

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 上午试卷 综合知识

（考试时间 150 分钟）

1. 在答题卡的指定位置上正确写入你的姓名和准考证号，并用正规 2B 铅笔在你写入的准考证号下填涂准考证号。
2. 本试卷的试题中共有 75 个空格，需要全部解答，每个空格 1 分，满分 75 分。
3. 每个空格对应一个序号，有 A、B、C、D 四个选项，请选择一个最恰当的选项作为解答，在答题卡相应序号下填涂该选项。
4. 解答前务必阅读例题和答题卡上的例题填涂样式及填涂注意事项。解答时用正规 2B 铅笔正确填涂选项，如需修改，请用橡皮擦干净，否则会导致不能正确评分。

试题一 CPU 是在()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

试题二 虚拟存储体系由()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

试题三 浮点数能够表示的数的范围是由其()的位数决定的。

- A. 尾数 B. 阶码 C. 数符 D. 阶符

试题四 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()。

- A. 隐含寻址 B. 寄存器寻址 C. 立即寻址 D. 直接寻址

试题五 内存按字节编址从 B3000H 到 DABFFH 的区域其存储容量为()。

A. 123KB B. 159KB C. 163KB D. 194KB

试题六 CISC 是()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

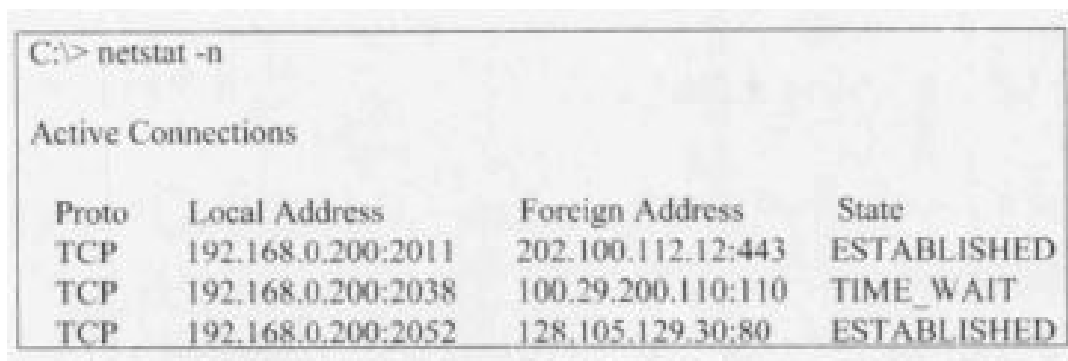
试题七 ()不属于主动攻击。

A. 流量分析 B. 重放 C. IP 地址欺骗 D. 拒绝服务

试题八 防火墙不具备()功能。

A. 记录访问过程 B. 查毒 C. 包过滤 D. 代理

试题九 根据下图所示的输出信息，可以确定的是：()



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

试题一十 以下著作权权利中，()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

试题一十一 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权人。王某的行为()。

- 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

试题一十二 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()。

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

试题一十三 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()种不同的颜色。

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

试题一十四 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()。

- A. 种数 B. 纯度 C. 感觉 D. 存储量

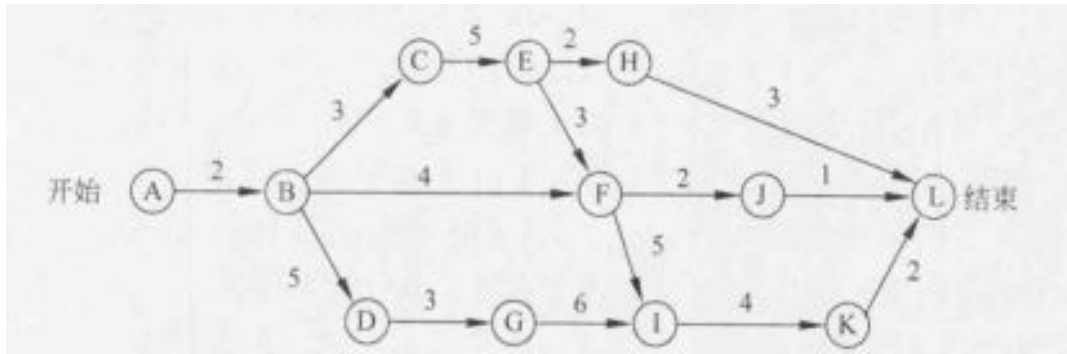
试题一十五 (第 1 空)若用户需求不清晰且经常发生变化,但系统规模不太大且不太复杂,则最适宜采用()开发方法,对于数据处理领域的问题,若系统规模不太大且不太复杂,需求变化也不大,则最适宜采用()开发方法。

- A. 结构化 B. Jackson C. 原型化 D. 面向对象

试题一十六 (第 2 空)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()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()开发方法。

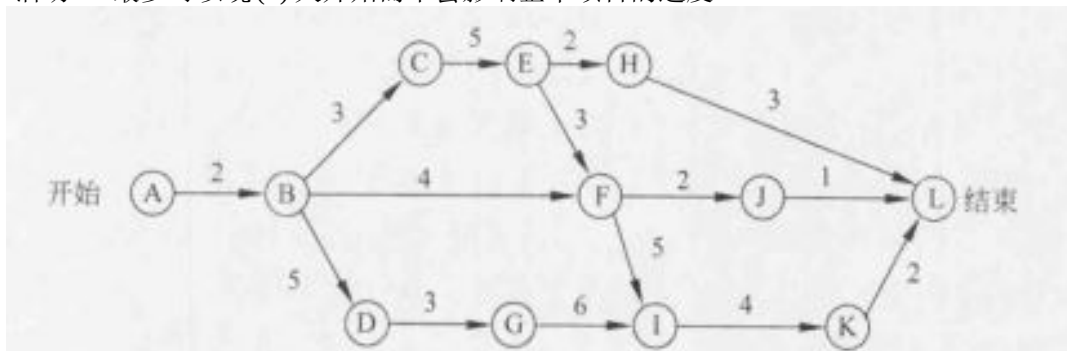
- A. 结构化 B. Jackson C. 原型化 D. 面向对象

试题一十七 (第 1 空)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()天。活动 BD 最多可以晚()天开始而不会影响整个项目的进度。



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

试题一十八 (第 2 空) 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() 天。活动 BD 最多可以晚() 天开始而不会影响整个项目的进度。



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

试题一十九 以下关于软件项目管理中人员管理的叙述, 正确的是()。

- 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

试题二十 (第 1 空) 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, () 并不是每个编译器都必需的, 与编译器相比, 解释器()。

- 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

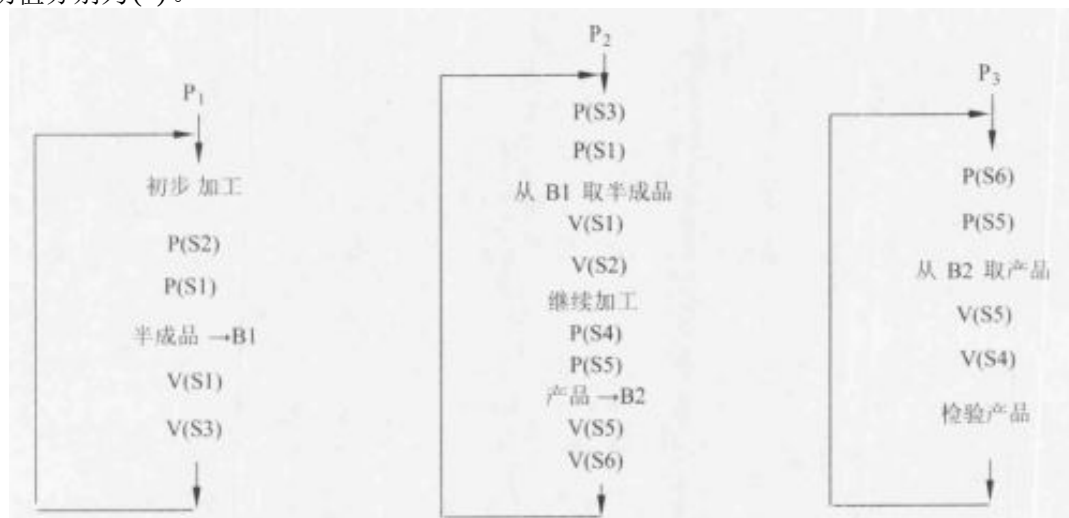
试题二十一 (第 2 空)编译器和解释器是两种基本的高级语言处理程序。编译器对高级语言源程序的处理过程可以划分为词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成等阶段，其中，()并不是每个编译器都必需的，与编译器相比，解释器()。

