全国计算机技术与软件专业技术资格（水平）考试

2009 年下半年 软件设计师 上午试卷

（考试时间 9：00～11：30 共 150 分钟）

请按下述要求正确填写答题卡

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。

例题

● 2009 年下半年全国计算机技术与软件专业技术资格（水平）考试日期是（88）月（89）日。

（88）A. 12　　B. 11　　C. 10　　D. 9

（89）A. 11　　B. 12　　C. 13　　D. 14

因为考试日期是“11 月 14 日”，故（88）选 B，（89）选 D，应在答题卡序号 88 下对 B 填涂，在序号 89 下对 D 填涂（参看答题卡）。

● 以下关于 CPU 的叙述中，错误的是__（1）__。

（1）A. CPU 产生每条指令的操作信号并将操作信号送往相应的部件进行控制
B. 程序计数器 PC 除了存放指令地址，也可以临时存储算术/逻辑运算结果
C. CPU 中的控制器决定计算机运行过程的自动化
D. 指令译码器是 CPU 控制器中的部件

● 以下关于 CISC（Complex Instruction Set Computer，复杂指令集计算机）和 RISC（Reduced Instruction Set Computer，精简指令集计算机）的叙述中，错误的是__（2）__。

（2）A. 在 CISC 中，其复杂指令都采用硬布线逻辑来执行
B. 采用 CISC 技术的 CPU，其芯片设计复杂度更高
C. 在 RISC 中，更适合采用硬布线逻辑执行指令
D. 采用 RISC 技术，指令系统中的指令种类和寻址方式更少

● 浮点数的一般表示形式为 $N = 2^E \times F$，其中 E 为阶码，F 为尾数。以下关于浮点表示的叙述中，错误的是__（3）__。两个浮点数进行相加运算，应首先__（4）__。

（3）A. 阶码的长度决定浮点表示的范围，尾数的长度决定浮点表示的精度
B. 工业标准 IEEE754 浮点数格式中阶码采用移码、尾数采用原码表示
C. 规格化指的是阶码采用移码、尾数采用补码
D. 规格化表示要求将尾数的绝对值限定在区间[0.5, 1)

（4）A. 将较大的数进行规格化处理　　B. 将较小的数进行规格化处理
C. 将这两个数的尾数相加　　D. 统一这两个数的阶码

● 以下关于校验码的叙述中，正确的是__（5）__。

（5）A. 海明码利用多组数位的奇偶性来检错和纠错
B. 海明码的码距必须大于等于 1
C. 循环冗余校验码具有很强的检错和纠错能力
D. 循环冗余校验码的码距必定为 1

● 以下关于 Cache 的叙述中，正确的是__（6）__。

（6）A. 在容量确定的情况下，替换算法的时间复杂度是影响 Cache 命中率的关键因素
B. Cache 的设计思想是在合理成本下提高命中率
C. Cache 的设计目标是容量尽可能与主存容量相等
D. CPU 中的 Cache 容量应大于 CPU 之外的 Cache 容量

● 网络安全体系设计可从物理线路安全、网络安全、系统安全、应用安全等方面来进行。其中，数据库容灾属于__（7）__。

（7）A. 物理线路安全和网络安全　　B. 物理线路安全和应用安全
C. 系统安全和网络安全　　D. 系统安全和应用安全

● 包过滤防火墙对数据包的过滤依据不包括（8）。

（8）A. 源 IP 地址　　B. 源端口号
　　C. MAC 地址　　D. 目的 IP 地址

● 某网站向 CA 申请了数字证书，用户通过（9）来验证网站的真伪。

（9）A. CA 的签名　　B. 证书中的公钥
　　C. 网站的私钥　　D. 用户的公钥

● 下列智力成果中，能取得专利权的是（10）。

（10）A. 计算机程序代码　　B. 游戏的规则和方法
　　C. 计算机算法　　D. 用于控制测试过程的程序

● 软件权利人与被许可方签订一份软件使用许可合同。若在该合同约定的时间和地域范围内，软件权利人不得再许可任何第三人以此相同的方法使用该项软件，但软件权利人可以自己使用，则该项许可使用是（11）。

（11）A. 独家许可使用　　B. 独占许可使用
　　C. 普通许可使用　　D. 部分许可使用

● 多媒体中的“媒体”有两重含义，一是指存储信息的实体；二是指表达与传递信息的载体。（12）是存储信息的实体。

（12）A. 文字、图形、磁带、半导体存储器
　　B. 磁盘、光盘、磁带、半导体存储器
　　C. 文字、图形、图像、声音
　　D. 声卡、磁带、半导体存储器

● RGB8:8:8 表示一帧彩色图像的颜色数为（13）种。

（13）A. 2^3　　B. 2^8　　C. 2^{24}　　D. 2^{512}

● 位图与矢量图相比，位图（14）。

（14）A. 占用空间较大，处理侧重于获取和复制，显示速度快
　　B. 占用空间较小，处理侧重于绘制和创建，显示速度较慢
　　C. 占用空间较大，处理侧重于获取和复制，显示速度较慢
　　D. 占用空间较小，处理侧重于绘制和创建，显示速度快

● 在采用结构化方法进行系统分析时，根据分解与抽象的原则，按照系统中数据处理的流程，用（15）来建立系统的逻辑模型，从而完成分析工作。

（15）A. ER 图　　B. 数据流图
　　C. 程序流程图　　D. 软件体系结构

● 面向对象开发方法的基本思想是尽可能按照人类认识客观世界的方法来分析和解决问题，（16）方法不属于面向对象方法。

（16）A. Booch　　B. Coad　　C. OMT　　D. Jackson

● 确定构建软件系统所需要的人数时，无需考虑（17）。

（17）A. 系统的市场前景　　B. 系统的规模
　　C. 系统的技术复杂性　　D. 项目计划

● 一个项目为了修正一个错误而进行了变更。但这个错误被修正后，却引起以前可以正确运行的代码出错。（18）最可能发现这一问题。

（18）A. 单元测试　　B. 接受测试
　　C. 回归测试　　D. 安装测试

● 风险预测从两个方面评估风险，即风险发生的可能性以及（19）。

（19）A. 风险产生的原因　　B. 风险监控技术
　　C. 风险能否消除　　D. 风险发生所产生的后果

● 许多程序设计语言规定，程序中的数据都必须具有类型，其作用不包括（20）。

（20）A. 便于为数据合理分配存储单元
　　B. 便于对参与表达式计算的数据对象进行检查
　　C. 便于定义动态数据结构
　　D. 便于规定数据对象的取值范围及能够进行的运算

