全国计算机技术与软件专业技术资格（水平）考试

2007 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

● 2007 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. 6　　B. 5　　C. 4　　D. 3

因为考试日期是“11 月 3 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）。

● 在指令系统的各种寻址方式中，获取操作数最快的方式是（1）。若操作数的地址包含在指令中，则属于（2）方式。

（1）A. 直接寻址　　B. 立即寻址　　C. 寄存器寻址　　D. 间接寻址

（2）A. 直接寻址　　B. 立即寻址　　C. 寄存器寻址　　D. 间接寻址

● 系统响应时间和作业吞吐量是衡量计算机系统性能的重要指标。对于一个持续处理业务的系统而言，（3），表明其性能越好。

（3）A. 响应时间越短，作业吞吐量越小　　B. 响应时间越短，作业吞吐量越大

C. 响应时间越长，作业吞吐量越大　　D. 响应时间不会影响作业吞吐量

● 若每一条指令都可以分解为取指、分析和执行三步。已知取指时间 $t_{取指}$=4△t，分析时间 $t_{分析}$=3△t，执行时间 $t_{执行}$=5△t。如果按串行方式执行完 100 条指令需要（4）△t。如果按照流水方式执行，执行完 100 条指令需要（5）△t。

（4）A. 1190　　B.1195　　C. 1200　　D.1205

（5）A. 504　　B. 507　　C. 508　　D. 510

● 若内存地址区间为 4000H～43FFH，每个存贮单元可存储 16 位二进制数，该内存区域用 4 片存储器芯片构成，则构成该内存所用的存储器芯片的容量是（6）。

（6）A. 512×16bit　　B. 256×8bit　　C. 256×16bit　　D. 1024×8bit

● 某 Web 网站向 CA 申请了数字证书。用户登录该网站时，通过验证（7），可确认该数字证书的有效性，从而（8）。

（7）A. CA 的签名　　B. 网站的签名　　C. 会话密钥　　D. DES 密码

（8）A. 向网站确认自己的身份　　B. 获取访问网站的权限

C. 和网站进行双向认证　　D. 验证该网站的真伪

● 实现 VPN 的关键技术主要有隧道技术、加解密技术、（9）和身份认证技术。

（9）A. 入侵检测技术　　B. 病毒防治技术

C. 安全审计技术　　D. 密钥管理技术

● 若某人持有盗版软件，但他本人确实不知道该软件是盗版的，则（10）承担侵权责任。

（10）A. 应由该软件的持有者　　B. 应由该软件的提供者

C. 应由该软件的提供者和持有者共同　　D. 该软件的提供者和持有者都不

● （11）不属于知识产权的范围。

（11）A. 地理标志权　　B. 物权　　C. 邻接权　　D. 商业秘密权

● W3C 制定了同步多媒体集成语言规范，称为（12）规范。

（12）A. XML　　B. SMIL　　C. VRML　　D. SGML

● 对同一段音乐可以选用 MIDI 格式或 WAV 格式来记录存储。以下叙述中（13）是不正确的。

（13）A. WAV 格式的音乐数据量比 MIDI 格式的音乐数据量大

B. 记录演唱会实况不能采用 MIDI 格式的音乐数据

C. WAV 格式的音乐数据没有体现音乐的曲谱信息

D. WAV 格式的音乐数据和 MIDI 格式的音乐数据都能记录音乐波形信息

● 设计制作一个多媒体地图导航系统，使其能根据用户需求缩放地图并自动搜索路径，最适合的地图数据应该是（14）。

（14）A. 真彩色图像　　B. 航拍图像　　C. 矢量化图形　　D. 高清晰灰度图像

● 给定 C 语言的数据结构

```
struct  T {
    int  w;
    union T { char  c;   int  i;  double d; } U;
};
```

假设 char 类型变量的存储区大小是 1 字节，int 类型变量的存储区大小是 4 字节，double 类型变量的存储区大小是 8 字节，则在不考虑字对齐方式的情况下，为存储一个 struct T 类型变量所需要的存储区域至少应为（15）字节。