- 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

试题二十二 表达式采用逆波兰式表示时，利用()进行求值。

- A. 栈 B. 队列 C. 符号表 D. 散列表

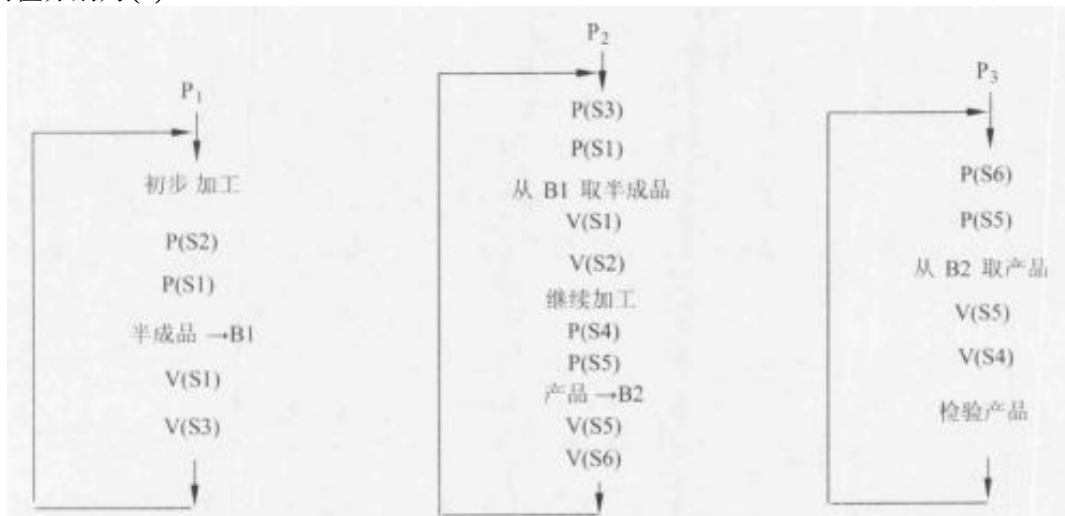
试题二十三 (第 1 空)某企业的生产流水线上有 2 名工人 P1 和 P2，1 名检验员 P3。P1 将初步加工的半成品放入半成品箱 B1；P2 从半成品箱 B1 取出继续加工，加工好的产品放入成品箱 B2；P3 从成品箱 B2 去除产品校验。假设 B1 可存放 n 件半成品，B2 可存放 m 件产品，并设置 6 个信号量 S1、S2、S3、S4、S5 和 S6，且 S3 和 S6 的初值都为 0。采用 PV 操作实现 P1、P2 和 P3 的同步模型如下图所示，则信号量 S1 和 S5()；S2、S4 的初值分别为()。



- 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

试题二十四 (第 2 空)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()；S2、S4 的初值分别为()。



- A. n、0 B. m、0 C. m、n D. n、m

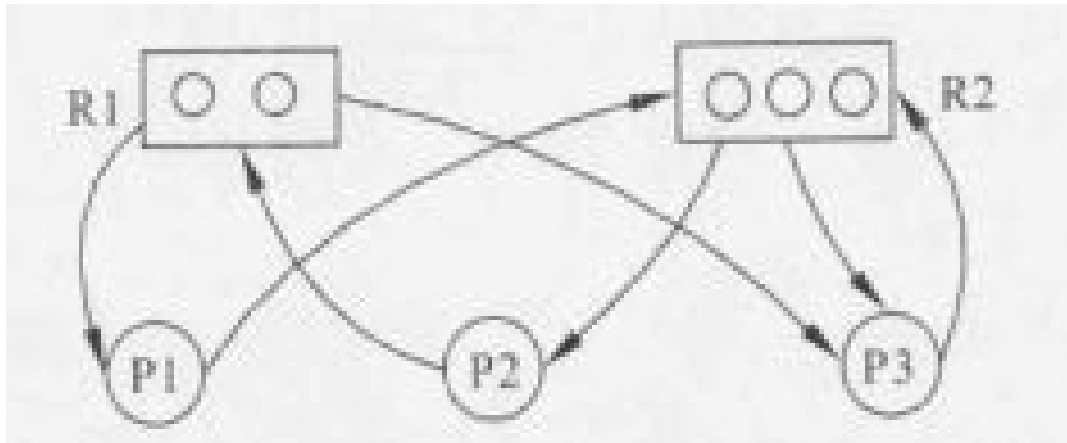
试题二十五 (第 1 空)假设磁盘块与缓冲区大小相同，每个盘块读入缓冲区的时间为 $15\mu s$ ，由缓冲区送至用户区的时间是 $5\mu s$ ，在用户区内系统对每块数据的处理时间为 $1\mu s$ ，若用户需要将大小为 10 个磁盘块的 Doc1 文件逐块从磁盘读入缓冲区，并送至用户区进行处理，那么采用单缓冲区需要花费的时间为() μs ；采用双缓冲区需要花费的时间为() μs 。

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

试题二十六 (第 2 空)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() μs ；采用双缓冲区需要花费的时间为() μs 。

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

试题二十七 在如下所示的进程资源图中，()。



- 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

试题二十八 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()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

试题二十九 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()过程模型。

- A. 瀑布
- B. 原型
- C. 螺旋
- D. 喷泉

试题三十 ()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

试题三十一 在 ISO/IEC 软件质量模型中，易使用性的子特性不包括()。

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

试题三十二 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()。

- 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

试题三十三 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()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

试题三十四 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()工具。

A. 软件开发 B. 软件维护 C. 软件管理 D. 软件支持

试题三十五 (第 1 空)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()个测试用例。采用 McCabe 度量法算出该程序的环路复杂性为()。

```
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

试题三十六 (第 2 空)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()个测试用例。采用 McCabe 度量法算出该程序的环路复杂性为()。