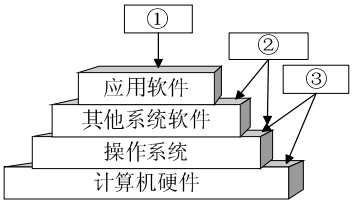
● 以下关于C/C++语言指针变量的叙述中，正确的是（21）。

（21）A. 指针变量可以是全局变量也可以是局部变量
　　B. 必须为指针变量与指针所指向的变量分配相同大小的存储空间
　　C. 对指针变量进行算术运算是没有意义的
　　D. 指针变量必须由动态产生的数据对象来赋值

● 将高级语言源程序翻译为机器语言程序的过程中常引入中间代码。以下关于中间代码的叙述中，错误的是（22）。

（22）A. 不同的高级程序语言可以产生同一种中间代码
　　B. 使用中间代码有利于进行与机器无关的优化处理
　　C. 使用中间代码有利于提高编译程序的可移植性
　　D. 中间代码与机器语言代码在指令结构上必须一致

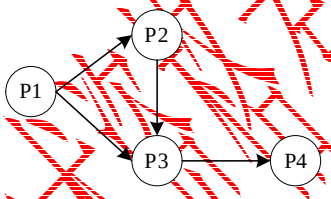
● 操作系统是裸机上的第一层软件，其他系统软件（如（23）等）和应用软件都是建立在操作系统基础上的。下图①②③分别表示（24）。



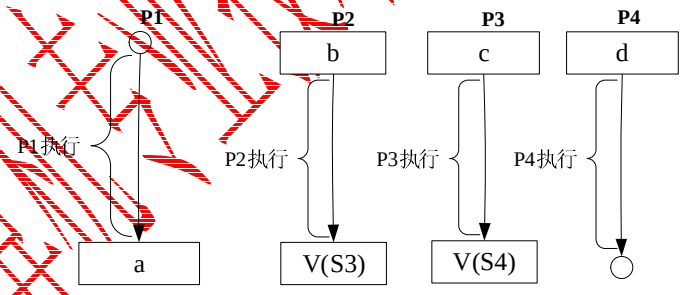
（23）A. 编译程序、财务软件和数据库管理系统软件
B. 汇编程序、编译程序和 Java 解释器
C. 编译程序、数据库管理系统软件和汽车防盗程序
D. 语言处理程序、办公管理软件和气象预报软件

（24）A. 应用软件开发者、最终用户和系统软件开发者
B. 应用软件开发者、系统软件开发者和最终用户
C. 最终用户、系统软件开发者和应用软件开发者
D. 最终用户、应用软件开发者和系统软件开发者

● 进程 P1、P2、P3 和 P4 的前趋图如下：



若用 PV 操作控制这几个进程并发执行的过程，则需要设置 4 个信号量 S1、S2、S3 和 S4，且信号量初值都等于零。下图中 a 和 b 应分别填写（25），c 和 d 应分别填写（26）。



（25）A. P（S1）P（S2）和 P（S3）　　B. P（S1）P（S2）和 V（S1）
C. V（S1）V（S2）和 P（S1）　　D. V（S1）V（S2）和 V（S3）

（26）A. P（S1）P（S2）和 P（S4）　　B. P（S2）P（S3）和 P（S4）
C. V（S1）V（S2）和 V（S4）　　D. V（S2）V（S3）和 V（S4）

● 若系统正在将（27）文件修改的结果写回磁盘时系统发生崩溃，则对系统的影响相对较大。

（27）A. 空闲块　　B. 目录　　C. 用户数据　　D. 用户程序

● UNIX 系统采用直接、一级、二级和三级间接索引技术访问文件，其索引结点有 13 个地址项（i_addr[0]～i_addr[12]）。如果每个盘块的大小为 1KB，每个盘块号占 4B，则进程 A 访问文件 F 中第 11264 字节处的数据时，（28）。

（28）A. 可直接寻址　　B. 需要一次间接寻址
　　C. 需要二次间接寻址　　D. 需要三次间接寻址

● 软件能力成熟度模型（CMM）的第 4 级（已管理级）的核心是（29）。

（29）A. 建立基本的项目管理和实践来跟踪项目费用、进度和功能特性
　　B. 组织具有标准软件过程
　　C. 对软件过程和产品都有定量的理解和控制
　　D. 先进的新思想和新技术促进过程不断改进

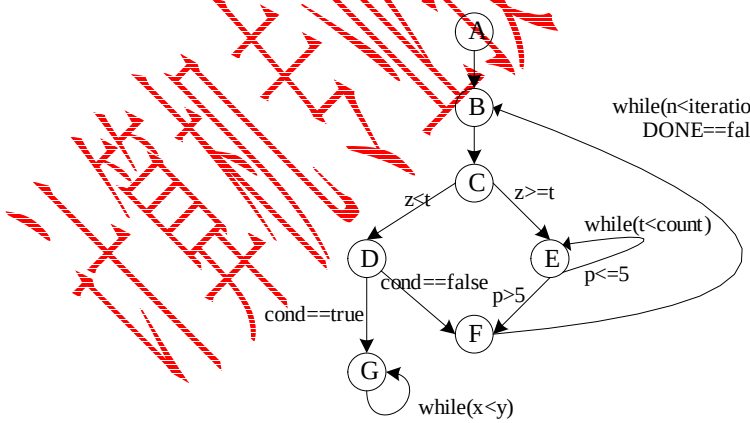
● 软件系统设计的主要目的是为系统制定蓝图，（30）并不是软件设计模型所关注的。

（30）A. 系统总体结构　　B. 数据结构　　C. 界面模型　　D. 项目范围

● ISO/IEC 9126 软件质量模型中，可靠性质量特性包括多个子特性。一软件在故障发生后，要求在 90 秒内恢复其性能和受影响的数据，与达到此目的有关的软件属性为（31）子特性。

