软件测试与质量 期末复习

单项选择10题 20分 + 简答题4题 20分 + 应用题3题 60分
应用题：按照某一种原则（等价类/边界值）对测试描述（白盒/黑盒/功能/结构性）进行测试设计
格式：输入+预期输出+（备注、操作细节说明）

1. 软件测试的基本思想

软件生存周期：软件计划与可行性研究（问题定义、可行性研究）、需求分析、软件设计（概要设计与详细设计）、编码、软件测试、运行与维护。

软件测试的技术与过程:

（1）测试需求：测试需求并不等同于软件需求，它是以测试的观点根据软件需求整理出一个Checklist。

（2）测试设计：

- Criteria Based：设计测试数值去满足覆盖规则；
- Human Based：基于程序的领域知识和测试技术。

（3）测试执行：在软件上运行测试并且记录结果

（4）测试评估：评估测试的结果

（5）测试自动化：将测试数据写入可运行脚本。

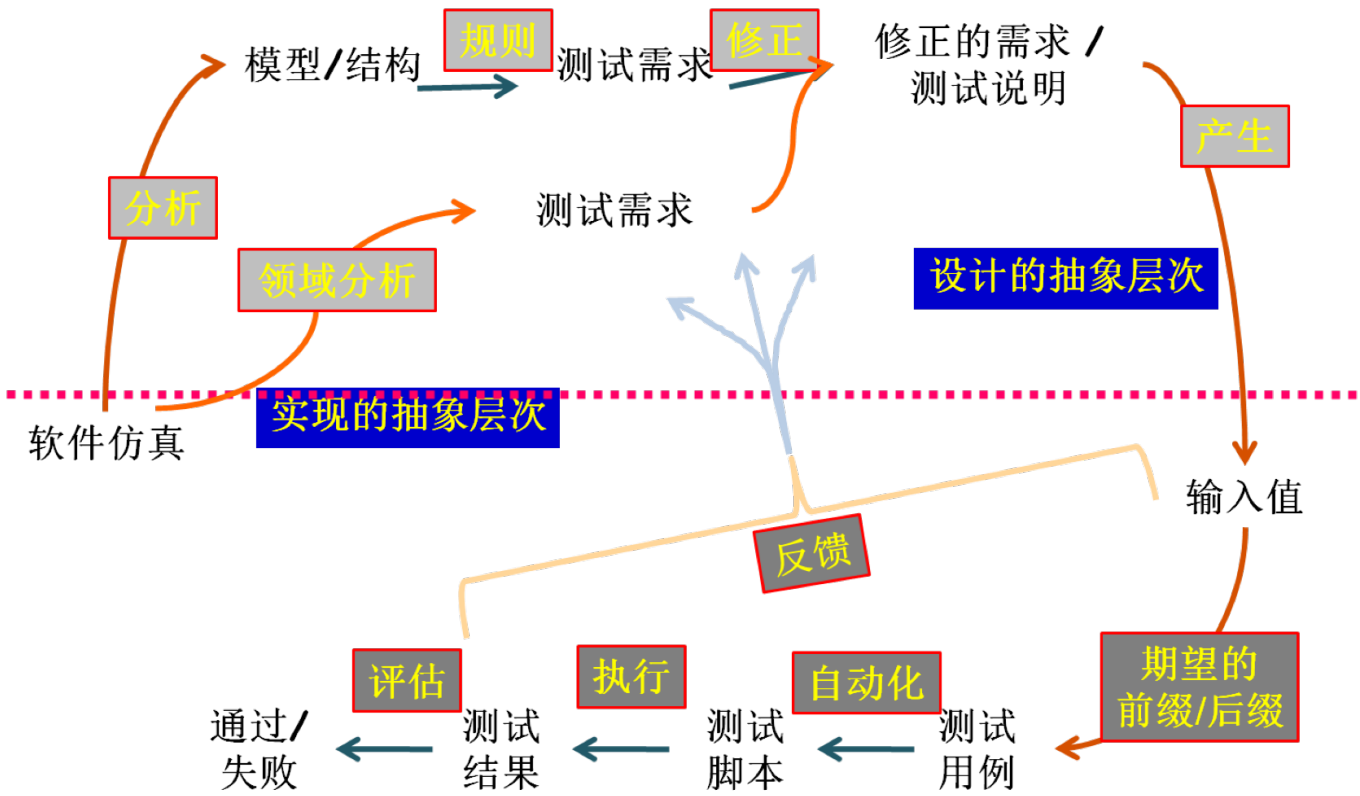
其他活动：

（6）测试管理：制定策略，组织团队，开发之间的接口，选择测试规则，决定所需要的自动化程度

（7）测试维护：保存测试用例以供软件衍化时的复用，需要测试设计人员和自动化人员的合作，决定何时整理测试套件既需要策略又需要技巧，测试必须纳入配置管理

（8）测试文档：需要各方参与，每个测试都需记录“为什么要这么做”—规则和满足测试需求或者人工设计的测试基本原理，保证整个过程可追溯，在自动化的测试中做记录

模型驱动式软件测试



持续的软件测试：

- 单元测试：是对软件设计的最小单位：模块的测试，目的是检验每个软件单元能否正确地实现其功能满足其性能和接口要求。单元测试验证程序与详细设计说明的一致性 下面可能会有模块测试，和单元测试差不多
- 集成测试：将经过单元测试的模块逐步进行集成和测试。集成测试验证程序和概要设计说明的一致性

- 系统测试：指软件与该软件所属的系统对接并测试其接口的过程，目的是在真实工作环境下检验软件是否能与系统正确连接，并满足软件研制的系统目标
- 验收测试：是检验所开发的软件是否按软件需求规格说明中确定的软件功能、性能、约束及限制等技术要求进行工作

测试和软件开发模型

- 测试和软件的开发模型（开发生命周期的测试活动嵌入，测试活动需要被嵌入到生命周期中）
- 比较晚的介入方式（测试晚于开发）
- 极限编程的测试驱动开发（先写测试用例）
- 专注于软件测试的软件开发模型（某一个开发活动与某个测试相关，会影响测试的标准和目标，所以要早期进行测试的计划和设计）

2. 软件测试中的若干问题

verification 软件是否实现了需求

validation 确认需求是否正确

1）Validation & Verification：

Validation（确认）：检查软件在最终的运行环境上是否达到预期的目标；

Verification（验证）：检查某样东西是否符合之前已定好的标准。

测试的目的：有不同的观点，一种观点认为软件测试应该把软件验证纳入工作范围内，另一种观点认为发现bug是软件测试的终极目标。验证扩展是指广义的测试，找到bug是狭义的测试。但是验证和确认是必不可少的，所以大多数认为是广义的软件测试，但是随着软件规模的扩大，验证活动也变得困难，比如可靠性测试就不好验证；为了排除心理障碍，也可以以找到bug作为终极目标，实用性较强