```
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

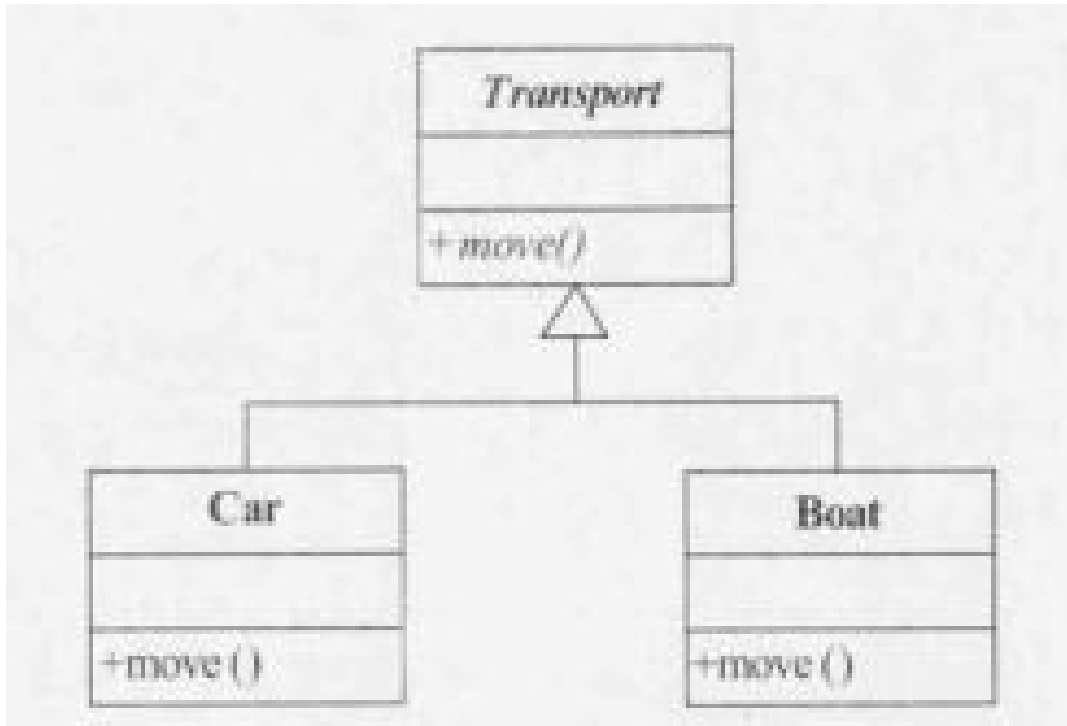
试题三十七 (第 1 空)在面向对象的系统中, 对象是运行时实体, 其组成部分不包括();
一个类定义了一组大体相似的对象, 这些对象共享()。

- A. 消息 B. 行为(操作) C. 对象名 D. 状态

试题三十八 (第 2 空)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();
一个类定义了一组大体相似的对象, 这些对象共享()。