（15）A. 4　　B. 8　　C. 12　　D. 17

● 在过程式程序设计（①）、数据抽象程序设计（②）、面向对象程序设计（③）、泛型（通用）程序设计（④）中，C++ 语言支持（16），C 语言支持（17）。

（16）A. ①　　B. ②③　　C. ③④　　D. ①②③④

（17）A. ①　　B. ①③　　C. ②③　　D. ①②③④

● C 语言是一种（18）语言。

（18）A. 编译型　　B. 解释型　　C. 编译、解释混合型　　D. 脚本

● 采用 UML 进行软件建模过程中，类图是系统的一种静态视图，用（19）可明确表示两类事物之间存在的整体/部分形式的关联关系。

（19）A. 依赖关系　　B. 聚合关系　　C. 泛化关系　　D. 实现关系

● 若程序运行时系统报告除数为 0，这属于（20）错误。

（20）A. 语法　　B. 语用　　C. 语义　　D. 语境

● 集合 $L=\{a^m b^m \mid m \geq 0\}$ （21）。

（21）A. 可用正规式“$a^* b^*$”表示

B. 不能用正规式表示，但可用非确定的有限自动机识别

C. 可用正规式“$a^m b^m$”表示

D. 不能用正规式表示，但可用上下文无关文法表示

● 表达式“X = A + B×(C – D)/E”的后缀表示形式可以为（22）（运算符优先级相同时，遵循左结合的原则）。

（22）A. XAB + CDE/–×=　　B. XA+BC–DE/×=

C. XABCD–×E/+=　　D. XABCDE+×–/=

● 设备驱动程序是直接与（23）打交道的软件模块。一般而言，设备驱动程序的任务是接受来自与设备（24）。

（23）A. 硬件　　B. 办公软件　　C. 编译程序　　D. 连接程序

（24）A. 有关的上层软件的抽象请求，进行与设备相关的处理

B. 无关的上层软件的抽象请求，进行与设备相关的处理

C. 有关的上层软件的抽象请求，进行与设备无关的处理

D. 无关的上层软件的抽象请求，进行与设备无关的处理

● 某系统中有四种互斥资源 R1、R2、R3 和 R4，可用资源数分别为 3、5、6 和 8。假设在 T_0 时刻有 P1、P2、P3 和 P4 四个进程，并且这些进程对资源的最大需求量和已分配资源数如下表所示，那么在 T_0 时刻系统中 R1、R2、R3 和 R4 的剩余资源数分别为（25）。如果从 T_0 时刻开始进程按（26）顺序逐个调度执行，那么系统状态是安全的。

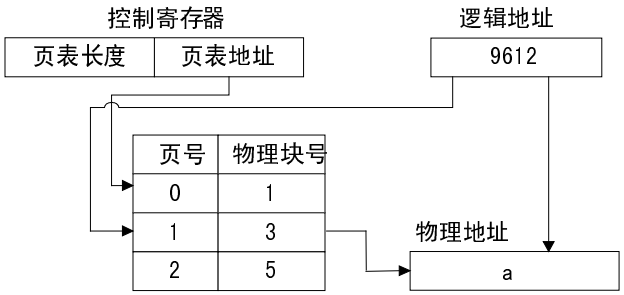
资源 / 进程	最大需求量				已分配资源数			
	R1	R2	R3	R4	R1	R2	R3	R4
P1	1	2	3	6	1	1	2	4
P2	1	1	2	2	0	1	2	2
P3	1	2	1	1	1	1	1	0
P4	1	1	2	3	1	1	1	1

（25）A. 3、5、6 和 8　　B. 3、4、2 和 2　　C. 0、1、2 和 1　　D. 0、1、0 和 1

（26）A. P1→P2→P4→P3　　B. P2→P1→P4→P3

C. P3→P2→P1→P4　　D. P4→P2→P3→P1

● 页式存储系统的逻辑地址是由页号和页内地址两部分组成，地址变换过程如下图所示。假定页面的大小为 8K，图中所示的十进制逻辑地址 9612 经过地址变换后，形成的物理地址 a 应为十进制（27）。