2）测试&质量保障（QA）：

测试：目的是寻找bug，尽早地发现他们确保他们已经被修正。

质量保障（QA）：目的是创建一个强制执行的标准和方法去提高开发过程以便了防止Bug的出现。

3）静态测试&动态测试：

静态测试：不运行被测程序本身，仅通过分析或检查源程序的语法、结构、过程、接口等来检查程序的正确性；

优点：成本低

动态测试：通过运行被测程序，检查运行结果与预期结果的差异，并分析运行效率和健壮性等性能。

缺点：成本高

优点：单边完备（如果发现问题，证明一定有问题；如果没发现问题，不一定证明没有问题）

4）Faults, Errors & Failures：

Fault（故障）：软件中的静态缺陷代码（开发、debug）

Error（错误）：错误的内部中间状态（联系fault和failure，完成二者之间的循环）

Failure（失效）：外部行为与预期不一样（失效）——通常测试指的是第三部分

发现bug的条件 PIE模型

可达性（reachability）：包含缺陷的代码地址必须在软件运行时刻到达

可感染（Infection）：必须出现程序的错误状态

可波及（Propagation）：被感染的状态必须可以导致一些输出的错

1. 是不是所有代码都能执行到？不一定
2. 执行到有缺陷的代码不代表一定产生错误的结果
3. 有可能任何测试输入都不能检测出bug

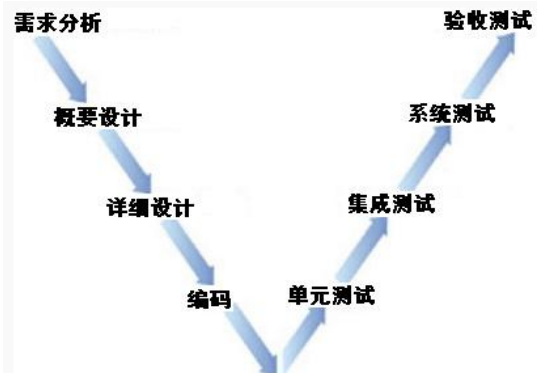
5）**测试&调试**：
测试：目的是显示存在的故障；
调试：目的是发现故障所在，并修复程序。

6）**测试用例**：
是为某个特殊目标而编制的一组测试输入、执行条件（环境）以及预期输出（测试预言），以便测试某个程序是否满足某个特定需求。

灰盒测试：不仅关注输出、输入的正确性，同时也关注程序内部的情况。
常常通过一些表征性的现象、事件、标志来判断内部的运行状态。

7）**黑盒测试&白盒测试**：
黑盒测试：不需要源代码，运行程序并不需要知道程序内部的信息；
白盒测试：是通过程序的源代码进行测试而不使用用户界面。

8）**V模型**：
软件测试的位置是不是固定的，有没有扩大的可能——软件测试的模型：V模型/W模型，全程化软件测试（如何全程化），需求–>系统级别测试，设计–>集成测试，代码–>单元测试



9）**PIE模型**
Execution（执行）：故障代码被执行到；
Infection（感染）：产生错误的中间状态；
Propagation（传播）：被感染的状态传播到系统外部被观察和发现。

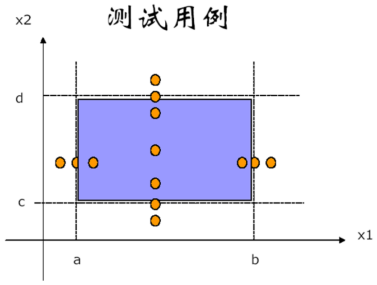
10）**可观察性&可控制性**
软件可观察性（Software Observability）：以其输出、对环境以及其他硬件软件部分的作用作为考量观察程序的容易程度。对设备，数据库或者远程文件有影响的软件具有较低的可观察性
软件可控制性（Software Controllability）：以数值、操作和行为为考量，为程序提供输入的容易程度。通过键盘很容易对软件进行控制；通过硬件传感器或者远程的分布式软件输入会难一点；数据抽象会降低可观察性和可控制性

11）**对可控制性和可观察新具有影响的输入**
Prefix Values：将软件切换至接受测试用例数据的正确状态的任何必要的输入。有两类postfix values，确认值（Verification Values）：查看测试用例结果的必要输入之；退出命令（Exit Commands）：终止程序或者将其返回到稳定状态时所需要输入的数据
Postfix Values：在测试用例数据之后必须被传到程序之中的数据
可运行测试脚本（Executable Test Script）：一个以在被测试软件上自动执行并且输出结果的形式编写的测试用例

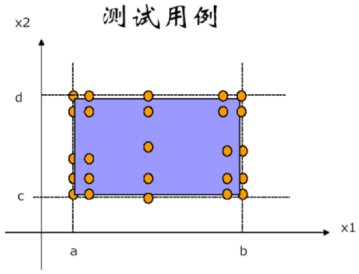
12）**其他**
- 穷举测试、全面测试是否存在？不存在，房间里扫灰尘总是扫不干净的，软件中穷举测试也是不可能的。人脑思维可能产生各种各样的错误，反映在代码里就产生各种各样的错误。这些错误可能会在各种情况下被引发（如被用户），但也有一些在整个产品生命周期中都不会被引发。软件测试的目的是找到尽可能多的bug，而不是为了找到或消灭所有的bug——方式：让用户少引发潜在的错误或缺陷。

- 对于bug的推断：一个模块如果容易查找到bug，那么这个模块中找到新的bug的可能性大还是小？更多，这个模块需要进行更详细的测试。
- 测试与开发的关系：以正能量的方式回答。软件质量的最终保障是由开发过程决定的，但是开发过程可能存在问题，需要软件测试做一些保证，最终目的是软件质量的进一步提升。软件测试的目标和开发的最终目标是一样的
- 功能测试：边界值/等价类/决策表（因果图）

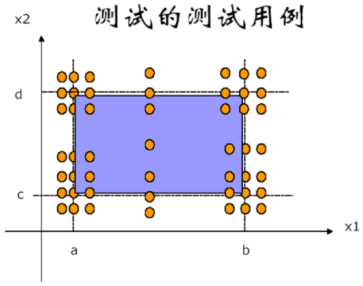
两个变量函数的健壮性测试的测试用例



两个变量函数的最坏情况测试的测试用例



两个变量函数的健壮最坏情况测试的测试用例



3. 边界值测试

边界测试就是对输入或输出范围的边界值进行测试的一种黑盒测试方法。其测试用例来自等价类的边界。

健壮性边界测试：又称为容错性测试，用于测试系统在出现故障时，是否能够自动恢复或者忽略故障继续运行。除了变量的五个边界值（min,min+,nom,max−,max）分析取值，还要通过采用一个略超过最大值（max+）的取值，以及一个略小于最小值（min−）的取值。（非法输入值）