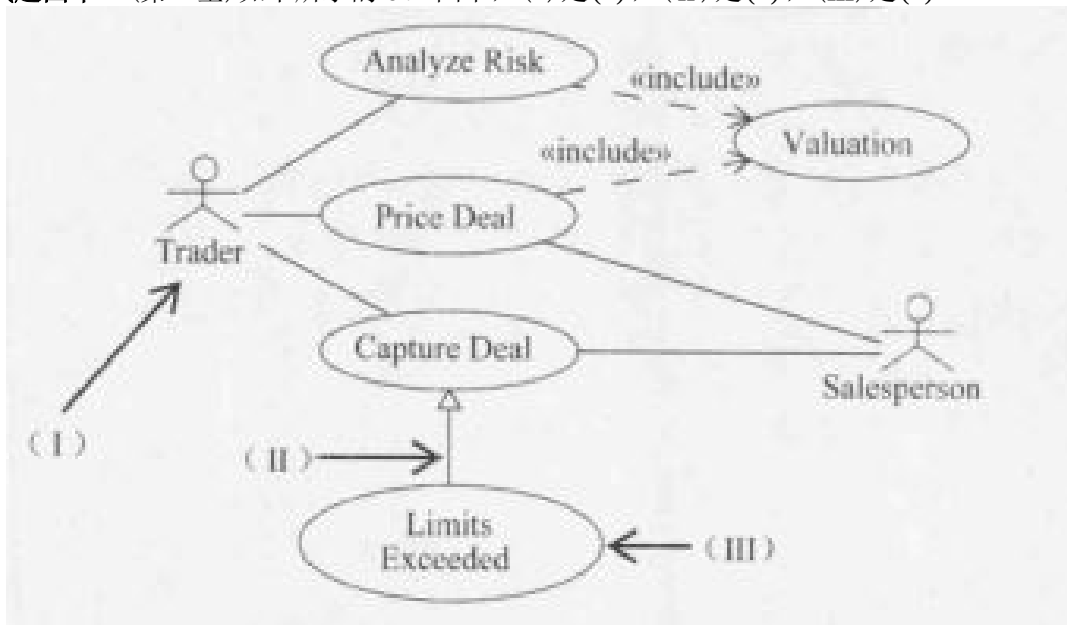
- A. 属性和状态 B. 对象名和状态 C. 行为和多重度 D. 属性和行为

试题三十九 如下所示的 UML 类图中, Car 和 Boat 类中的 move()方法()了 Transport 类中的 move()方法。



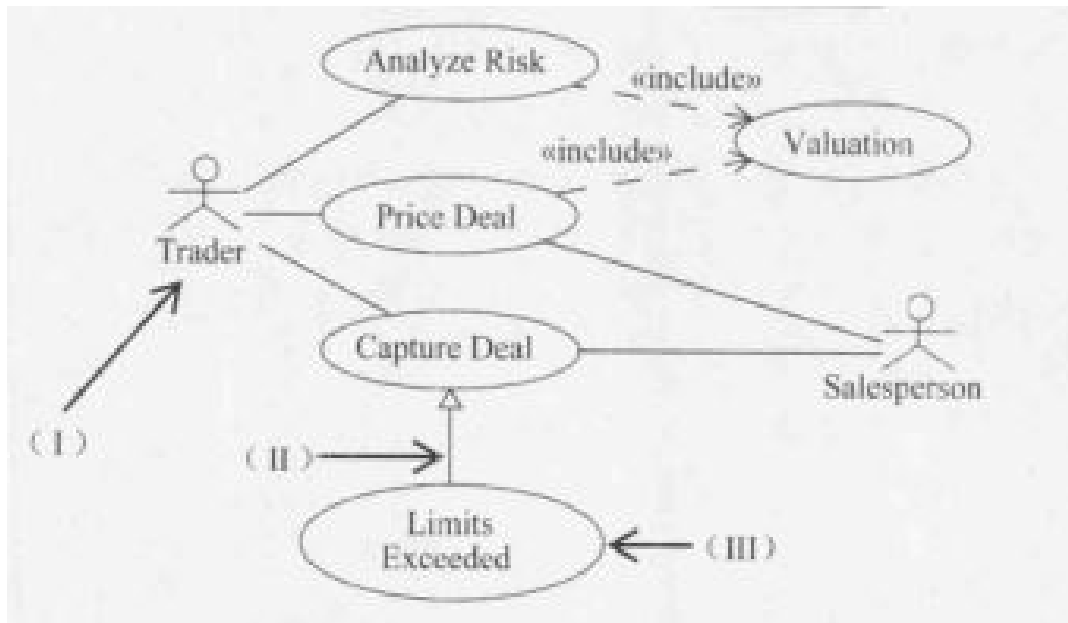
A. 继承 B. 覆盖(重置) C. 重载 D. 聚合

试题四十 (第 1 空)如下所示的 UML 图中, (I)是(), (II)是(), (III)是()。



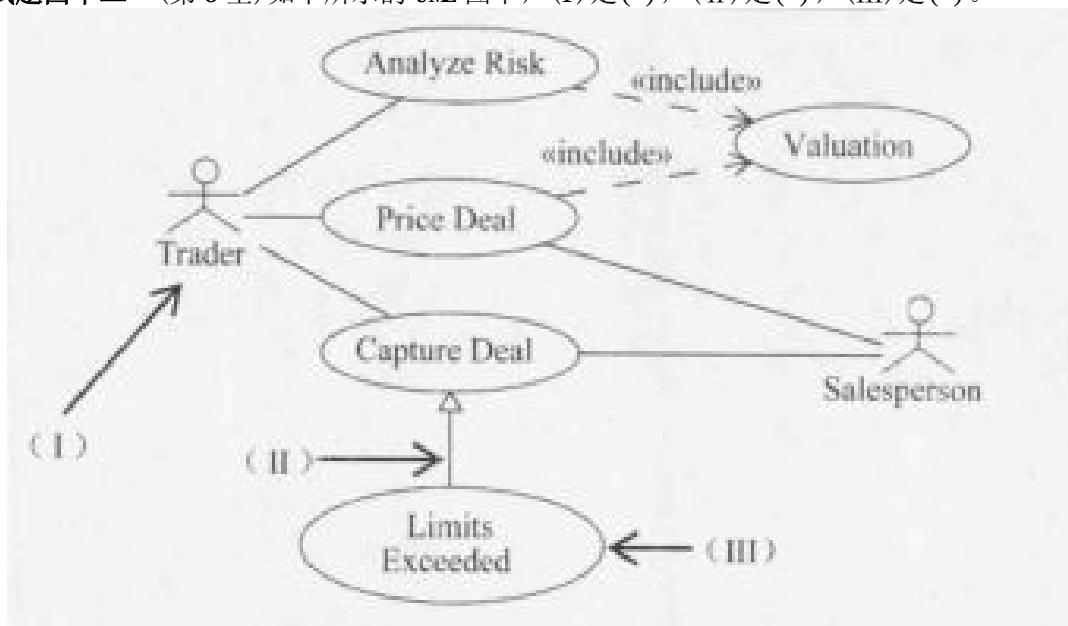
A. 参与者 B. 用例 C. 泛化关系 D. 包含关系

试题四十一 (第 2 空)如下所示的 UML 图中, (I)是(), (II)是(), (III)是()。



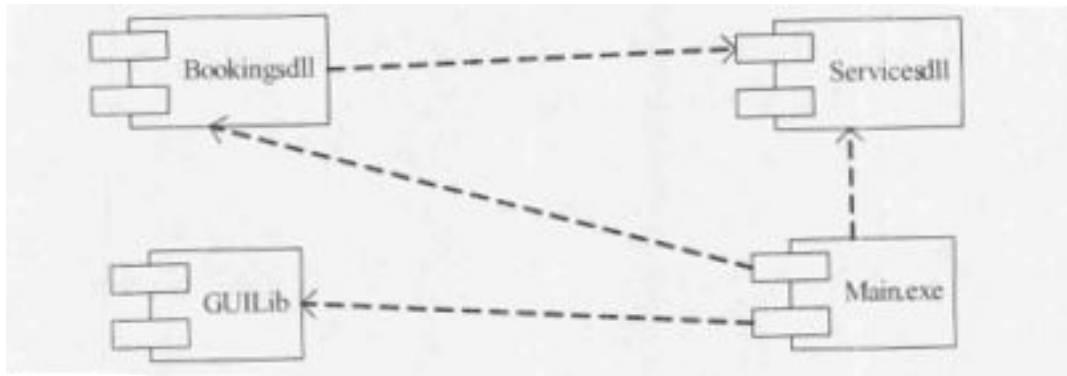
A. 参与者 B. 用例 C. 泛化关系 D. 包含关系

试题四十二 (第3空)如下所示的UML图中，(I)是()，(II)是()，(III)是()。



A. 参与者 B. 用例 C. 泛化关系 D. 包含关系

试题四十三 下所示为UML()。



- A. 类图 B. 部署图 C. 组件图 D. 网络图

试题四十四 以下关于 Singleton(单例)设计模式的叙述中, 不正确的是()。

- 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

试题四十五 (第 1 空)()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; ()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()模式最适合。

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

试题四十六 (第 2 空)()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; ()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()模式最适合。

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

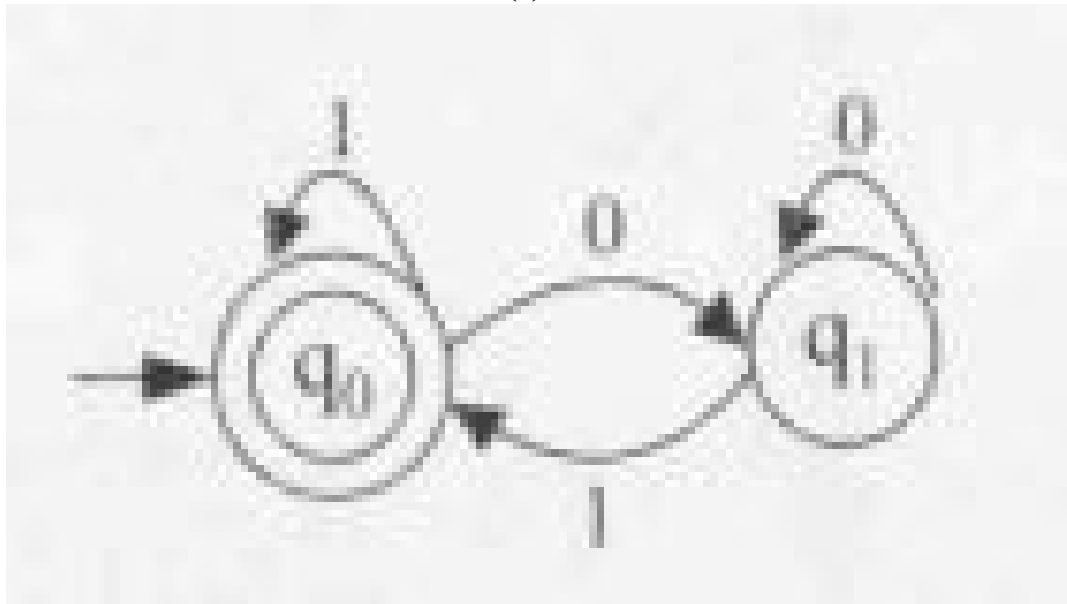
试题四十七 (第 3 空)()设计模式能够动态地给一个对象添加一些额外的职责而无需修改此对象的结构; ()设计模式定义一个用于创建对象的接口, 让子类决定实例化哪一个类; 欲使一个后端数据模型能够被多个前端用户界面连接, 采用()模式最适合。

- A. 装饰器(Decorator) B. 享元(Flyweight)
C. 观察者(Observer) D. 中介者(Mediator)

试题四十八 某程序运行时陷入死循环, 则可能的原因是程序中存在()。

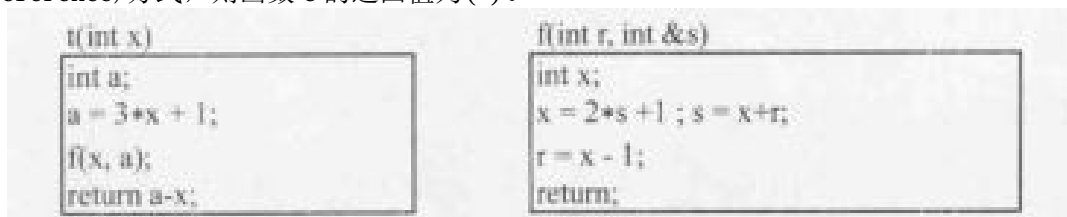
- A. 词法错误 B. 语法错误 C. 动态的语义错误 D. 静态的语义错误

试题四十九 某非确定的有限自动机(NFA)的状态转换图如下图所示(q_0 既是初态也是终态)。以下关于该NFA的叙述中,正确的是()。



- 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

试题五十 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()。



- A. 33 B. 22 C. 11 D. 负数

试题五十一 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()。

- 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

试题五十二 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,(),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

试题五十三 (第 1 空)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: ()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; ()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

试题五十四 (第 2 空)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: ()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; ()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

试题五十五 (第 1 空)设有关系模式 $R(A1, A2, A3, A4, A5, A6)$, 其中: 函数依赖集 $F=\{A1 \rightarrow A2, A1A3 \rightarrow A4, A5A6 \rightarrow A1, A2A5 \rightarrow A6, A3A5 \rightarrow A6\}$, 则()是关系模式 R 的一个主键, R 规范化程度最高达到()。
A. $A1A4$ B. $A2A4$ C. $A3A5$ D. $A4A5$

试题五十六 (第 2 空)设有关系模式 $R(A1, A2, A3, A4, A5, A6)$, 其中: 函数依赖集 $F=\{A1 \rightarrow A2, A1A3 \rightarrow A4, A5A6 \rightarrow A1, A2A5 \rightarrow A6, A3A5 \rightarrow A6\}$, 则()是关系模式 R 的一个主键, R 规范化程度最高达到()。
A. 1NF B. 2NF C. 3NF D. BCNF