（27）A. 42380　　B. 25996　　C. 9612　　D. 8192

● 若文件系统容许不同用户的文件可以具有相同的文件名，则操作系统应采用（28）来实现。

（28）A. 索引表　　B. 索引文件　　C. 指针　　D. 多级目录

● 在软件开发中，（29）不能用来描述项目开发的进度安排。在其他三种图中，可用（30）动态地反映项目开发进展情况。

（29）A. 甘特图　　B. PERT 图　　C. PERT/CPM 图　　D. 鱼骨图

（30）A. 甘特图　　B. PERT 图　　C. PERT/CPM 图　　D. 鱼骨图

● 选择软件开发工具时，应考虑功能、（31）、稳健性、硬件要求和性能、服务和支持。

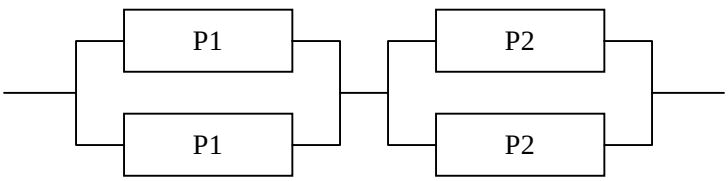
（31）A. 易用性　　B. 易维护性　　C. 可移植性　　D. 可扩充性

● 内聚性和耦合性是度量软件模块独立性的重要准则，软件设计时应力求（32）。

（32）A. 高内聚，高耦合　　B. 高内聚，低耦合

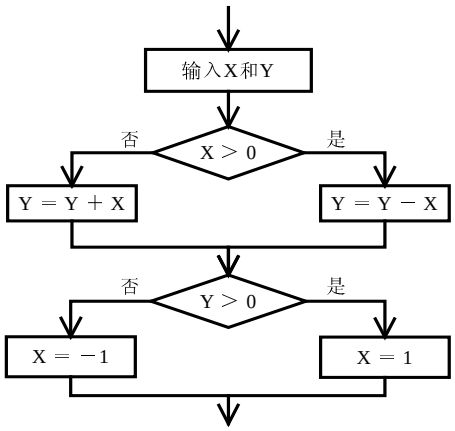
C. 低内聚，高耦合　　D. 低内聚，低耦合

● 某大型软件系统按功能可划分为 2 段 P1 和 P2。为提高系统可靠性，软件应用单位设计了如下图给出的软件冗余容错结构，其中 P1 和 P2 均有一个与其完全相同的冗余备份。若 P1 的可靠度为 0.9，P2 的可靠度为 0.9，则整个系统的可靠度是（33）。



（33）A. 0.6561　　B. 0.81　　C. 0.9801　　D. 0.9

● 对于如下的程序流程，当采用语句覆盖法设计测试案例时，至少需要设计（34）个测试案例。



（34）A. 1　　B. 2　　C. 3　　D. 4

● 为验证程序模块 A 是否正确实现了规定的功能，需要进行（35）；为验证模块 A 能否与其他模块按照规定方式正确工作，需要进行（36）。

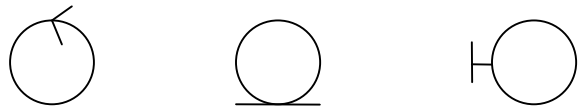
（35）A. 单元测试　　B. 集成测试　　C. 确认测试　　D. 系统测试

（36）A. 单元测试　　B. 集成测试　　C. 确认测试　　D. 系统测试

● （37）表示了系统与参与者之间的接口。在每一个用例中，该对象从参与者处收集信息，并将之转换为一种被实体对象和控制对象使用的形式。

（37）A. 边界对象　　B. 可视化对象　　C. 抽象对象　　D. 实体对象

● 在 UML 语言中，下图中的 a、b、c 三种图形符号按照顺序分别表示（38）。

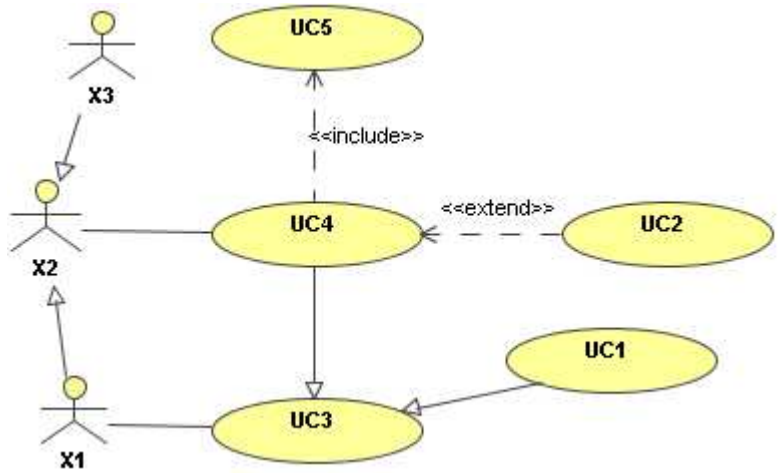


a　　b　　c

（38）A. 边界对象、实体对象、控制对象　　B. 实体对象 、边界对象、控制对象

C. 控制对象、实体对象、边界对象　　D. 边界对象 、控制对象、实体对象

● 在下面的用例图（UseCase Diagram）中，X1、X2 和 X3 表示（39），已知 UC3 是抽象用例，那么 X1 可通过（40）用例与系统进行交互。并且，用例（41）是 UC4 的可选部分，用例（42）是 UC4 的必须部分。