（31）A. 容错性　　B. 成熟性　　C. 易恢复性　　D. 易操作性

● 某程序的程序图如下所示，运用 McCabe 度量法对其进行度量，其环路复杂度是（32）。



（32）A. 2　　B. 3　　C. 4　　D. 5

● 系统开发计划用于系统开发人员与项目管理人员在项目期内进行沟通，它包括（33）和预算分配表等。

（33）A. PERT 图　　B. 总体规划　　C. 测试计划　　D. 开发合同

● 改正在软件系统开发阶段已经发生而系统测试阶段还没有发现的错误，属于（34）维护。

（34）A. 正确性　　B. 适应性　　C. 完善性　　D. 预防性

● 某系统重用了第三方组件（但无法获得其源代码），则应采用（35）对组件进行测试。

（35）A. 基本路径覆盖　　B. 分支覆盖　　C. 环路覆盖　　D. 黑盒测试

● 极限编程（XP）由价值观、原则、实践和行为四个部分组成，其中价值观包括沟通、简单性、（36）。

（36）A. 好的计划　　B. 不断的发布　　C. 反馈和勇气　　D. 持续集成

● 以下关于类和对象的叙述中，错误的是（37）。

（37）A. 类是具有相同属性和服务的一组对象的集合

B. 类是一个对象模板，用它仅可以产生一个对象

C. 在客观世界中实际存在的是类的实例，即对象

D. 类为属于该类的全部对象提供了统一的抽象描述

● （38）是把对象的属性和服务结合成一个独立的系统单元，并尽可能隐藏对象的内部细节；（39）是指子类可以自动拥有父类的全部属性和服务；（40）是对象发出的服务请求，一般包含提供服务的对象标识、服务标识、输入信息和应答信息等。

（38）A. 继承　　B. 多态　　C. 消息　　D. 封装

（39）A. 继承　　B. 多态　　C. 消息　　D. 封装

（40）A. 继承　　B. 多态　　C. 消息　　D. 封装

● 以下关于面向对象分析的叙述中，错误的是（41）。

（41）A. 面向对象分析着重分析问题域和系统责任

B. 面向对象分析需要考虑系统的测试问题

C. 面向对象分析忽略与系统实现有关的问题

D. 面向对象分析建立独立于实现的系统分析模型

● 以下关于面向对象设计的叙述中，错误的是（42）。

（42）A. 高层模块不应该依赖于底层模块

B. 抽象不应该依赖于细节

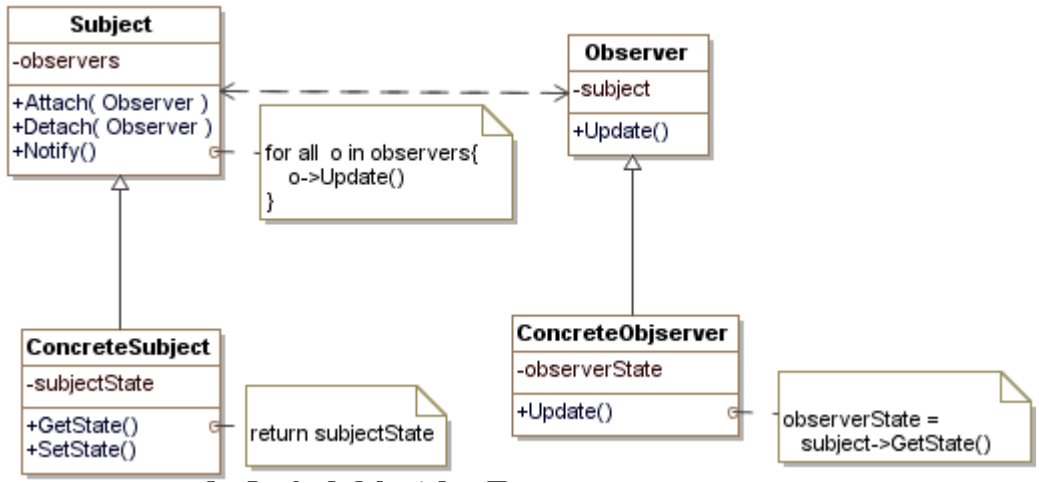
C. 细节可以依赖于抽象

D. 高层模块无法不依赖于底层模块

● 采用（43）设计模式可保证一个类仅有一个实例；采用（44）设计模式可将对象组合成树形结构以表示“部分-整体”的层次结构，使用户对单个对象和组合对象的使用具有一致性；采用（45）设计模式可动态地给一个对象添加一些额外的职责。

（43）A. 命令（Command）　　B. 单例（Singleton）
　　C. 装饰（Decorate）　　D. 组合（Composite）
（44）A. 命令（Command）　　B. 单例（Singleton）
　　C. 装饰（Decorate）　　D. 组合（Composite）
（45）A. 命令（Command）　　B. 单例（Singleton）
　　C. 装饰（Decorate）　　D. 组合（Composite）

● 下列 UML 类图表示的是（46）设计模式。该设计模式中，（47）。



（46）A. 备忘录（Memento）　　B. 策略（Strategy）
　　C. 状态（State）　　D. 观察者（Observer）
（47）A. 一个 Subject 对象可对应多个 Observer 对象
　　B. Subject 只能有一个 ConcreteSubject 子类
　　C. Observer 只能有一个 ConcreteObserver 子类
　　D. 一个 Subject 对象必须至少对应一个 Observer 对象

● 以下关于编译系统对某高级语言进行翻译的叙述中，错误的是（48）。

（48）A. 词法分析将把源程序看作一个线性字符序列进行分析
　　B. 语法分析阶段可以发现程序中所有的语法错误
　　C. 语义分析阶段可以发现程序中所有的语义错误
　　D. 目标代码生成阶段的工作与目标机器的体系结构相关

● 若一个程序语言可以提供链表的定义和运算，则其运行时的（49）。

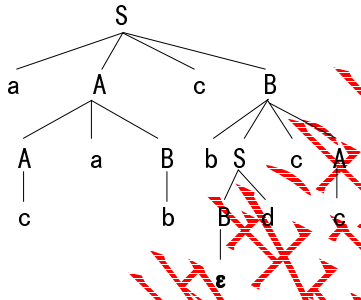
（49）A. 数据空间适合采用静态存储分配策略

B. 数据空间必须采用堆存储分配策略

C. 指令空间需要采用栈结构

D. 指令代码必须放入堆区

● 由某上下文无关文法 M[S]推导出某句子的分析树如下图所示，则错误的叙述是（50）。



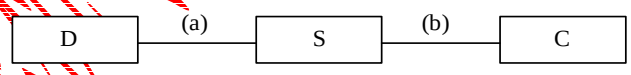
（50）A. 该文法推导出的句子必须以“a”开头

B. acabcbdcc 是该文法推导出的一个句子

C. “S->aAcB”是该文法的一个产生式

D. a、b、c、d 属于该文法的终结符号集

● 假设有学生 S（学号，姓名，性别，入学时间，联系方式），院系 D（院系号，院系名称，电话号码，负责人）和课程 C（课程号，课程名）三个实体，若一名学生属于一个院系，一个院系有多名学生；一名学生可以选择多门课程，一门课程可被多名学生选择，则图中（a）和（b）分别为（51）联系。假设一对多联系不转换为一个独立的关系模式，那么生成的关系模式（52）。