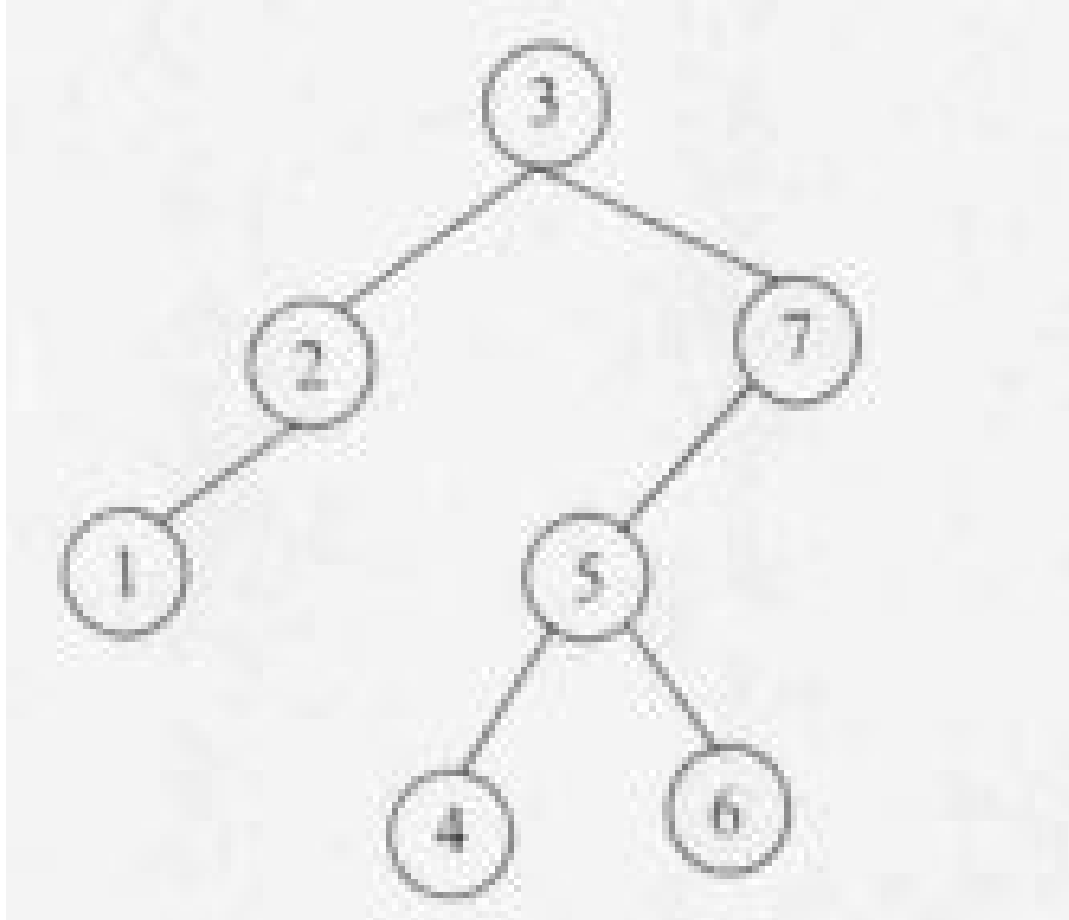
试题五十七 对于一个长度为 $n(n>1)$ 且元素互异的序列, 每其所有元素依次通过一个初始为空的栈后, 再通过一个初始为空的队列。假设队列和栈的容量都足够大, 且只要栈非空就可以进行出栈操作, 只要队列非空就可以进行出队操作, 那么以下叙述中, 正确的是()。
A. 出队序列和出栈序列一定互为逆序 B. 出队序列和出栈序列一定相同
C. 入栈序列与入队序列一定相同 D. 入栈序列与入队序列一定互为逆序

试题五十八 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 $B[k] (1 \leq k \leq 3*n-2)$ 中, 则 k 与 i 、 j 的对应关系是()。

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

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

试题五十九 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 7654321，则遍历方式是()。



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

试题六十 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()、 $A[19]$ 。

- 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. $A[28]$ 、 $A[18]$ 、 $A[22]$ 、 $A[20]$

试题六十一 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()。

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

试题六十二 (第 1 空) 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()。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()。

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

试题六十三 (第 2 空) 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()。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()。

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

试题六十四 (第 1 空) 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()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()排序算法。

A. 插入 B. 归并 C. 快速 D. 计数

试题六十五 (第 2 空) 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()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()排序算法。

A. 插入 B. 归并 C. 快速 D. 计数

试题六十六 集线器与网桥的区别是: ()。

- 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

试题六十七 POP3 协议采用()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

试题六十八 TCP 使用的流量控制协议是()。

- 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

试题六十九 以下 4 种路由中，()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

试题七十 以下关于层次化局域网模型中核心层的叙述，正确的是()。

- 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

试题七十一 (第 1 空)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 () 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 () 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 () requirement capture approaches, with their

emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the () 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 (). 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

试题七十二 (第2空) 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 () 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 () 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 () requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the () 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 (). 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

试题七十三 (第3空) 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 () 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 () 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 () requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the () 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 (). 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

试题七十四 (第4空) 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 () 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 () 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 () requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the () 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 (). 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. statics B. nature C. dynamics D. originals

试题七十五 (第5空) 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 () 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 () 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 () requirement capture approaches, with their emphasis on declarative requirements and "shall" statements, completely fail to capture fail to capture the () 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 (). 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. misapplied B. applied C. used D. powerful

试题一 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。

试题二 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。

试题三 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。

试题四 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。

试题五 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。

试题六 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。

试题七 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。

试题八 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。

试题九 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。

试题一十 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。

试题一十一 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。

试题一十二 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)。

试题一十三 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的颜色数，或者确定灰度图像的每个像素可能的灰度级数。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，贝最大颜色数目为 $2^{4+4+2}=2^{10}=1024$ ，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。