（39）A. 人　　B. 系统　　C. 参与者　　D. 外部软件

（40）A. UC4、UC1　　B. UC5、UC1　　C. UC5、UC2　　D. UC1、UC2

（41）A. UC1　　B. UC2　　C. UC3　　D. UC5

（42）A. UC1　　B. UC2　　C. UC3　　D. UC5

● （43）设计模式定义了对象间的一种一对多的依赖关系，以便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自动刷新。

（43）A. Adapter（适配器）　　B. Iterator（迭代器）

C. Prototype（原型）　　D. Observer（观察者）

● UML 中有多种类型的图，其中，（44）对系统的使用方式进行分类，（45）显示了类及其相互关系，（46）显示人或对象的活动，其方式类似于流程图，通信图显示在某种情况下对象之间发送的消息，（47）与通信图类似，但强调的是顺序而不是连接。

（44）A. 用例图　　B. 顺序图　　C. 类图　　D. 活动图

（45）A. 用例图　　B. 顺序图　　C. 类图　　D. 活动图

（46）A. 用例图　　B. 顺序图　　C. 类图　　D. 活动图

（47）A. 用例图　　B. 顺序图　　C. 类图　　D. 活动图

● 正则表达式 1*(0|01)*表示的集合元素的特点是（48）。

（48）A. 长度为奇数的 0、1 串

B. 开始和结尾字符必须为 1 的 0、1 串

C. 串的长度为偶数的 0、1 串

D. 不包含子串 011 的 0、1 串

● 设某程序中定义了全局整型变量 x 和 r，且函数 f()的定义如下所示，则在语句“x = r*r +1 ;”中（49）。

```
int f(int r)
{   int x;
    x = r*r +1 ;
    return x;
}
```

（49）A. x 和 r 均是全局变量　　B. x 是全局变量、r 是形式参数

C. x 是局部变量、r 是形式参数　　D. x 是局部变量、r 是全局变量

● 程序语言的大多数语法现象可用上下文无关文法描述。对于一个上下文无关文法 G=(N，T，P，S)，其中 N 是非终结符号的集合，T 是终结符号的集合，P 是产生式集合，S 是开始符号。令集合 V= N∪T，那么 G 所描述的语言是__（50）__的集合。

（50）A. 从 S 出发推导出的包含 V 中所有符号的串

B. 从 S 出发推导出的仅包含 T 中符号的串

C. N 中所有符号组成的串

D. T 中所有符号组成的串

● 在数据库系统中，数据的完整性约束的建立需要通过数据库管理系统提供的__（51）__语言来实现。

（51）A. 数据定义　　B. 数据操作　　C. 数据查询　　D. 数据控制

● 若某个关系的主码为全码，则该主码应包含__（52）__。

（52）A. 单个属性　　B. 两个属性　　C. 多个属性　　D. 全部属性

● 建立一个供应商、零件数据库。其中“供应商”表 S（Sno，Sname，Zip，City）分别表示：供应商代码、供应商名、供应商邮编、供应商所在城市，其函数依赖为：Sno→（Sname，Zip，City ），Zip→City。“供应商”表 S 属于__（53）__。

（53）A. 1NF　　B. 2NF　　C. 3NF　　D. BCNF

● 关系 R、S 如下图所示，R⋈S 可由__（54）__基本的关系运算组成，R⋈S =__（55）__。

A	B	C
a	b	c
b	a	d
c	d	e
d	f	g

R

A	C	D
a	c	d
d	f	g
b	d	g

S

（54）A. π、σ 和×　　B. −、σ 和×　　C. ∩、σ 和×　　D. π、σ 和∩

（55）A.

A	B	C
a	b	c
b	a	d
c	d	e

B.

A	B	C	D
a	b	c	d
b	a	d	g
d	f	g	g

C.

A	B	C
a	b	c
b	a	d

D.