（51）A. 1 * 和 1 *　　B. 1 * 和 * 1

C. 1 * 和 * *　　D. * 1 和 * *

（52）A. S 中应加入关系模式 D 的主键

B. S 中应加入关系模式 C 的主键

C. D 中应加入关系模式 S 的主键

D. C 中应加入关系模式 S 的主键

● 软硬件故障常造成数据库中的数据破坏。数据库恢复就是（53）。

（53）A. 重新安装数据库管理系统和应用程序
B. 重新安装应用程序，并将数据库做镜像
C. 重新安装数据库管理系统，并将数据库做镜像
D. 在尽可能短的时间内，把数据库恢复到故障发生前的状态

● 设有员工实体 Emp（员工号，姓名，性别，年龄，出生年月，联系方式，部门号），其中“联系方式”要求记录该员工的手机号码和办公室电话，部门号要求参照另一部门实体 Dept 的主码“部门号”。Emp 实体中存在派生属性和多值属性：（54），对属性部门号应该进行（55）约束；可以通过命令（56）修改表中的数据。

（54）A. 年龄和出生年月　　B. 年龄和联系方式
C. 出生年月和联系方式　　D. 出生年月和年龄
（55）A. 非空主键　　B. 主键
C. 外键　　D. 候选键
（56）A. INSERT　　B. DELETE　　C. UPDATE　　D. MODIFY

● 已知一个二叉树的先序遍历序列为①、②、③、④、⑤，中序遍历序列为②、①、④、③、⑤，则该二叉树的后序遍历序列为（57）。对于任意一棵二叉树，叙述错误的是（58）。

（57）A. ②、③、①、⑤、④
B. ①、②、③、④、⑤
C. ②、④、⑤、③、①
D. ④、⑤、③、②、①
（58）A. 由其后序遍历序列和中序遍历序列可以构造该二叉树的先序遍历序列
B. 由其先序遍历序列和后序遍历序列可以构造该二叉树的中序遍历序列
C. 由其层序遍历序列和中序遍历序列可以构造该二叉树的先序遍历序列
D. 由其层序遍历序列和中序遍历序列不能构造该二叉树的后序遍历序列

● 邻接矩阵和邻接表是图（网）的两种基本存储结构，对于具有 n 个顶点、e 条边的图，（59）。

（59）A. 进行深度优先遍历运算所消耗的时间与采用哪一种存储结构无关
B. 进行广度优先遍历运算所消耗的时间与采用哪一种存储结构无关
C. 采用邻接表表示图时，查找所有顶点的邻接顶点的时间复杂度为 O(n*e)
D. 采用邻接矩阵表示图时，查找所有顶点的邻接顶点的时间复杂度为 $O(n^2)$

● 单向链表中往往含有一个头结点，该结点不存储数据元素，一般令链表的头指针指向该结点，而该结点指针域的值为第一个元素结点的指针。以下关于单链表头结点的叙述中，错误的是（60）。

（60）A. 若在头结点中存入链表长度值，则求链表长度运算的时间复杂度为 O(1)
B. 在链表的任何一个元素前后进行插入和删除操作可用一致的方式进行处理
C. 加入头结点后，代表链表的头指针不因为链表为空而改变
D. 加入头结点后，在链表中进行查找运算的时间复杂度为 O(1)

● 对于长度为 m（m>1）的指定序列，通过初始为空的一个栈、一个队列后，错误的叙述是（61）。

（61）A. 若入栈和入队的序列相同，则出栈序列和出队序列可能相同
B. 若入栈和入队的序列相同，则出栈序列和出队序列可以互为逆序
C. 入队序列与出队序列关系为 1:1，而入栈序列与出栈序列关系是 1:n(n≥1)
D. 入栈序列与出栈序列关系为 1:1，而入队序列与出队序列关系是 1:n(n≥1)

● 字符串采用链表存储方式时，每个结点存储多个字符有助于提高存储密度。若采用结点大小相同的链表存储串，则串比较、求子串、串连接、串替换等串的基本运算中，（62）。

（62）A. 进行串的比较运算最不方便　　B. 进行求子串运算最不方便
C. 进行串连接最不方便　　D. 进行串替换最不方便

● 某算法的时间复杂度表达式为 $T(n)=an^2+bn\lg n+cn+d$，其中，n 为问题的规模，a、b、c 和 d 为常数，用 O 表示其渐近时间复杂度为（63）。

（63）A. $O(n^2)$　　B. $O(n)$　　C. $O(n\lg n)$　　D. $O(1)$

● 以下关于快速排序算法的描述中，错误的是（64）。在快速排序过程中，需要设立基准元素并划分序列来进行排序。若序列由元素{12,25,30,45,52,67,85}构成，则初始排列为（65）时，排序效率最高（令序列的第一个元素为基准元素）。

（64）A. 快速排序算法是不稳定的排序算法
B. 快速排序算法在最坏情况下的时间复杂度为 O(nlgn)
C. 快速排序算法是一种分治算法
D. 当输入数据基本有序时，快速排序算法具有最坏情况下的时间复杂度

（65）A. 45,12,30,25,67,52,85　　B. 85,67,52,45,30,25,12
C. 12,25,30,45,52,67,85　　D. 45,12,25,30,85,67,52

● 下列网络互连设备中，属于物理层的是（66），属于网络层的是（67）。

（66）A. 中继器　　B. 交换机　　C. 路由器　　D. 网桥

（67）A. 中继器　　B. 交换机　　C. 路由器　　D. 网桥

● 下图是 HTML 文件 test.html 在 IE 中的显示效果，实现图中①处效果的 HTML 语句是（68），实现图中②处效果的 HTML 语句是（69），实现图中③处效果的 HTML 语句是（70）。

（68）A．<TITLE>我的主页</TITLE>　　B．<HEAD>我的主页</HEAD >
　　C．<BODY>我的主页</BODY >　　D．<H1>我的主页</H1>
（69）A．<HR>　　B．<LINE> </LINE >
　　C．<CELL> </CELL >　　D．<TR> </TR>
（70）A．<B>Welcome</B>　　B．<UL>Welcome</UL>
　　C．<I>Welcome</I>　　D．<H>Welcome</H>

● Why is （71） fun? What delights may its practitioner expect as his reward? First is the sheer joy of making things. As the child delights in his mud pie, so the adult enjoys building things, especially things of his own design. Second is the pleasure of making things that are useful to other people. Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles, playing out the consequences of principles built in from the beginning. Fourth is the joy of always learning, which springs from the （72） nature of the task. In one way or another the problem is ever new, and its solver learns something: sometimes （73）, sometimes theoretical, and sometimes both. Finally, there is the delight of working in such a tractable medium. The （74）, like the poet, works only slightly removed from pure thought-stuff. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures.