随机测试：

- 随机测试是根据测试说明书执行样例测试的重要补充手段，是保证测试覆盖完整性的有效方式和过程。主要是对被测软件的一些重要功能进行复测，也包括测试那些当前的测试样例没有覆盖到的部分。
- 对于软件更新和新增加的功能要重点测试。重点对一些特殊点情况点、特殊的使用环境、并发性、进行检查。尤其对以前测试发现的重大Bug，进行再次测试，可以结合回归测试一起进行。
- 理论上，每一个被测软件版本都需要执行随机测试，尤其对于最后的将要发布的版本更要重视随机测试
- 随机测试最好由具有丰富测试经验的熟悉被测软件的测试人员进行测试。对于被测试的软件越熟悉，执行随机测试越容易。

边界值测试方针：

（1）如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。

（2）如果输入条件规定了值的个数，则用最大个数，最小个数，比最小个数少一，比最大个数多一的数作为测试数据。

（3）应用于输出条件，即设计测试用例使输出值达到边界值及其左右的值。

（4）如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试用例。

（5）如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试用例。

4. 等价类测试

一种黑盒测试用例设计方法，把所有可能的输入数据，即程序的输入域划分成若干部分（子集），然后从每一个子集中选取少数具有代表性的数据作为测试用例。

等价类划分基础知识

等价类是指某个中，各个输入数据对于揭露程序中的错误都是等效的。

无冗余（两两互补相交）、完备性（所有Ii并集为整个集合I)

等价类划分方法

1) 如果输入条件规定了取值范围，可定义一个有效等价类和两个无效等价类。

例：输入值是学生成绩，范围是0～100

有效等价类：①0≤成绩≤100，无效等价类：①成绩<0，②成绩>100

2) 如规定了输入数据的一组值，且程序对不同输入值做不同处理，则每个允许的输入值是一个有效等价类，并有一个无效等价类（所有不允许的输入值的集合）。

例：输入条件说明学历可为:专科、本科、硕士、博士四种之一

有效等价类：①专科、②本科、③硕士、④博士，无效等价类：①其它任何学历

3) 如果规定了输入数据的个数，则类似地可以划分出一个有效等价类和两个无效等价类。

例：一个学生每学期只能选修1～3门课

有效等价类：①选修1～3门，无效等价类：①不选②选修超过3门

4) 如果规定了输入数据必须遵循的规则，可确定一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。

例：校内电话号码拨外线为9开头

有效等价类：①9＋外线号码，无效等价类：①非9开头＋外线号码②9＋非外线号码，…

等价类测试的四种类型（可以看图示）

（1）弱一般等价类测试：假设只有一个缺陷或者输入变量相互独立；

（2）强一般等价类测试：与弱一般等价类相似，但是它关注多种缺陷因素和变量之间的依赖，各种变量的每个等价类组合都需要被包括；

（3）弱健壮等价类测试：与弱一般等价类测试相似，但是考虑了输入的非法值；

（4）强健壮等价类测试：与强一般等价类测试类似，但是考虑了输入的非法值。

一般基于等价类的测试方法

（1）按照规格说明中的“输入条件”（或者输出条件）划分等价类：有效等价类、无效等价类

（2）设计一个新的测试用例，使其尽可能多的覆盖尚未覆盖的有效等价类；重复这一步骤，直到所有的有效等价类都被覆盖为止

（3）设计一个新的测试用例，使其仅覆盖一个无效等价类；重复这一步骤，直到所有的无效等价类都被覆盖为止

5. 静态测试

定义：不执行程序代码而寻找代码中可能存在的错误或评估程序代码的过程。静态测试可以手工进行，也可以借助软件工具自动进行。

特点：静态测试不必动态的执行程序，也就是不必进行测试用例设计和结果判读等工作；静态测试可以由人手工方式进行，充分发挥人的优势，行之有效；静态测试实施不需要特别条件，容易开展。

静态测试技术1 代码审查：

- 测试内容：检查代码和设计的一致性；检查代码对标准的遵循、可读性；检查代码的逻辑表达的正确性；检查代码结构的合理性。
- 组成和方式：代码审查由一组程序和错误检查技术组成，以代码审查组方式组织。代码审查组一般为4个人，其中一人为组长，还包括资深程序员、程序编写者与专职测试人员

- 步骤：准备、程序阅读、审查会议、跟踪及报告

静态测试技术2 代码走查：

- 代码走查与代码审查相似，它也是由一组程序和错误检查技术组成，只是程序和错误检查技术不完全相同
- 组成和方式：代码走查以小组方式进行。代码走查组包括组长、秘书（负责记录）、测试人员。走查过程与审查相似。
- 走查会议内容：与代码审查不同，不是读程序和使用代码审查单，而是由被指定的作为测试员的小组成员提供若干测试用例，让参加会的成员当计算机，在会议上对每个测试用例用头脑来执行程序，也就是用测试用例沿程序逻辑走一遍，并由测试人员讲述程序执行过程，在纸上或黑板上监视程序状态。如果发现问题由秘书记下来，中间不讨论任何纠错问题，主要是发现错误。
- 缺点：代码走查使用测试用例启发检测错误，人们注意力会相对集中在随测试用例游历的程序逻辑路径上，不如代码审查检查的范围广，错误覆盖面全。

静态测试的内容

（1）需求定义的静态测试：对需求定义的测试着重于测试对用户需求的描述和解释是否完整、准确。
对照条例：兼容性、完备性、一致性、正确性、可行性、易修改性、健壮性、易追溯性、易理解性、易测试性和可验证性。
（2）设计文档的静态测试：对设计文档的静态测试着重于分析设计是否与需求定义一致，所采用的数值方法和算法是否适用于待解问题，程序的设计中对程序的划分是否与待解问题相适应，需求是否都被满足了等等。
对照条例：完备性、一致性、正确性、可行性、易修改性、模块性、可预测性、结构化、易追溯性、易理解性、可验证性/易测试性。
（3）源代码的静态测试：对源代码的静态测试着重于分析实现是否正确、完备。
对照条例：完备性、一致性、正确性、易修改性、可预测性、健壮性、结构化、易追溯性、易理解性、可验证性。

6. 结构性测试–控制流测试

- 语句覆盖SC：设计若干测试用例，运行被测程序，使程序中每个可执行语句至少执行一次
- 判定覆盖DC：设计若干测试用例，运行被测程序，使得程序中每个分支的取真值和取假值至少一次，即判断真假值均曾被满足。
- 条件覆盖CC：设计若干测试用例，执行被测程序以后要使每个判断中每个条件的可能取值至少满足一次
- 条件判定组合覆盖CDC：设计足够的测试用例，使得判断条件中的所有条件可能至少执行一次取值，同时，所有判断的可能结果至少执行一次。 测试用例要满足以下条件：所有条件可能至少执行一次取值；所有判断的可能结果至少执行一次。
- 多条件覆盖MCC：设计足够的测试用例，使得所有可能的条件取值组合至少执行一次，又称条件组合覆盖
- 修正条件判定覆盖MCDC：要求在一个程序中每一种输入输出至少得出现一次，在程序中的每一个条件必须产生所有可能的输出结果至少一次，并且每一个判定中的每一个条件必须能够独立影响一个判定的输出，即在其他条件不变的前提下仅改变这个条件的值，而使判定结果改变。
  条件的含义：不含有布尔操作符号的布尔表达式
  判定的含义：判定表示由条件和零或者很多布尔操作符号所组成的一个布尔表达式
- 路径覆盖：设计所有的测试用例，来覆盖程序中的所有可能的执行路径