试题一十四 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。

试题一十五 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。

试题一十六 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。

试题一十七 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。

试题一十八 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。

试题一十九 答案： A 解析：

本题考查软件项目管理的基础知识。

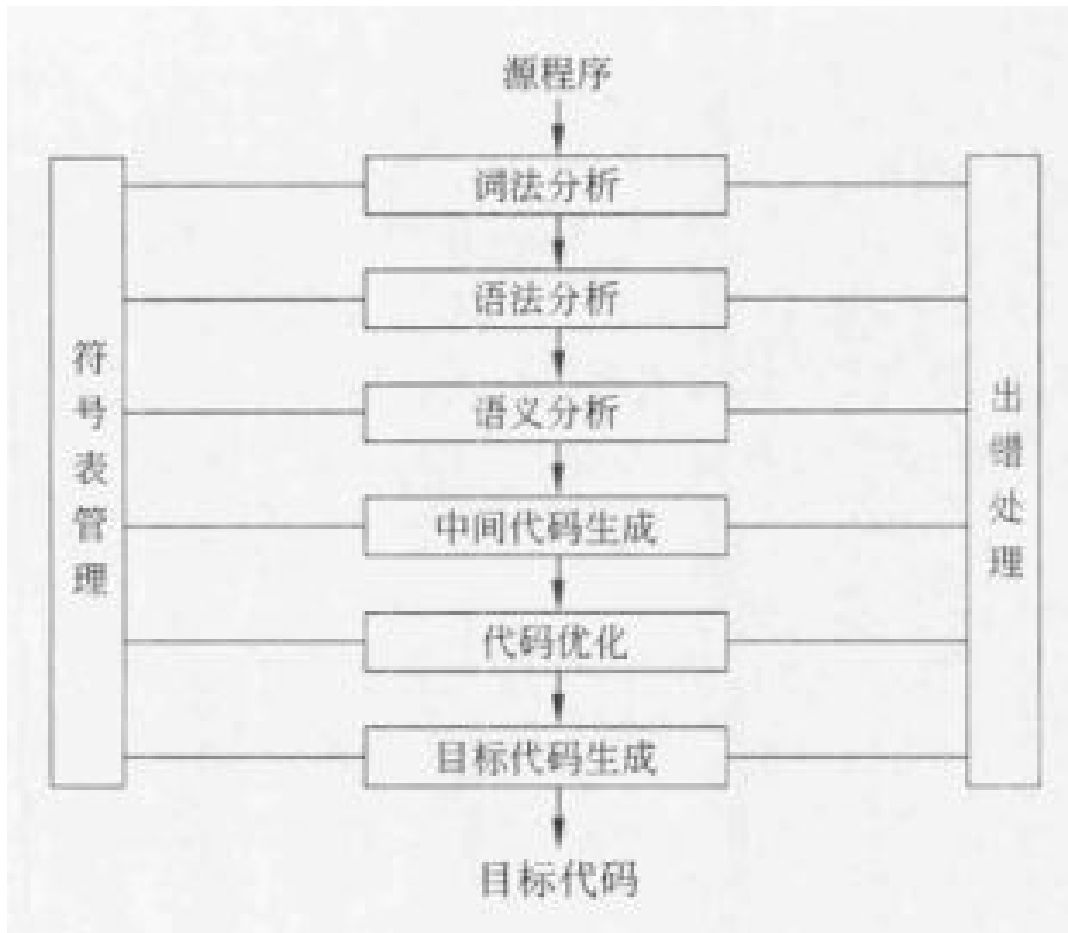
人员管理是软件项目管理的一个重要部分，在组织开发团队时，应该考虑开发人员的工作能力、知识背景、工作风格、兴趣爱好等多方面的因素。每个成员的工作任务分配清楚，不应该参与所有阶段的工作。当项目进度滞后于项目计划时，增加开发人员不一定可以加快开发进度。

试题二十 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。



试题二十一 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。



试题二十二 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。

试题二十三 答案： C 解析：

信号量 S_1 是一个互斥信号量，表示半成品箱 B_1 当前无工人(生产者)使用，所以初值为 1。信号量 S_5 也是一个互斥信号量，表示成品箱 B_2 当前无工人或检验员使用，所以初值为 1。

试题二十四 答案： D 解析：

信号量 S_2 表示半成品箱 B1 的容量，故 S_2 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 $P(S_2)$ ，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 $V(S_2)$ 释放空间。

同理，信号量 S_4 表示成品箱 B2 的容量，故 S_4 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 $P(S_4)$ ，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 $V(S_4)$ 释放空间。

试题二十五 答案： D 解析：

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。

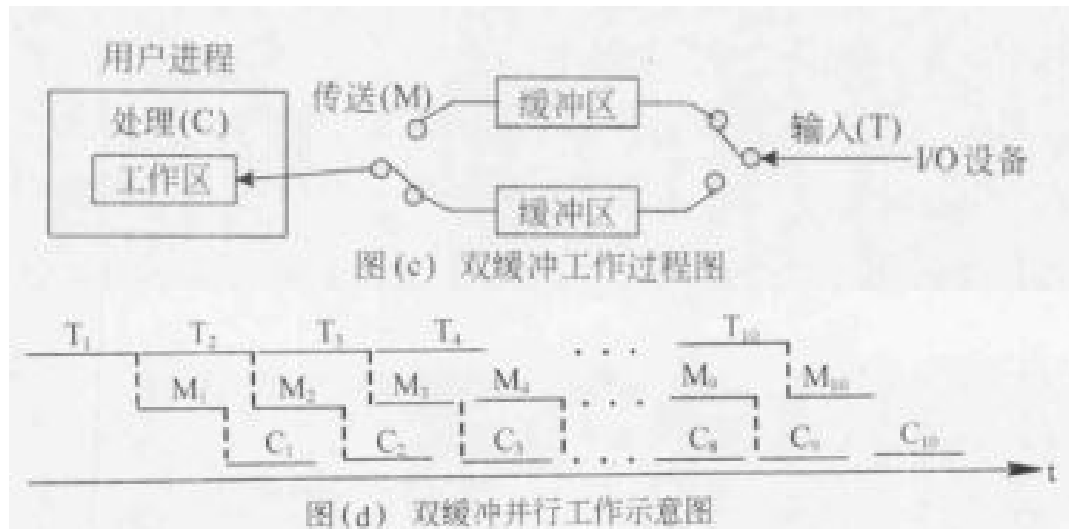
当第一块数据送入用户工作区后，缓冲区是空闲的就可以传送第二块数据。这样第一块数据的处理 C_1 与第二块数据的输入 T_2 是可以并行的，依次类推，如图(b)所示。系统对每一块数据的处理时间为： $\max(C, T) + M$ 。因为，当 $T > C$ 时，处理时间为 $M + T$ ；当 T



试题二十六 答案： C 解析：

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 $C < T$ ，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 15，采用双缓冲需要花费的时间为 $15 \times 10 + 5 + 1 = 156$ 。



试题二十七 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、P3 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。

试题二十八 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。

试题二十九 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。

试题三十 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。

试题三十一 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。

试题三十二 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。

试题三十三 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。

试题三十四 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。

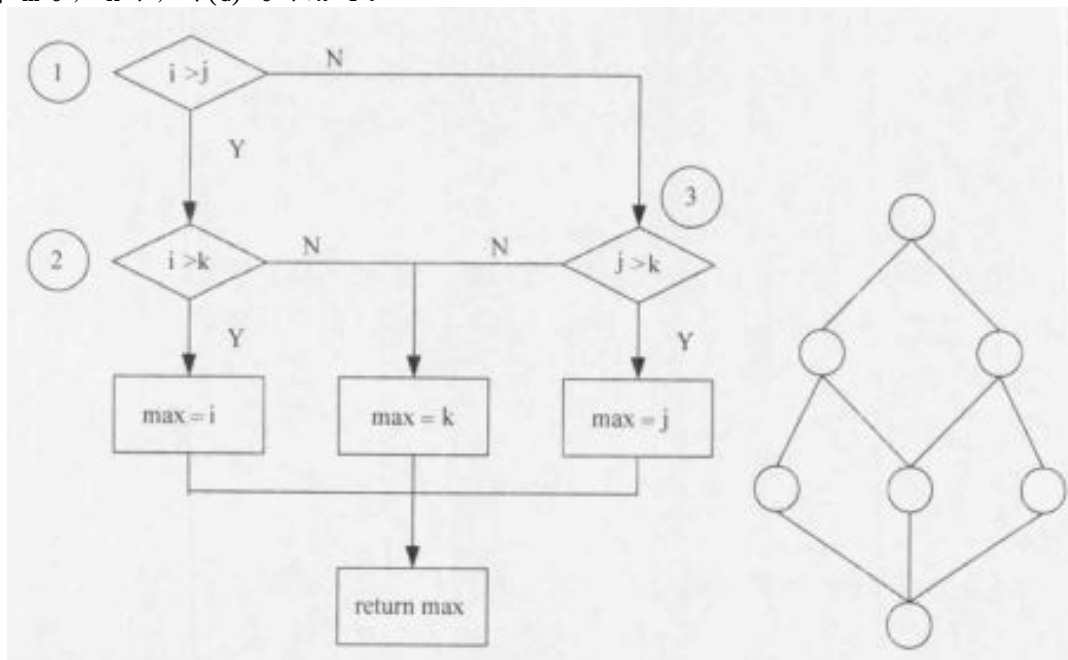
试题三十五 答案： B 解析：

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 $m=9$ ， $n=7$ ， $V(G)=9-7+2=4$ 。