Yet the program （75）, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. It prints results, draws pictures, produces sounds, moves arms. Programming then is fun because it gratifies creative longings built deep within us and delights sensibilities we have in common with all men.

（71）A. programming　　B. composing
　　C. working　　D. writing

（72）A. repeating　　B. basic
　　C. non-repeating　　D. advance
（73）A. semantic　　B. practical
　　C. lexical　　D. syntactical
（74）A. poet　　B. architect
　　C. doctor　　D. programmer
（75）A. construct　　B. code
　　C. size　　D. scale

全国计算机技术与软件专业技术资格（水平）考试

2009 年下半年　软件设计师　下午试卷

（考试时间　14:00～16:30　共 150 分钟）

请按下述要求正确填写答题纸

1. 在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。
2. 在答题纸的指定位置填写准考证号、出生年月日和姓名。
3. 答题纸上除填写上述内容外只能写解答。
4. 本试卷共 7 道题，试题一至试题四是必答题，试题五至试题七选答 1 道。每题 15 分，满分 75 分。
5. 解答时字迹务必清楚，字迹不清时，将不评分。
6. 仿照下面例题，将解答写在答题纸的对应栏内。

例题

2009 年下半年全国计算机技术与软件专业技术资格（水平）考试日期是（1）月（2）日。

因为正确的解答是“11 月 14 日”，故在答题纸的对应栏内写上“11”和“14”（参看下表）。

例题	解答栏
（1）	11
（2）	14

试题一（共 15 分）

阅读以下说明和数据流图，回答问题1至问题4，将解答填入答题纸的对应栏内。

【说明】

现准备为某银行开发一个信用卡管理系统 CCMS，该系统的基本功能为：

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

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

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

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

图 1-1 和图 1-2 分别给出了该系统的顶层数据流图和 0 层数据流图的初稿。

【问题 1】（3 分）

根据【说明】，将图 1-1 中的 E1~E3 填充完整。

【问题 2】（3 分）

图 1-1 中缺少三条数据流，根据【说明】，分别指出这三条数据流的起点和终点。（注：数据流的起点和终点均采用图中的符号和描述）

【问题 3】（5 分）