A	B	C	D
a	b	c	d
b	a	d	g

● 若事务 T1 对数据 A 已加排它锁，那么其它事务对数据 A（56）。

（56）A. 加共享锁成功，加排它锁失败　　B. 加排它锁成功，加共享锁失败
　　C. 加共享锁、加排它锁都成功　　D. 加共享锁、加排它锁都失败

● 拓扑排序是指有向图中的所有顶点排成一个线性序列的过程，若在有向图中从顶点 v_i 到 v_j 有一条路径，则在该线性序列中，顶点 v_i 必然在顶点 v_j 之前。因此，若不能得到全部顶点的拓扑排序序列，则说明该有向图一定（57）。

（57）A. 包含回路　　B. 是强连通图
　　C. 是完全图　　D. 是有向树

● 设栈 S 和队列 Q 的初始状态为空，元素按照 a、b、c、d、e 的次序进入栈 S，当一个元素从栈中出来后立即进入队列 Q。若队列的输出元素序列是 c、d、b、a、e，则元素的出栈顺序是（58），栈 S 的容量至少为（59）。

（58）A. a、b、c、d、e　　B. e、d、c、b、a
　　C. c、d、b、a、e　　D. e、a、b、d、c

（59）A. 2　　B. 3　　C. 4　　D. 5

● 对于 n（n≥0）个元素构成的线性序列 L，在（60）时适合采用链式存储结构。

（60）A. 需要频繁修改 L 中元素的值　　B. 需要频繁地对 L 进行随机查找
　　C. 需要频繁地对 L 进行删除和插入操作　　D. 要求 L 存储密度高

● 对于二叉查找树（Binary Search Tree），若其左子树非空，则左子树上所有结点的值均小于根结点的值；若其右子树非空，则右子树上所有结点的值均大于根结点的值；左、右子树本身就是两棵二叉查找树。因此，对任意一棵二叉查找树进行（61）遍历可以得到一个结点元素的递增序列。在具有 n 个结点的二叉查找树上进行查找运算，最坏情况下的算法复杂度为（62）。

（61） A. 先序　　B. 中序　　C. 后序　　D. 层序

（62） A. $O(n^2)$　　B. $O(n\log_2 n)$　　C. $O(\log_2 n)$　　D. $O(n)$

● 迪杰斯特拉（Dijkstra)算法按照路径长度递增的方式求解单源点最短路径问题，该算法运用了（63）算法策略。

（63）A. 贪心　　B. 分而治之　　C. 动态规划　　D. 试探＋回溯

● 关于算法与数据结构的关系，（64）是正确的。

（64）A. 算法的实现依赖于数据结构的设计

B. 算法的效率与数据结构无关

C. 数据结构越复杂，算法的效率越高

D. 数据结构越简单，算法的效率越高

● 若一个问题既可以用迭代方式也可以用递归方式求解，则（65）方法具有更高的时空效率。

（65）A. 迭代　　B. 递归

C. 先递归后迭代　　D. 先迭代后递归

● 在 FTP 协议中，控制连接是由（66）主动建立的。

（66）A. 服务器端　　B. 客户端　　C. 操作系统　　D. 服务提供商

● 网页中代码<input type=text name="foo" size=20>定义了（67）。

（67）A. 一个单选框

B. 一个单行文本输入框

C. 一个提交按纽

D. 一个使用图像的提交按纽

● 电子邮件应用程序利用POP3协议（68）。

（68）A. 创建邮件　　B. 加密邮件

C. 发送邮件　　D. 接收邮件

● 在进行金融业务系统的网络设计时，应该优先考虑（69）原则。在进行企业网络的需求分析时，应该首先进行（70）。

（69）A. 先进性　　B.开放性　　C. 经济性　　D. 高可用性

（70）A. 企业应用分析　　B. 网络流量分析

C. 外部通信环境调研　　D. 数据流向图分析

● The Rational Unified Process (RUP) is a software engineering process, which captures many of best practices in modern software development. The notions of （71） and scenarios have been proven to be an excellent way to capture function requirements. RUP can be described in two dimensions – time and content. In the time dimension, the software lifecycle is broken

into cycles. Each cycle is divided into four consecutive （72） which is concluded with a well-defined （73） and can be further broken down into （74） – a complete development loop resulting in a release of an executable product, a subset of the final product under development, which grows incrementally to become the final system. The content structure refers to the disciplines, which group （75） logically by nature.

（71）A.artifacts	B. use-cases	C. actors	D. workers
（72）A. orientations	B. views	C. aspects	D. phases
（73）A. milestone	B. end-mark	C. measure	D. criteria
（74）A. rounds	B. loops	C. iterations	D. circularities
（75）A. functions	B. workflows	C. actions	D. activities