正如听起来一样简单——这很重要。在面对很多需求的时候，通常不太可能理解需求的作者真正想要系统做什么。在前面的例子中，通过指定特定行为发生的时间和条件，用例减少了需求的

不确定性。这样的话, 行为的顺序就可以当作是一种需求。用例特别适用于捕捉这类需求。尽管听起来可能很简单, 但事实情况是由于常规的需求捕捉方法所侧重的是声明需求和“应该怎么样”的陈述, 因此完全无法捕捉系统行为的动态方面。用例是一种简单而有效的表达系统行为的方式, 使用这种方式所有参与者都很容易理解。

但是与任何事物一样, 用例也存在自己的问题——在用例非常有用的同时, 人们也可能误用它, 结果就产生了比原来更为糟糕的问题。因此重点在于: 如何有效地使用用例, 而又不会产生出比原来更严重的问题。

第74题, 参考答案: B

解析:

在这个世界上, 似乎我们有太多的事情要去做, 有太多的事情要去思考, 那么需要做的最后一件事就是必须学习新事物。

而用例恰恰可以解决带有需求的问题: 如果具有严格声明的需求, 则很难描述事件的步骤和序列。

简单地说, 用例可以将事件序列的说明放在一起, 引导系统完成有用的任务。正如听起来一样简单——这很重要。在面对很多需求的时候, 通常不太可能理解需求的作者真正想要系统做什么。在前面的例子中, 通过指定特定行为发生的时间和条件, 用例减少了需求的不确定性。这样的话, 行为的顺序就可以当作是一种需求。用例特别适用于捕捉这类需求。尽管听起来可能很简单, 但事实情况是由于常规的需求捕捉方法所侧重的是声明需求和“应该怎么样”的陈述, 因此完全无法捕捉系统行为的动态方面。用例是一种简单而有效的表达系统行为的方式, 使用这种方式所有参与者都很容易理解。

但是与任何事物一样, 用例也存在自己的问题——在用例非常有用的同时, 人们也可能误用它, 结果就产生了比原来更为糟糕的问题。因此重点在于: 如何有效地使用用例, 而又不会产生出比原来更严重的问题。

第75题, 参考答案: A

解析:

在这个世界上, 似乎我们有太多的事情要去做, 有太多的事情要去思考, 那么需要做的最后一件事就是必须学习新事物。

而用例恰恰可以解决带有需求的问题: 如果具有严格声明的需求, 则很难描述事件的步骤和序列。

简单地说, 用例可以将事件序列的说明放在一起, 引导系统完成有用的任务。正如听起来一样简单——这很重要。在面对很多需求的时候, 通常不太可能理解需求的作者真正想要系统做什么。在前面的例子中, 通过指定特定行为发生的时间和条件, 用例减少了需求的不确定性。这样的话, 行为的顺序就可以当作是一种需求。用例特别适用于捕捉这类需求。尽管听起来可能很简单, 但事实情况是由于常规的需求捕捉方法所侧重的是声明需求和“应该怎么样”的陈述, 因此完全无法捕捉系统行为的动态方面。用例是一种简单而有效的表达系统行为的方式, 使用这种方式所有参与者都很容易理解。

但是与任何事物一样, 用例也存在自己的问题——在用例非常有用的同时, 人们也可能误用它, 结果就产生了比原来更为糟糕的问题。因此重点在于: 如何有效地使用用例, 而又不会产生出比原来更严重的问题。

下午案例分析答案与解析

第1题：跨步软考[www.kuabu.xyz]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。

跨步软考[www.kuabu.xyz]答案解析：

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。

跨步软考[www.kuabu.xyz]答案解析：

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

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。

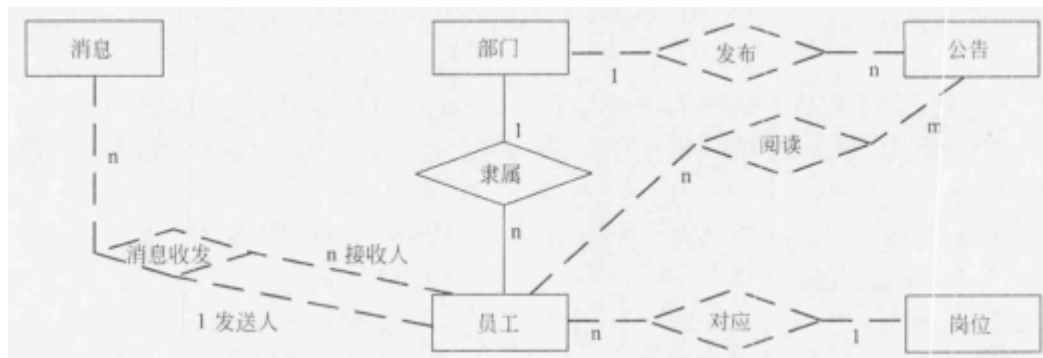
跨步软考[www.kuabu.xyz]答案解析:

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”。

第2题:跨步软考[www.kuabu.xyz]答案解析:

联系名称可不作要求,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。

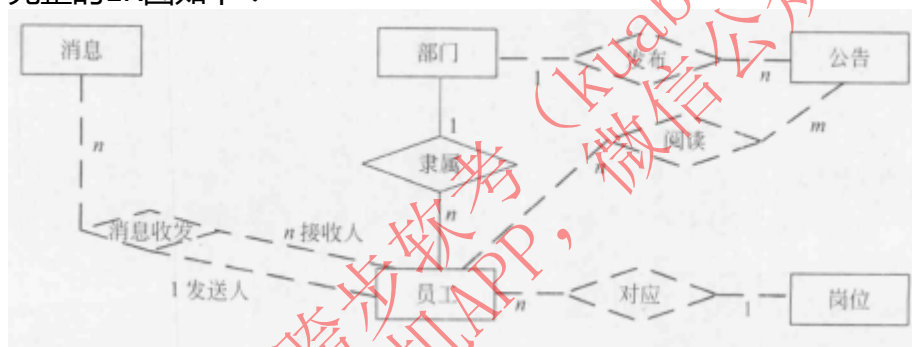
根据题意“一个员工只对应一个岗位, 但一个岗位可对应多名员工”, 可以得出员工与岗位之间的对应联系类型为n:1。

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:



跨步软考[www.kuabu.xyz]答案解析:

- (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, 完整的数据库模式如下:

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。

跨步软考[www.kuabu.xyz]答案解析:

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。

第3题: 跨步软考[www.kuabu.xyz]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”。

跨步软考[www.kuabu.xyz]答案解析:

“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。

跨步软考[www.kuabu.xyz]答案解析:

(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”。

第4题: 跨步软考[www.kuabu.xyz]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 $x[i-1]\neq y[j-1]$ 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“ $i=\max i-\max$ ”。

跨步软考[www.kuabu.xyz]答案解析:

(5)动态规划

(6) $O(m\times n)$ 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 $O(mn)$ 。

跨步软考[www.kuabu.xyz]答案解析:

(7)AB

根据题干和C代码, 计算出下表的值。

		表1 二维数组c						
			B	D	C	A	B	A
		0	1	2	3	4	5	6
A	0	0	0	0	0	0	0	0
B	1	0	0	0	0	1	0	1
C	2	0	1	0	0	0	2	0
A	3	0	0	0	1	0	0	0
D	4	0	0	0	0	2	0	1
A	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

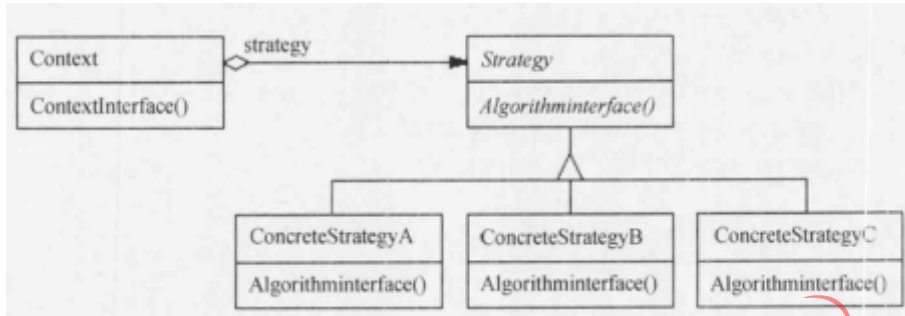
最大值为2。在计算过程中, 我们记录第一个最大值, 即表中阴影部分元素, 因此得到最长公共子串为AB。

第5题: 跨步软考[www.kuabu.xyz]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、CashDiscount和CashReturn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。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、CashDiscount和CashReturn的代码可以看出, 公共操作接口为double acceptCash(double money)。由于不需要父类CashSuper提供任何促销实现方式, 所以这里采用纯虚拟函数。应填入空(1)处的语句是“virtual double acceptCash(double money)=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=new CashNormal()”、“cs=new CashReturn(300, 100)”和“cs=new CashDiscount(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“return cs->acceptCash(money)”。

第6题: 跨步软考[www.kuabu.xyz]答案解析:

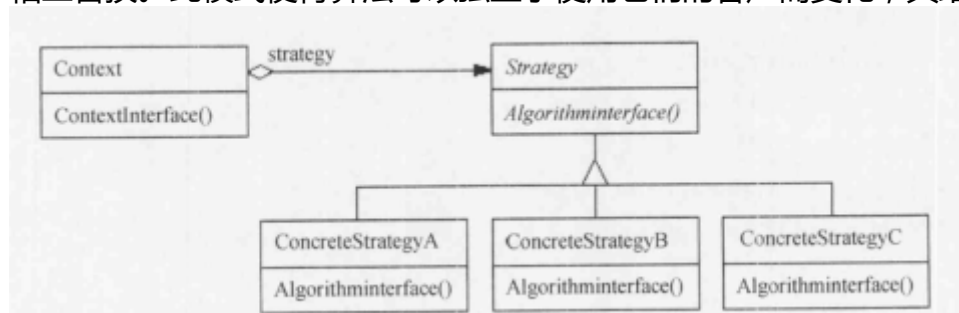
- (1) double acceptCash(double money)
- (2) cs=new CashNormal()
- (3) cs=new CashReturn(300,100)

(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、CashDiscount和CashReturn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、CashDiscount和CashReturn的代码可以看出, 公共操作接口为doubleacceptCash(double money), 因此应填入空(1)处的语句

是“doubleacceptCash(double money)”。

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=new CashNormal()”、“cs=new CashDiscount(0.8)”和“cs=new CashReturn(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“return cs.acceptCash(money)”。