图 1-2 中有两条数据流是错误的，请指出这两条数据流的名称，并改正。（注：数据流的起点和终点均采用图中的符号和描述）

【问题 4】（4 分）

根据【说明】，将图 1-2 中 P1~P4 的处理名称填充完整。

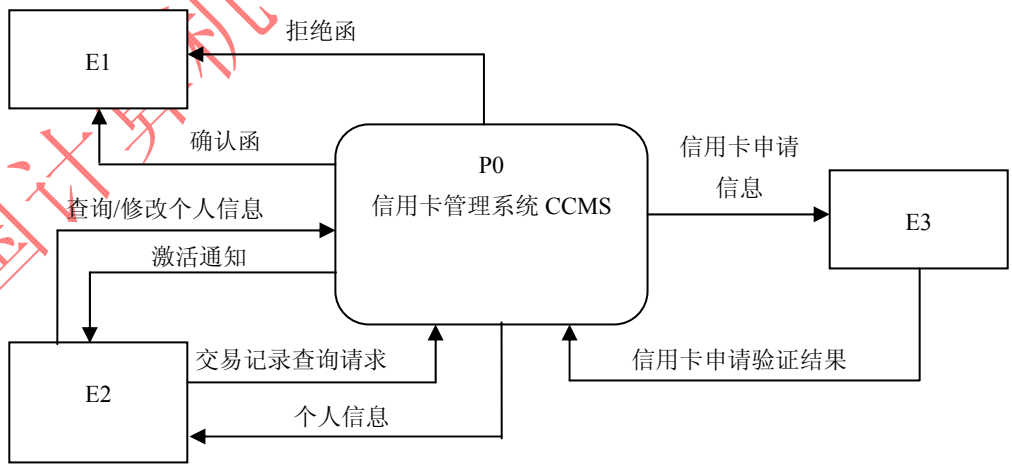


图 1-1 顶层数据流图

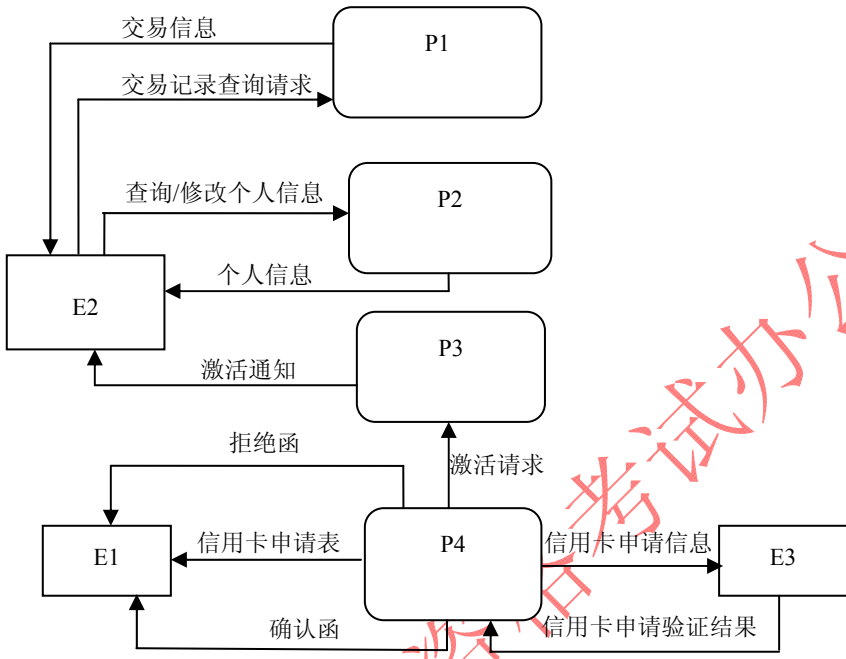


图 1-2　0 层数据流图

试题二（共 15 分）

阅读下列说明，回答问题1至问题3，将解答填入答题纸的对应栏内。

【说明】

某公司拟开发一多用户电子邮件客户端系统，部分功能的初步需求分析结果如下：

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

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

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

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

【问题 1】（5 分）

根据以上说明设计的 E-R 图如图 2-1 所示，请指出地址簿与用户、电子邮件帐号与邮件、邮件与附件之间的联系类型。

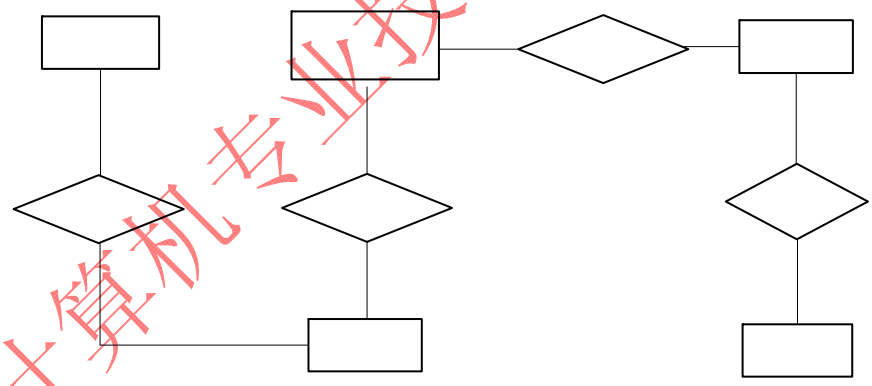


图 2-1 电子邮件客户端系统 E-R 图

【问题 2】（4 分）

该邮件客户端系统的主要关系模式如下，请填补(a)～(c)的空缺部分。

用户(用户名，用户密码)

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

邮件帐号（邮件地址，邮件密码，用户名）

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

附件(　　(c)　　，附件号，附件文件名，附件大小)

【问题 3】(6 分)

（1）请指出**【问题 2】**中给出的地址簿、邮件和附件关系模式的主键，如果关系模式存在外键请指出。

（2）附件属于弱实体吗？请用 50 字以内的文字说明原因。

试题三（共 15 分）

阅读下列说明和 UML 图，回答问题 1 至问题 4，将解答填入答题纸的对应栏内。

【说明】

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

企业的任何员工都可以查看菜单和今日特价。

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

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

菜单管理员是餐厅特定员工，可以管理菜单。

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

顾客订餐过程如下：

1. 顾客请求查看菜单；
2. 系统显示菜单和今日特价；
3. 顾客选菜；
4. 系统显示订单和价格；
5. 顾客确认订单；
6. 系统显示可送餐时间；
7. 顾客指定送餐时间、地点和支付方式；
8. 系统确认接受订单，然后发送 Email 给顾客以确认订餐，同时发送相关订餐信息通知给餐厅员工。