- Int function1（bool a, bool b, bool c） {
  Int x = 0;

```
if （a && （b||c））
   x=1;
return x;
}
```

语句覆盖：

	a	b	c
case1	T	T	T

判定覆盖：

	a	b	c	a && (b||c)	a || (b||c)	判定覆盖/%
case1	T	T	T	T	T	50
case2	F	F	F	F	F	50

条件覆盖：

	a	b	c	a && (b||c)	条件覆盖/%	判定覆盖/%
case3	F	T	T	F	100	50
case4	T	F	F	F		

条件判定组合覆盖：

	a	b	c	a && (b||c)	a || (b||c)	条件判定组合覆盖/%
case1	T	T	T	T	T	100
case2	F	F	F	F	F	

- 基本路径测试方法：

1. 绘制程序的控制流图
2. 计算McCabe圈复杂度（设为n）：V(G) = e – n + 2 = d + 1
3. 确定基本路径集的确定，即构造n条独立路径

（1） 任意构造一条从（唯一）入口结点到（唯一）出口结点的路径，将该路径加入基本路径集

（2） 修改基本路径集中路径，至少经过一条以前未走过的边，将新路径加入基本路径集

重复第（2）步，直到基本路径集中包含n条路径

4.设计测试用例，使基本路径集中的路径能走通

7. 集成测试

将经过单元测试的模块按照设计要求连接起来，组成所规定的软件系统的过程称为“集成”。

主要任务：

- 将各模块连接起来，检查模块相互调用时，数据经过接口是否丢失；
- 将各个子功能组合起来，检查能否达到预期要求的各项功能；
- 一个模块的功能是否会对另一个模块的功能产生不利的影响；
- 全局数据结构是否有问题，会不会被异常修改；
- 单个模块的误差积累起来，是否被放大，从而达到不可接受的程度。

必要性：实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。由于在接口部分存在问题，程序在某些局部反映不出来的问题，在全局上很可能暴露出来，影响功能的实现。
驱动模块：用以模拟被测模块的上级模块。接受测试数据，把相关的数据传送给被测模块，启动被测模块，并获得相应的结果。
桩模块：用以模拟被测模块工作过程中所调用的模块。由被测模块调用，一般只进行很少的数据处理，例如打印入口和返回，以便于检验被测模块与其下级模块的接口。

与单元测试和系统测试的比较：

测试类型	对象	目的	测试依据	测试方法
单元测试	局部模块	消除局部模块内的逻辑和功能上的错误和缺陷	模块逻辑设计，外部说明	白盒为主
集成测试	模块间的集成和调用关系	找出与设计相关的程序结构、模块调用关系、模块间接口方面的问题	程序结构设计	白盒+黑盒，采用较多黑盒方法构造测试用例
系统测试	整个系统（包括硬件等）	对整个系统进行一系列的整体、有效性测试	系统结构设计、需求规格说明等	黑盒

集成测试的方法：
1. 逐步集成（增量集成）
•自顶向下增量式测试：按结构图自上而下逐步集成和逐步测试；首先集成主控模块（主程序），然后按照软件控制层次结构向下进行集成
•自底向上增量式测试：从最底层的模块开始，按结构图自下而上逐步进行集成和测试
•混合集成（三明治集成）：以中间层为目标层，以上用自顶向下，以下用自底向上

集成方法	优点	缺点	使用情况
自顶向下 增量式测试	测试和集成可以较早开始； 减少驱动器的开发； 底层接口修改灵活。	桩模块的开发代价较大； 底层模块无法意料的需求可能促使顶层修改； 底层模块测试可能不充分。	增量式开发； 并行软件开发。
自底向上 增量式测试	无需构造桩模块，桩模块往往千差万别； 驱动模块具有某种统一性，且随着测试层次的提供，驱动模块数量减少； 涉及复杂算法和真正输入、输出的模块一般在底层，且是较易出错的模块。可以尽早发现错误； 各子树的集成和测试可以并行。	驱动模块开发量大； 高层模块的可操作性和互操作性测试不充分。	重要模块在底层
混合集成 （三明治集成）	它将自顶向下和自底向上的集成方法有机地结合起来，不需要写桩程序因为在测试初自底向上集成已经验证了底层模块的正确性	在真正集成之前每一个独立的模块没有完全测试过。	

2. 一次性集成（非增量集成）
优点：迅速完成集成测试；测试用例较少
缺点：错误难以定位；即使通过测试，许多接口错误也可能隐藏
适用范围：小的、良构的系统，其模块已经接受了充分测试；一个已经存在的系统，只有少量修改；通过复用可信赖的模块构造系统。

8. 系统和外部测试

系统测试的目的：验证系统是否满足了需求规格的定义，找出与需求规格不符或与之矛盾的地方，从而提出更加完善的方案。
系统测试的各种类型：
（1）**恢复测试**：恢复测试作为一种系统测试，主要关注导致软件运行失败的各种条件，并验证其恢复过程能否正确执行。在特定情况下，系统需具备容错能力。另外，系统失效必须在规定时间段内被更正，否则将会导致严重的经济损失。
（2）**安全测试**：安全测试用来验证系统内部的保护机制，以防止非法侵入。在安全测试中，测试人员扮演试图侵入系统的角色，采用各种办法试图突破防线。因此系统安全设计的准则时要想方设法使侵入系统所需的代价更加昂贵。
（3）**压力测试**：压力测试是指在正常资源下使用异常的访问量、频率或数据量来执行系统。在压力测试中可执行以下测试：
1）如果平均中断数量是每秒一到两次，那么设计特殊的测试用例产生每秒十次中断。
2）输入数据量增加一个量级，确定输入功能将如何响应。
3）在虚拟操作系统下，产生需要最大内存量或其它资源的测试用例，或产生需要过量磁盘存储的数据。

α和β测试：
（1）**α测试**：指软件开发公司组织内部人员模拟各类用户对即将面市软件产品（称为α版本）进行测试，试图发现错误并修正。α测试的关键在于尽可能逼真地模拟实际运行环境和用户对软件产品的操作并尽最大努力涵盖所有可能的用户操作方式。
（2）**β测试**：指软件开发公司组织各方面的典型用户在日常工作中实际使用β版本，并要求用户报告异常情况、提出批评意见，然后软件开发公司再对β版本进行改错和完善。β测试也是黑盒测试。黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。