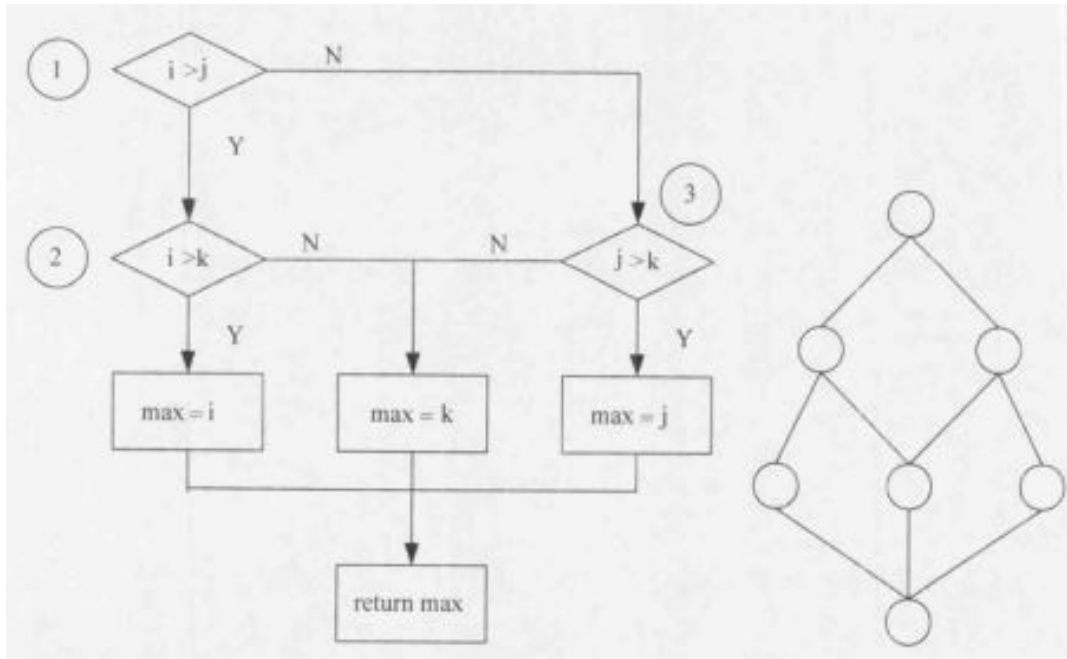
试题三十六 答案： D 解析：

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 $m=9$ ， $n=7$ ， $V(G)=9-7+2=4$ 。



试题三十七 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。

试题三十八 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。

试题三十九 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。

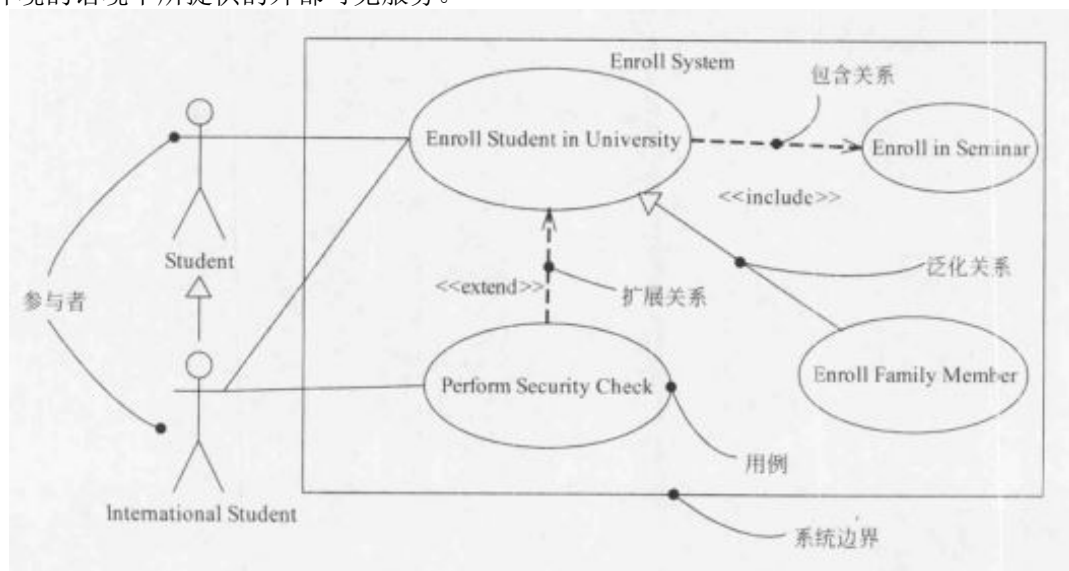
一个类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的共同行为和属性。有些类之间存在一般和特殊关系，即一些类是某个类的特殊情况，某个类是一些类的一般情况，即继承关系。继承是父类和子类之间共享数据和方法的机制。父类描述了这些子类的公共属性和方法。一个子类可以继承它的父类(或祖先类)中的属性和方法，这些属性和操作在子类中不必定义，子类中还可以定义自己的属性和方法，也可以重新定义父类中已经定义的方法，即重置或覆盖(overriding)。UML类图中，如果父类中已有方法名在子类中不出现，表示子类继承父类中的方法；如果父类中已有方法名在子类中出现，就表示子类在继承父类接口定义的前提下，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