系统采用面向对象方法开发，使用 UML 进行建模。系统的顶层用例图和一次订餐的活动图初稿分别如图 3-1 和图 3-2 所示。

【问题 1】（2 分）

根据**【说明】**中的描述，给出图 3-1 中 A1 和 A2 所对应的参与者。

【问题 2】（8 分）

根据**【说明】**中的描述，给出图 3-1 中缺少的四个用例及其所对应的参与者。

【问题 3】（4 分）

根据**【说明】**中的描述，给出图 3-2 中（1）～（4）处对应的活动名称或图形符号。

【问题 4】（1 分）

指出图 3-1 中员工和顾客之间是什么关系，并解释该关系的内涵。

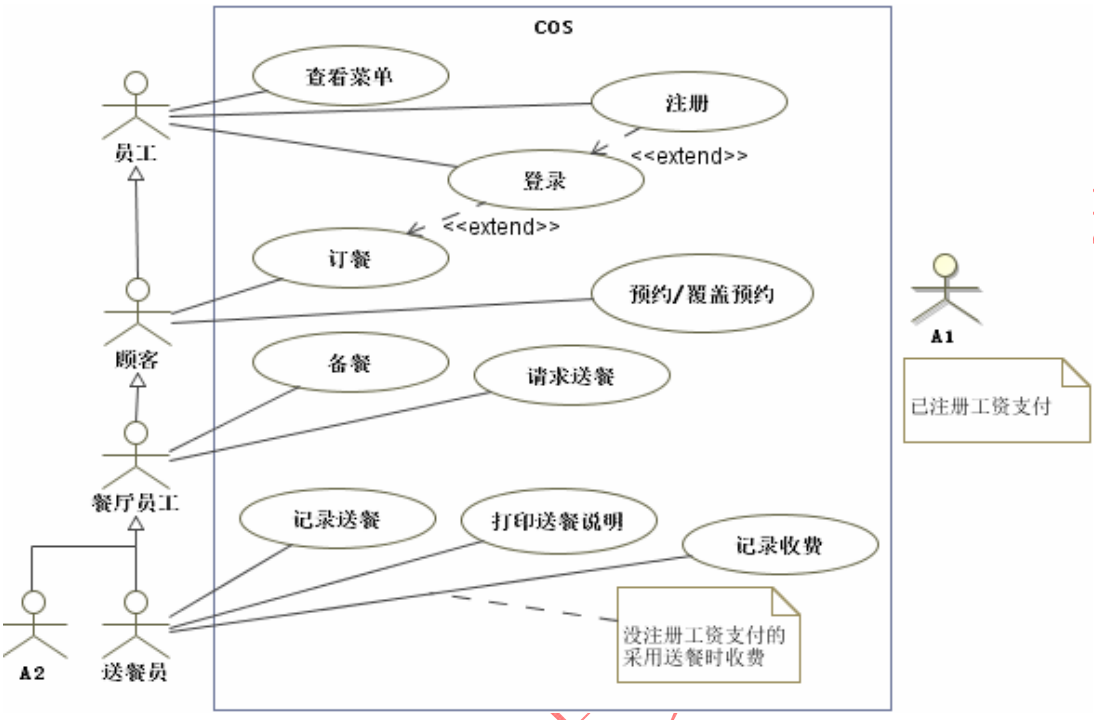


图 3-1　COS 系统顶层用例图

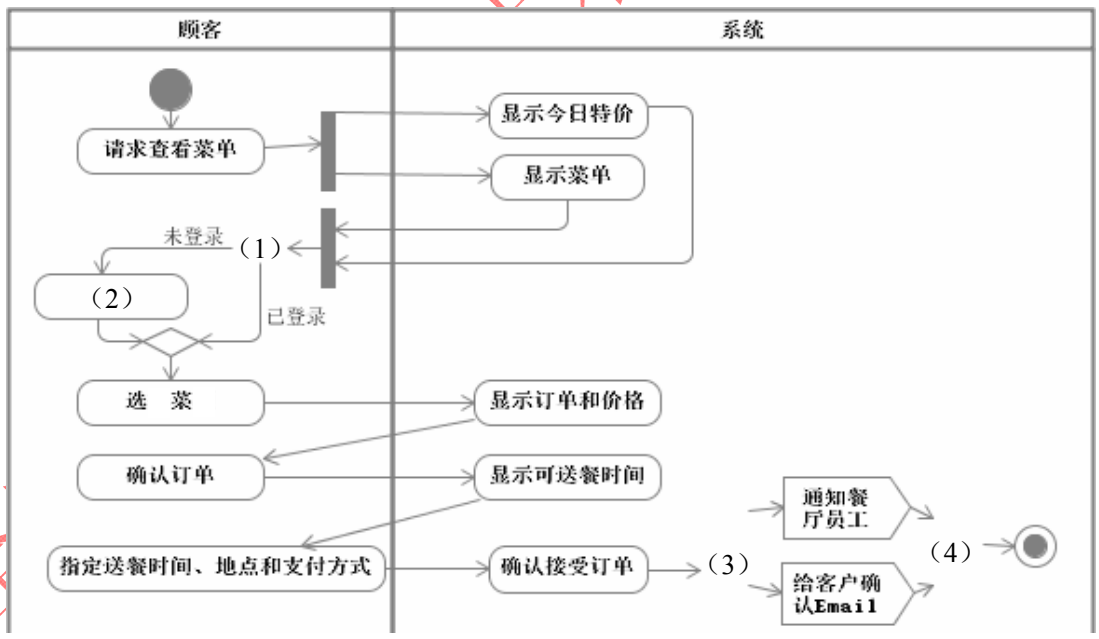


图 3-2　一次订餐的活动图

试题四（共 15 分）

阅读下列说明，回答问题 1 至问题 2，将解答填入答题纸的对应栏内。

【说明】

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

【问题 1】（8 分）

用回溯法求解此 0-1 背包问题，请填充下面伪代码中（1）～（4）处空缺。

回溯法是一种系统的搜索方法。在确定解空间后，回溯法从根结点开始，按照深度优先策略遍历解空间树，搜索满足约束条件的解。对每一个当前结点，若扩展该结点已经不满足约束条件，则不再继续扩展。为了进一步提高算法的搜索效率，往往需要设计一个限界函数，判断并剪枝那些即使扩展了也不能得到最优解的结点。现在假设已经设计了 BOUND(v,w,k,W)函数，其中 v、w、k 和 W 分别表示当前已经获得的价值、当前背包的重量、已经确定是否选择的物品数和背包的总容量。对应于搜索树中的某个结点，该函数值表示确定了部分物品是否选择之后，对剩下的物品在满足约束条件的前提下进行选择可能获得的最大价值，若该价值小于等于当前已经得到的最优解，则该结点无需再扩展。

下面给出 0-1 背包问题的回溯算法伪代码。

函数参数说明如下：

W：背包容量；n：物品个数；w：重量数组；v：价值数组；fw：获得最大价值时背包的重量；fp：背包获得的最大价值；X：问题的最优解。

变量说明如下：

cw：当前的背包重量；cp：当前获得的价值；k：当前考虑的物品编号；Y：当前已获得的部分解。

```
BKNAP( W,n,w,v,fw,fp,X )
1 cw ← cp ← 0
2 ____（1）____
3 fp ← -1
4 while true
5       while k ≤ n and cw + w[k] ≤ W do
6             ____（2）____
7             cp ← cp + v[k]
8             Y[k] ← 1
9             k ← k + 1
10      if k > n then
11            if fp < cp then
```

```
12                      fp ← cp
13                      fw ← cw
14                      k ← n
15                      X ← Y
16          else Y(k) ← 0
17          while BOUND(cp,cw,k,W) ≤ fp do
18              while k ≠ 0 and Y(k) ≠ 1 do
19                  （3）
20              if k = 0 then return
21              Y[k] ← 0
22              cw ← cw - w[k]
23              cp ← cp - v[k]
24          （4）
```

【问题 2】（7 分）

考虑表 4-1 的实例，假设有 3 个物品，背包容量为 22。图 4-1 中是根据上述算法构造的搜索树，其中结点的编号表示了搜索树生成的顺序，边上的数字 1/0 分别表示选择/不选择对应物品。除了根结点之外，每个左孩子结点旁边的上下两个数字分别表示当前背包的重量和已获得的价值，右孩子结点旁边的数字表示扩展了该结点后最多可能获得的价值。为获得最优解，应该选择物品（5），获得的价值为（6）。