9. 专项测试活动

（1）配置测试：配置测试是检查被测软件在各种硬件设备上的操作情况的过程。
如何确定是配置Bug：最准确的方法就是在另外一个配置完全不同的机器上一步一步执行产生Bug的操作，如果这种问题没有重现，那么很有可能是配置Bug。否则，有可能是普通缺陷。
识别硬件标准：了解一些硬件设备的细节可以帮助自己更好地做等价划分的决策
配置测试其他硬件：硬件、软件以及它们所连接的东西并不重要；如果连接到其他设备，配置需要被测试

步骤：
1）确定所需要的硬件；
2）确定现在能够得到的硬件品牌、模型和驱动设备；
3）确定所有可能用到的硬件特点、模式和选择（features, modes, and options）；
4）将所有确认过的硬件配置缩减到可管理的数量；
5）标记出软件在特定的配置下独特的风格；
6）设计在各个配置下运行的测试用例；
7）在每种配置下执行测试用例；
8）重复运行测试用例直到结果符合要求。

（2）**兼容性测试：**检测被测软件能否与其他软件正确地交互和共享信息。
被测软件需要和哪些其他软件相互兼容？定义被测软件和其他软件交互的兼容性标准和指导方针是什么？被测软件与其他软件或者平台之间共享信息的数据类型是什么？

平台和应用的版本：选择目标平台或者兼容的应用实际上是一个项目管理或者市场任务，他们会识别软件需要兼容的版本。
1）向后兼容/向前兼容：如果一个软件被称为向后/向前兼容，那么就说明这个软件可以与之前/之后的版本一起工作；
2）多版本测试的影响：不能测试在操作系统中成百上千的软件，所以只挑选重要的软件进行测试。
标准和指导方针
1）高层标准和指导方针：知道产品的总体运行情况、外观和使用体验、支持的风格等等；
2）低层标准和指导方针：底层标准是本质细节，比如文件格式和网络交换协议，我们应将底层标准看作是软件具体细节的拓展。
数据共享兼容性：一个支持遵循标准并且允许用户容易地发送信息到其他软件或者从其他软件获取信息的程序，是具有很高兼容性的产品（文件的保存、加载、导入、导出、复制、剪切、粘贴）。

（3）外国语言测试
国际化i18N （internationalization）：是设计一个可以不用在工程上进行改变便可适用于各个语言和地区的软件应用的过程。国际化是软件开发者的任务。
特点：
- 通过添加本地化的数据，这个软件可以在全球各个地方运行；
- 文本元素，如状态信息和图形用户界面组件的label，不是通过硬编码实现的，而是保存在源代码之外可以动态提取的；
- 添加新的语言支持不需要重新编译；
- 与文化相关的数据，如日期等，以符合某个语言和地区的格式类型出现；
- 可以很快地被本地化。

如何使软件国际化：
- 对于网页来说，所有需要本地化的字符串都应由tag括起来
- 对于二进制的客户端，所有的GUI字符串都应保存在资源文件中，而不是通过硬编码写在程序代码中

语言翻译的问题：热键和快捷方式、拓展的字符、字符数量的估算、从左向右阅读还是从右向左阅读、图形中的文本、保留代码中的文本
网页中的国际化中的问题：布局和用户接口、无序的代码和混乱的文本、字符索引、欧亚的全名差异

本地化L10N （localization）：将一个国际化的软件翻译和调整到某个特定的语言和文化的过程。总的来说，国际化被认为是一个工程的过程，而本地化被认为是一个翻译的过程。
本地化的问题：
- 内容：内容是除了源代码以外，进入产品的其他“资料”。我们需要考虑组成最终产品的所有成分；
- 数据格式：不同的地区使用不同的格式来组织数据单元。所以在测试的过程中，我们需要对测量的单元很熟悉。

全球化G11N （Globalization）：是一个包括了国际化和本地化的广义的称呼，G11N = I18N + L10N。
G11N测试：为了确定软件的全球化程度。
测试包含的内容：国际化测试、本地化测试、语言/翻译测试、用户界面测试、功能测试、更多的功能测试、发布测试。

（4）易用性测试（UI）

易用性的含义：易被发现、易于学习、易于使用、易于获得。

用户界面/易用性测试：

- UI被称作用户界面：从用户那里获取输入、将输出反馈给用户
- 可用性反映了软件交互的合理性、功能性和有效性
- 一个好的UI决定了一个软件的易用性
- GUI需要经过可用性测试
- 易用性测试的主要部分便是用户界面的测试

好的用户界面所具备的品质：遵循标准和指导、满足直觉特征、正确性、一致性、灵活性、使用舒适、易于使用、简单

遵循标准和指导：

- 如果没有很好的不遵循标准的理由，那么最好遵循已有的标准；
- 经过大量的测试、尝试和错误，已经产生了许多衡量一个软件是否用户友好的标准；
- 可以为自己的软件单独制定标准

直觉：用户界面简洁、良好的布局、功能之间易于切换、没有过度堆砌的功能、帮助系统真的能够起到帮助作用

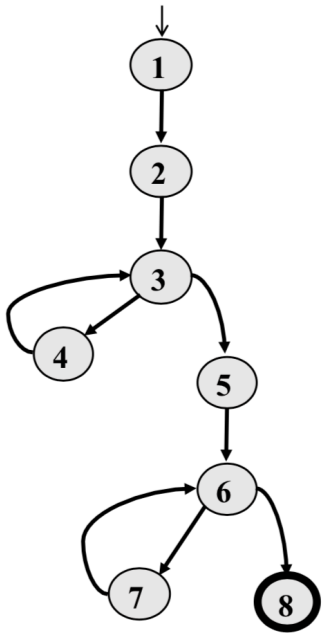
正确性：市场差别、语言和拼写、媒介问题、所见即所得

一致性：快捷键和菜单选项、使用正确的术语和命名、音频、键盘/按钮等价和布局、灵活性：状态的跳转、状态的终止和跳过、数据的输入输出、使用舒适：界面的合理性/错误处理/性能、简单

残障人员对软件的使用：法律已经有相关条例规定，在软件设计过程中需要考虑

10. 性能测试

- **负载测试**：用来观察系统在一定负载下的表现，这个负载一般在系统可承受范围内，比如测试一定数量用户同时在线时系统的表现
- 压力测试：用来了解系统能承受的负载的最大值，来测试系统的鲁棒性
- 持久性能测试：给系统加以持久性的一定负载，观察系统的表现
- 峰值测试：突然加大负载，观察系统的表现