试题四十 答案： A 解析：

本题考查统一建模语言(UML)的基本知识。

用例图(usecase 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。

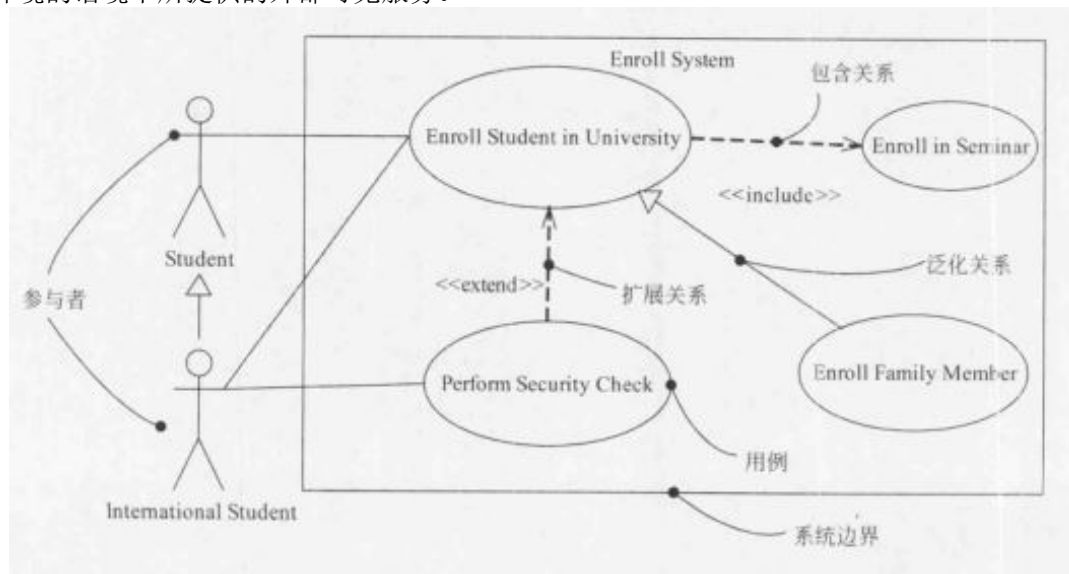


试题四十一 答案： C 解析：

本题考查统一建模语言 (UML) 的基本知识。

用例图 (usecase 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。

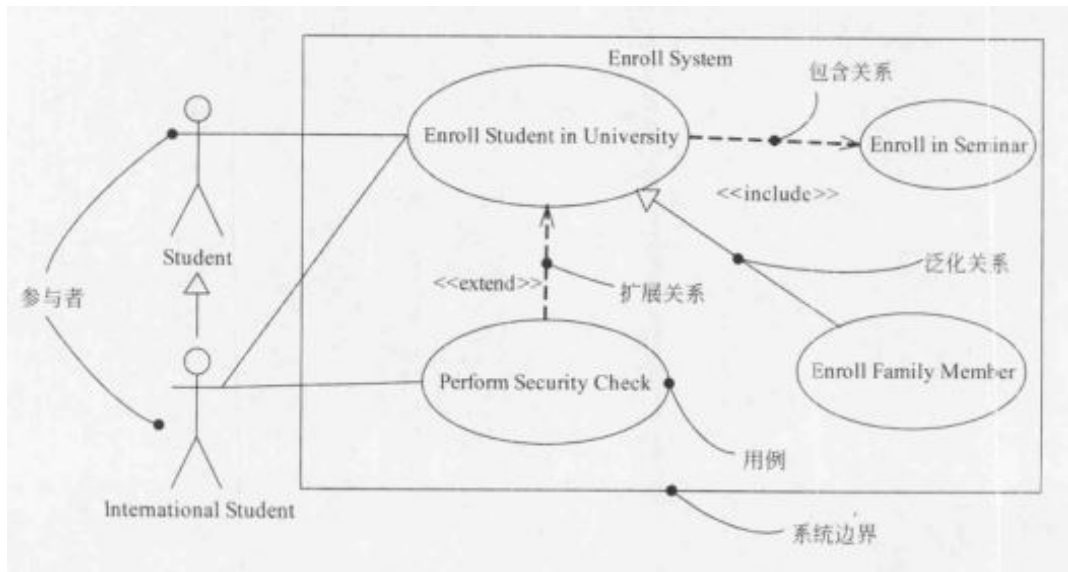


试题四十二 答案： B 解析：

本题考查统一建模语言 (UML) 的基本知识。

用例图 (usecase 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。



试题四十三 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。

试题四十四 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。

试题四十五 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，Fapde 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 Fapde 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(FactoryMethod)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类的時候；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。

试题四十六 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，Fapde 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 Fapde 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(FactoryMethod)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。

试题四十七 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，Fapde 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 Fapde 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(FactoryMethod)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。

试题四十八 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。

试题四十九 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。

试题五十 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，也可以简化管理为：在被调用函数中对形参的修改等同于是对实参进行修改。

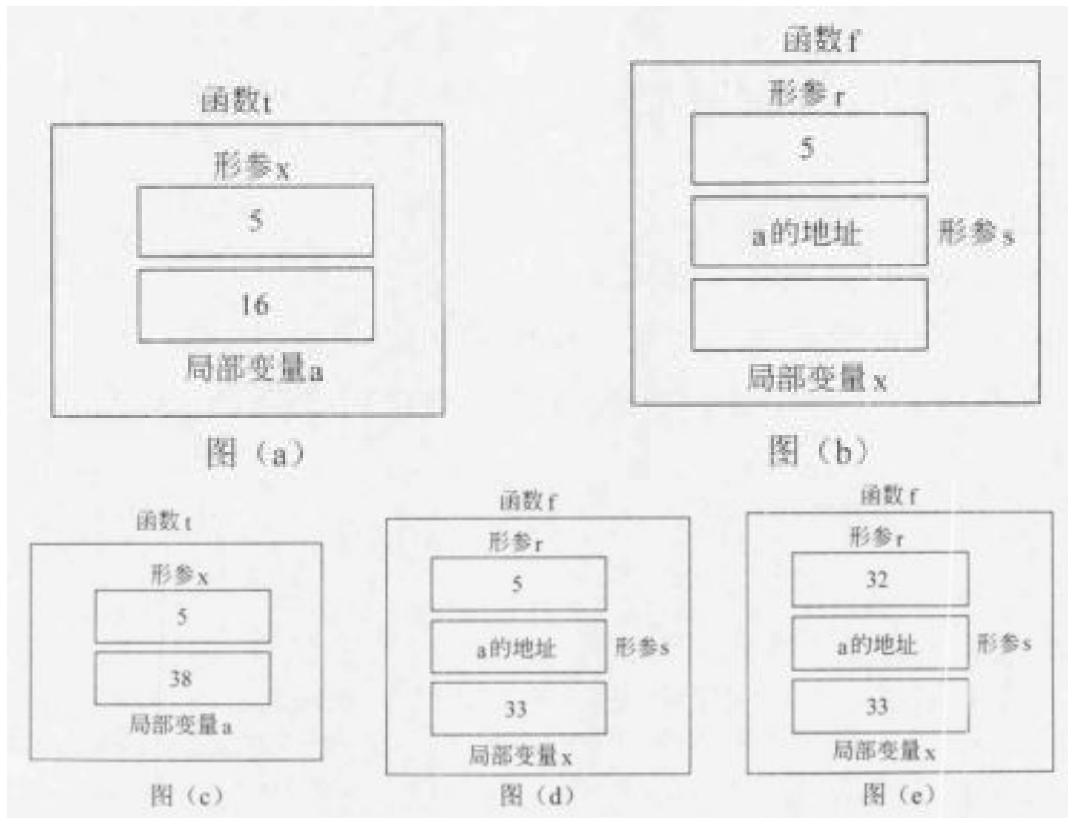
当函数 t 和 f 运行时，其每个形参和局部变量都有各自的存储单元，下面图中矩形框表示存储单元。

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。



试题五十一 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。

试题五十二 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，若包含多值属性，通常一个实体对应一个关系模式。对实体的多值属性，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。

试题五十三 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。

试题五十四 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。

试题五十五 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。

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 。

试题五十六 答案： B 解析：

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。

试题五十七 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。

试题五十八 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

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 $k = 2i + j - 3 + 1$ 。

试题五十九 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。

试题六十 答案： B 解析：

本题考查数据结构基础知识。

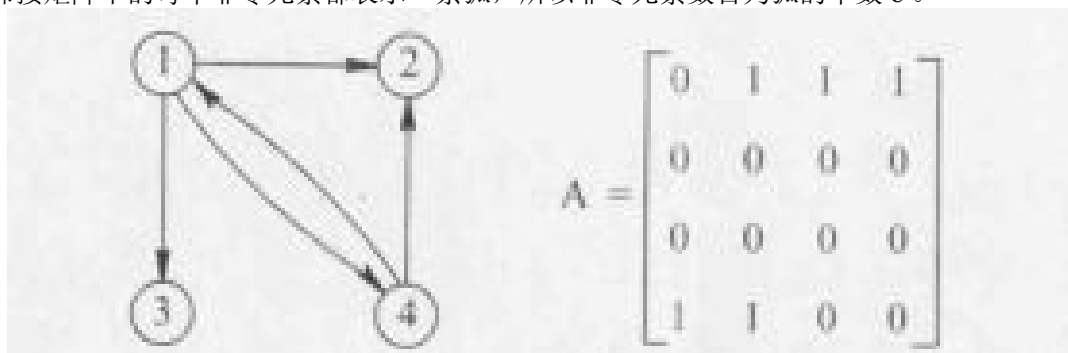
对 55 个元素构成的有序表进行折半查找时，可用判定树描述查找过程。由于 $A[19]$ 小于中间元素 $A[28]$ ，所以判定树的左分支如下图所示。从中可知，查找过程中参与比较的元素分别为 $A[28]$ 、 $A[14]$ 、 $A[21]$ 、 $A[17]$ 、 $A[19]$ 。

试题六十一 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 。



试题六十二 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 $\theta(n^{\sup 3})$ 。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。

试题六十三 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 $\theta(n^{\sup 3})$ 。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。

试题六十四 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。

试题六十五 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。

试题六十六 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。

试题六十七 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。

试题六十八 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。

试题六十九 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。

试题七十 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。

试题七十一 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。

试题七十二 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。

试题七十三 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。

试题七十四 答案： C 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。

试题七十五 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。



苹果 扫码或应用市场搜索“软考
真题”下载获取更多试卷



安卓 扫码或应用市场搜索“软考
真题”下载获取更多试卷