表 4-1　0-1 背包问题实例

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

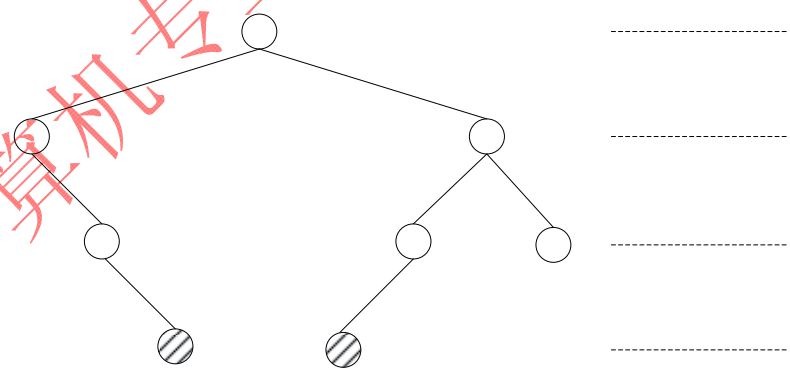


图 4-1　表 4-1 实例的搜索树

对于表 4-1 的实例，若采用穷举法搜索整个解空间，则搜索树的结点数为（7），而用了上述回溯法，搜索树的结点数为（8）。

从下列的 3 道试题（试题五至试题七）中任选 1 道解答。
如果解答的试题数超过 1 道，则题号小的 1 道解答有效。

试题五（共 15 分）

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

【说明】

现欲构造一文件/目录树，采用组合（Composite）设计模式来设计，得到的类图如 5-1 所示：

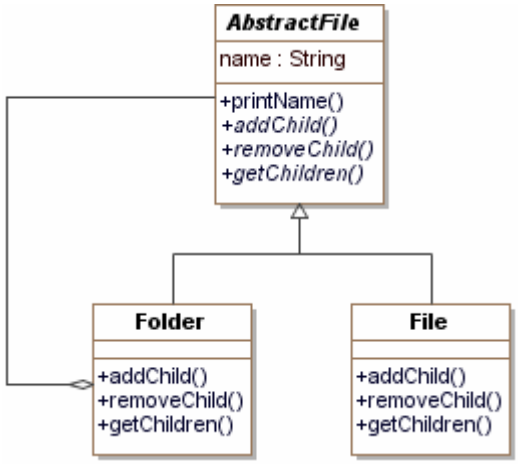


图 5-1 类图

【C++代码】

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

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

```
class File : public AbstractFile {
public :
    File(string name) {__（1）__= name; }
    void addChild(AbstractFile    *file) { return ; }
    void removeChild(AbstractFile *file) { return ; }
    __（2）__ getChildren() { return __（3）__; }
};

class Folder :public AbstractFile {
private :
    list <AbstractFile*> childList;    // 存储子目录或文件
public :
    Folder(string name) {__（4）__ = name; }
    void addChild(AbstractFile *file) { childList.push_back(file); }
    void removeChild(AbstractFile *file) { childList.remove(file);}
    list<AbstractFile*> *getChildren() { return __（5）__;}
};



void main() {
    // 构造一个树形的文件/目录结构
    AbstractFile *rootFolder = new Folder("c:\\");
    AbstractFile *compositeFolder = new Folder("composite");
    AbstractFile *windowsFolder = new Folder("windows");
    AbstractFile *file = new File("TestComposite.java");
    rootFolder->addChild(compositeFolder);
    rootFolder->addChild(windowsFolder);
    compositeFolder->addChild(file);
}
```

试题六（共 15 分）

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

【说明】

现欲构造一文件/目录树，采用组合（Composite）设计模式来设计，得到的类图如 6-1 所示：

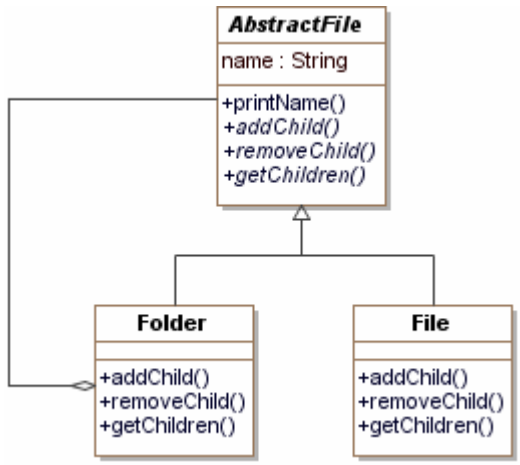


图 6-1 类图

【Java 代码】

```
import java.util.ArrayList;
import java.util.List;

__（1）__ class AbstractFile {
    protected String name;
    public void printName(){System.out.println(name);}
    public abstract boolean addChild(AbstractFile file);
    public abstract boolean removeChild(AbstractFile file);
    public abstract List<AbstractFile> getChildren();
}

class File extends AbstractFile {
    public File(String name) { this.name = name; }
    public boolean addChild(AbstractFile file) { return false; }
    public boolean removeChild(AbstractFile file) { return false; }
    public List<AbstractFile> getChildren() { return __（2）__; }
}

class Folder extends AbstractFile {
```

```
    private List <AbstractFile> childList;
    public Folder(String name) {
        this.name = name;
        this.childList = new ArrayList<AbstractFile>();
    }
    public boolean addChild(AbstractFile file) { return childList.add(file); }
    public boolean removeChild(AbstractFile file) { return childList.remove(file); }
    public ___（3）___<AbstractFile> getChildren() { return ___（4）___; }
}

public class Client {
    public static void main(String[] args) {
        // 构造一个树形的文件/目录结构
        AbstractFile rootFolder = new Folder("c:\\");
        AbstractFile compositeFolder = new Folder("composite");
        AbstractFile windowsFolder = new Folder("windows");
        AbstractFile file = new File("TestComposite.java");
        rootFolder.addChild(compositeFolder);
        rootFolder.addChild(windowsFolder);
        compositeFolder.addChild(file);

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

该程序运行后输出结果为:

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

试题七（共 15 分）

阅读以下说明和C程序，将应填入 __（n）__ 处的字句写在答题纸的对应栏内。

【说明】

现有 n（n < 1000）节火车车厢，顺序编号为 1,2,3,...,n，按编号连续依次从 A 方向的铁轨驶入，从 B 方向铁轨驶出，一旦车厢进入车站（Station）就不能再回到 A 方向的铁轨上；一旦车厢驶入 B 方向铁轨就不能再回到车站，如图 7-1 所示，其中 Station 为栈结构，初始为空且最多能停放 1000 节车厢。

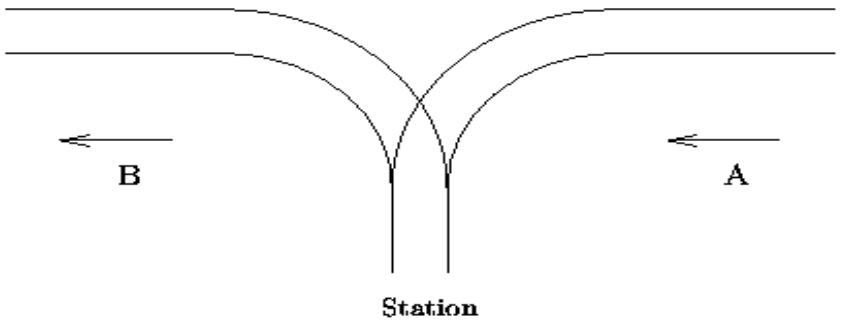


图 7-1　车站示意图

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

1, 3, 2, 5

void InitStack(STACK *s)：初始化栈。

void Push(STACK *s, int e)：将一个整数压栈，栈中元素数目增 1。

void Pop(STACK *s)：栈顶元素出栈，栈中元素数目减 1。

int Top(STACK s)：返回非空栈的栈顶元素值，栈中元素数目不变。

int IsEmpty(STACK s)：若是空栈则返回 1，否则返回 0。

【C 程序】

```
#include<stdio.h>

/*此处为栈类型及其基本操作的定义，省略*/

int main( ){
    STACK station;
    int state[1000];
    int n;                          /*车厢数*/
    int begin, i, j, maxNo;         /*maxNo 为 A 端正待入栈的车厢编号*/
    printf("请输入车厢数：  ");
    scanf("%d",&n);
```

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