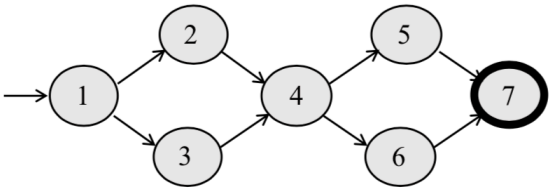

```
public static void CSta (int [ ] numbers)
{
    int length = numbers.length;
    double var, mean, sum, varsum;

    sum = 0.0;
    for (int i = 0; i < length; i++)
    {
        sum += numbers [ i ];
    }
    mean = sum / (double) length;

    varsum = 0.0;
    for (int i = 0; i < length; i++)
    {
        varsum = varsum + ((numbers [i] - mean) * (numbers [i] - mean));
    }
    var = varsum / ( length - 1.0 );

    System.out.println ("length:              " + length);
    System.out.println ("mean:                " + mean);
    System.out.println ("variance:            " + var);
}
```

Source Code

• 配置测试：改变系统的配置组件，分别观察系统的表现
• 隔离测试：重复做能发现系统某一个问题的一个测试，常用来隔离并确定这个错误域

11. 流图与路径

源代码的行对应和真正流图的对应有差异
3号节点是for循环的开始，但是i = 0的初始化在2号节点
1号节点和2号节点可以合并，但理论可以保证均为等价



讨论
1.一个点算不算一个图？算
2.V是有限集合，E也是有限集合

要求图只能有一个输入和一个输出，如果有多个输入和多个输出，可以加一个虚拟入口/出口，转化为唯一的输入口和输出口

路径
路径有很多，特别的，单点路径长度为0
测试路径：图中有4条测试路径（初始节点为图的初始节点，结束节点为图的结束节点）

1.可能会存在测试路径，不能有测试输入路径这个路径
如果1和4有半段矛盾，就有一条语法可行的路径走不到
图中有路径，那么语法可行；能找到实际输入走这条路径，那么语义可行。
语义可行一定语法可行，但是语法可行不一定语义可行。

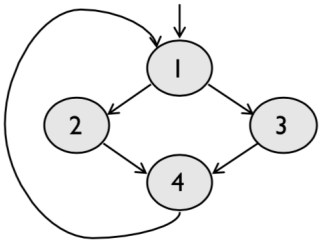
2.相同的输入走的路径不一定一样
典型反例：并发（这门课不考虑这种情况）

结构化覆盖
E：点是图最基本的元素，点覆盖（VC）就是源代码覆盖/界面元素覆盖 —— n=0的路径覆盖
I：边覆盖（EC）一定会满足点覆盖，点覆盖不一定满足边覆盖 —— n=1的路径覆盖
边对覆盖（EPC） —— n=2的路径覆盖

n越大，覆盖越强
边覆盖的测试用例集发现的bug不一定多于点覆盖的测试用例集发现的bug
在点覆盖的基础上扩张测试用例到边覆盖，那么边覆盖的测试用例集发现的bug一定多于点覆盖的测试用例集发现的bug

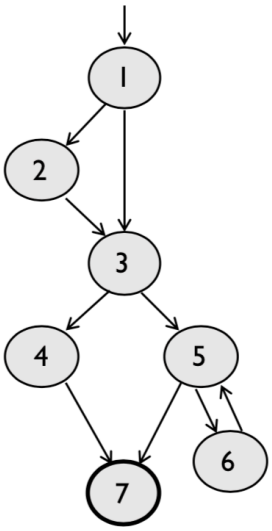
12. 简单路径和主路径

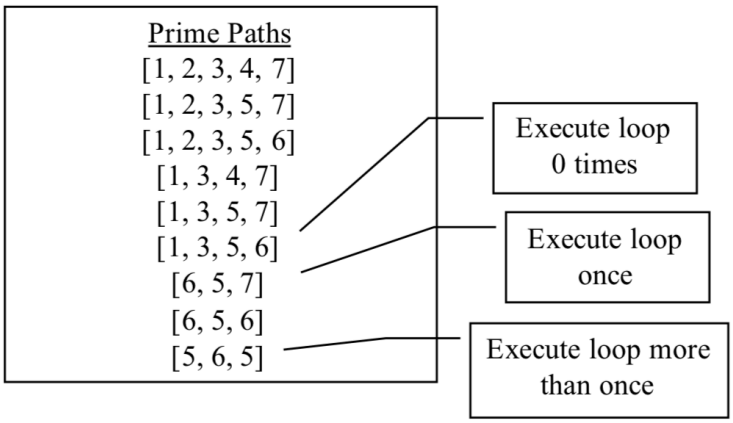
没有重复点，除非收尾节点在一起的路径叫做简单路径
圈是简单路径，单点也是简单路径
主路径：最长的简单路径
[2,4,1,2], [2,4,1,3], [1,3,4,1], [1,2,4,1],
[3,4,1,2], [4,1,3,4], [4,1,2,4], [3,4,1,3]



主路径覆盖（PPC）[必考]

[例1]

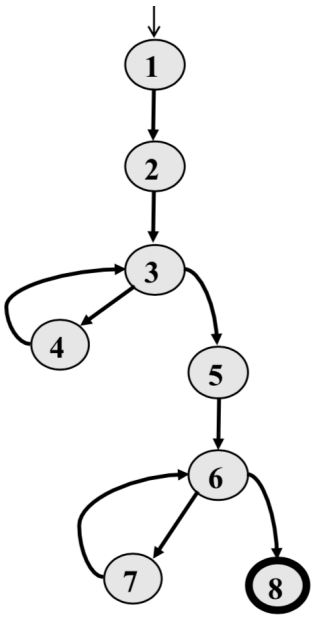




主路径：9条

测试路径：6条，[1,2,3,4,7], [1,2,3,5,7], [1,2,3,5,6,5,7], [1,3,4,7], [1,3,5,7], [1,3,5,6,5,7]

Prime Path Coverage	
TR	Test Path
A. [3, 4, 3]	i. [1, 2, 3, 4, 3, 5, 6, 7, 6, 8]
B. [4, 3, 4]	ii. [1, 2, 3, 4, 3, 4, 3, 5, 6, 7, 6, 7, 6, 8]
C. [7, 6, 7]	~~iii. [1, 2, 3, 4, 3, 5, 6, 8]~~
D. [7, 6, 8]	~~iv. [1, 2, 3, 5, 6, 7, 6, 8]~~
E. [6, 7, 6]	v. [1, 2, 3, 5, 6, 8]
F. [1, 2, 3, 4]	
G. [4, 3, 5, 6, 7]	
H. [4, 3, 5, 6, 8]	
I. [1, 2, 3, 5, 6, 7]	
J. [1, 2, 3, 5, 6, 8]	



基本路径覆盖 [必考]

- 基本路径测试是路径测试和分支测试的结合，这类测试满足了分支测试的需求，并且测试了在这个计算机程序中所有被用来构建任何随机路径的独立路径
- 圈复杂度：构造控制流图，复杂度=边−点+2
- 一个简单路径，满足其不会是任何其他简单路径的子路径
- 基本路径中不包括内部循环
- 各种测试强度比较：Path Testing>=BPT>=Branch Testing

13. AI系统测试

2016年真题

选择题：

2. JMeter中模拟用户测试的是哪个控件

3. 系统测试验证的是?

4. 用边界值分析法:10<=x<=100,X的取值应该是

5. Timer#1作用于Request的?（TestPlan->ThreadGroup）

问答题：

1. A xor （B and C） or D写出逻辑的100%MC/DC覆盖测试极小集合

2.某机构只允许16-35周岁报名（2010年12.31止）划分等价类；根据等价类划分设计测试用例

3.（给出简单的控制流程图）计算图复杂度;设计独立测试路径

大题通关

一、分析中国象棋中走马的实际情况

（下面未注明的均指的是对马的说明）

1、如果落点在棋盘外，则不移动棋子；

2、如果落点与起点不构成日字型，则不移动棋子；

3、如果落点处有自己方棋子，则不移动棋子；

4、如果在落点方向的邻近交叉点有棋子（绊马腿），则不移动棋子；

5、如果不属于1-4条，且落点处无棋子，则移动棋子；

6、如果不属于1-4条，且落点处为对方棋子(非老将)，则移动棋子并除去对方棋子；

7、如果不属于1-4条，且落点处为对方老将，则移动棋子，并提示战胜对方，游戏结束。

二、根据分析明确原因和结果

原因：

1、 落点在棋盘上；

2、 落点与起点构成日字；

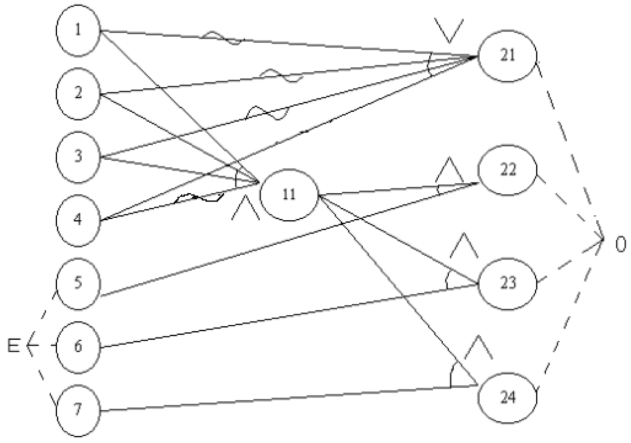
3、 落点处为自己方棋子；

4、 落点方向的邻近交叉点无棋子；

5、 落点处无棋子；

6、 落点处为对方棋子（非老将）；

7、 落点处为对方老将。



结果：

21、不移动棋子；

22、移动棋子；

23、移动棋子，并除去对方棋子；

24、移动棋子，并提示战胜对方，结束游戏。

添加中间节点11，目的是作为导出结果的进一步原因，简化因果图导出的判定表

(考虑结果不能同时发生，所以对其施加唯一约束O。原因5、6、7不能同时发生，所以对其施加异约束E.)

三、根据因果图建立判定表：(分为两表)

注：1、以上判定表中由于表格大小限制没有列出最后所选的测试用例；2、第2表中部分列被合并表示不可能发生的现象；3、通过中间节点将用例的判定表简化为两个小表。减少工作量。

四、根据判定表写测试用例表（略）

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
原因	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
结果	11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	21	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
用例																	

		1	2	3	4	5	6	7	8	9	`0	11	12	13	14	15	16
原因	11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
结果	22	0		0	1	0	0			0	0						
	23	0		0	0	0	1			0	0						
	24	0		0	0	0	0			0	1						
用例																	

2. 某程序规定："输入三个整数 a 、 b 、 c 分别作为三边的边长构成三角形。通过程序判定所构成的三角形的类型，当此三角形为一般三角形、等腰三角形及等边三角形时，分别作计算 … "。用等价类划分方法为该程序进行测试用例设计。（三角形问题的复杂之处在于输入与输出之间的关系比较复杂。）

分析题目中给出和隐含的对输入条件的要求：

（1）整数　（2）三个数　（3）非零数　（4）正数

（5）两边之和大于第三边　（6）等腰　（7）等边

如果 a 、 b 、 c 满足条件（ 1 ） ~ （ 4 ），则输出下列四种情况之一：

1)如果不满足条件（5），则程序输出为 " 非三角形 " 。

2)如果三条边相等即满足条件（7），则程序输出为 " 等边三角形 " 。

3)如果只有两条边相等、即满足条件（6），则程序输出为 " 等腰三角形 " 。

4)如果三条边都不相等，则程序输出为 " 一般三角形 " 。

列出等价类表并编号

覆盖有效等价类的测试用例：

a	b	c	覆盖等价类号码
3	4	5	(1) -- (7)
4	4	5	(1) -- (7) , (8)
4	5	5	(1) -- (7) , (9)
5	4	5	(1) -- (7) , (10)
4	4	4	(1) -- (7) , (11)

		有效等价类型	号码	无效等价类	号码
输入条件	输入三个整数	整数	1	一边为非整数：a为非整数	12
				一边为非整数：b为非整数	13
				一边为非整数：c为非整数	14
				两边为非整数：a,b为非整数	15
				两边为非整数：b,c为非整数	16
				两边为非整数：a,c为非整数	17
				三边a，b，c均为非整数	18
		三个数	2	只给一边：只给a	19
				只给一边：只给b	20
				只给一边：只给c	21
				只给两边：只给ab	22
				只给两边：只给b,c	23
				只给两边：只给ac	24
				给出三个以上	25
		非零数	3	一边为零：a为0	26
				一边为零：b为0	27
				一边为零：c为0	28
				二边为零：a,b为0	29
				二边为零：b,c为0	30
				二边为零：a,c为0	31
				三边a,b,c均为0	32
		正数	4	一边<0：a<0	33
				一边<0：b<0	34
				一边<0：c<0	35
				二边<0：a<0且b<0	36
				二边<0：a<0且c<0	37
				二边<0：b<0且c<0	38
				三边均<0：a<0且b<0且c<0	39
输出条件	构成一般三角形	a+b>c	5	a+b<0	40
				a+b=0	41
		b+c>a	6	b+c<a	42
				b+c=a	43
		a+c>b	7	a+c<b	44
				a+c=b	45
	构成等腰三角形	a=b且b=c且a=c且两边之和大于第三边	8 9 10		
	构成等腰三角形	a=b=c	11		

覆盖无效等价类的测试用例：（下页）

3. 设有一个档案管理系统，要求用户输入以年月表示的日期。假设日期限定在1990年1月~2049年12月，并规定日期由6位数字字符组成，前4位表示年，后2位表示月。现用等价类划分法设计测试用例，来测试程序的"日期检查功能"。（不考虑2月的问题）

1)划分等价类并编号,下表等价类划分的结果

输入等价类	有效等价类	无效等价类
日期的类型及长度	①6位数字字符	②有非数字字符③少于6位数字字符④多于6位数字字符
年份范围	⑤在1990~2049之间	⑥小于1990⑦大于2049
月份范围	⑧在01~12之间	⑨等于00⑩大于12

a	b	c	覆盖等价类号码	a	b	c	覆盖等价类号码
2.5	4	5	12	0	0	5	29
3	4.5	5	13	3	0	0	30
3	4	5.5	14	0	4	0	31
3.5	4.5	5	15	0	0	0	32
3	4.5	5.5	16	-3	4	5	33
3.5	4	5.5	17	3	-4	5	34
4.5	4.5	5.5	18	3	4	-5	35
3			19	-3	-4	5	36
	4		20	-3	4	-5	37
		5	21	3	-4	-5	38
3	4		22	-3	-4	-5	39
	4	5	23	3	1	5	40
3		5	24	3	2	5	41
3	4	5	25	3	1	1	42
0	4	5	26	3	2	1	43
3	0	5	27	1	4	2	44
3	4	0	28	3	4	1	45

2)设计测试用例，以便覆盖所有的有效等价类在表中列出了3个有效等价类，编号分别为①、⑤、⑧，设计的测试用例如下：

测试数据　期望结果　　覆盖的有效等价类

200211　　输入有效　　①、⑤、⑧

3)为每一个无效等价类设计一个测试用例，设计结果如下：

测试数据	期望结果	覆盖的无效等价类
95June	无效输入	②
20036	无效输入	③
2001006	无效输入	④
198912	无效输入	⑥
200401	无效输入	⑦
200100	无效输入	⑨
200113	无效输入	⑩

4. NextDate 函数包含三个变量：month 、 day 和 year ， 函数的输出为输入日期后一天的日期。 例如，输入为 2006年3月 7日，则函数的输出为 2006年3月8日 。要求输入变量 month 、 day 和 year 均为整数值，并且满足下列条件：

①1≤month≤12

②1≤day≤31

③1920≤year≤2050

1)有效等价类为：

M1＝{月份：1≤月份≤12}

D1＝{日期：1≤日期≤31}

Y1＝{年：1812≤年≤2012}

2)若条件 ① ~ ③中任何一个条件失效，则 NextDate 函数都会产生一个输出，指明相应的变量超出取值范围，比如 "month 的值不在 1–12 范围当中 " 。显然还存在着大量的 year 、 month 、 day 的无效组合，

NextDate 函数将这些组合作统一的输出：＂无效输入日期＂。其无效等价类为：

M2＝{月份：月份<1}

M3＝{月份：月份>12}

D2＝{日期：日期<1}

D3＝{日期：日期>31}

Y2＝{年：年<1812}

Y3＝{年：年>2012}

弱一般等价类测试用例

月份	日期	年	预期输出
6	15	1912	1912年6月16日

强一般等价类测试用例同弱一般等价类测试用例

注：弱--有单缺陷假设；健壮--考虑了无效值

(一)弱健壮等价类测试

用例ID	月份	日期	年	预期输出
WR1	6	15	1912	1912年6月16日
WR2	–1	15	1912	月份不在1～12中
WR3	13	15	1912	月份不在1～12中
WR4	6	–1	1912	日期不在1～31中
WR5	6	32	1912	日期不在1～31中
WR6	6	15	1811	年份不在1812～2012中
WR7	6	15	2013	年份不在1812～2012中

(二)强健壮等价类测试

用例ID	月份	日期	年	预期输出
SR1	–1	15	1912	月份不在1～12中
SR2	6	–1	1912	日期不在1～31中
SR3	6	15	1811	年份不在1812～2012中
SR4	–1	–1	1912	两个无效一个有效
SR5	6	–1	1811	两个无效一个有效
SR6	–1	15	1811	两个无效一个有效
SR7	–1	–1	1811	三个无效

5. 佣金问题等价类测试用例，它是根据佣金函数的输出值域定义等价类，来改进测试用例集合。

输出销售额≤1000元　佣金10%

1000<销售额≤1800　佣金=100+(销售额–1000)*15%

销售额>1800　佣金=220+(销售额–1800)*20%

测试用例	枪机(45)	枪托(30)	枪管(25)	销售额	佣金
1	5	5	5	500	50
2	15	15	15	1500	175
3	25	25	25	2500	360

根据输出域选择输入值，使落在输出域等价类内，可以结合弱健壮测试用例结合。

小题狂练

1. 下列哪个是瀑布型开发的正确过程？（）

A. 设计 –〉编码 –〉单元测试 –〉集成 –〉系统测试

B. 需求分析 –〉设计 –〉编码&单元测试 –〉集成 –〉系统测试
C. 需求分析 –〉设计 –〉编码&单元测试 –〉系统测试 –〉集成
D. 需求分析 –〉设计 –〉单元测试 –〉系统测试 –〉集成

3. 持续的质量验证应该做到： ()
A. 尽早的质量验证 B. 在项目后期验证质量 C. 只需验证可执行系统的工程

4 测试的基本流程:()
1.开发人员将开放出来的产品交给测试部门。
2.测试人员使用某种测试方法测试产品并收集产品的缺陷。
3.与开发人员沟通被发现的缺陷。
4.开发人员修复缺陷并送回到测试部门重新测试。
A. 1,2,3,4 B. 2.3.1.4 C.1,3,2,4 D.2,1,3,4

5.语句覆盖方法的正确步骤：()
1 程序结构化：将程序转化为流图
2 环形复杂度计算：得到的是要覆盖所有语句，独立路径数量的上限 N
3 构造独立路径：构造 N 条独立路径
4 确认分支点：针对每条独立路径，确认独立路径上的所有分支点
5 构造执行条件：根据一条独立路径上的所有分支点集合确认路径执行条件集合
6 编制用例：对这些路径和路径执行条件集合进行分析，编制用例
A. 1,2,3,4,5,6 B. 1,2,4,3,5,6 C. 1,3,2,4,5,6 D. 1,2,3,5,4,6

6.下面的观点哪些是错误的？ ()
A.你永远也不可能完成测试，这个重担将会简单地从你(或者开发人员)身上转移到你的客户身上
B. 当你时间不够或者资金不够用的时候，就完成了测试
C. 不能绝对地认定软件永远也不会再出错 ，但可以根据合理和有效的统计模型来判定软件的测试程度

7.单元测试是有谁来完成？ ()
A. 开发人员 B.测试人员 C.客户 D.项目经理

10.下面哪个不是测试用例设计基本原则是： ()
A.测试用例能够发现至今没有发现的错误
B.测试用例应由测试输入数据和与之对应的预期输出结果这两部分组成
C.在测试用例设计时，应当包含合理的输入条件和不合理的输入条件
D.测试用例设计应该以功能为线索

11.在语句覆盖、判断覆盖、条件覆盖、判断－条件覆盖 、 条件组合覆盖和路径覆盖中用例设计难度最大的是：( )
A.语句覆盖 B.判断覆盖 C.条件覆盖 D.判断－条件覆盖 E. 条件组合覆盖 F.路径覆盖

12.在语句覆盖、判断覆盖、条件覆盖、判断－条件覆盖 、 条件组合覆盖和路径覆盖中用例覆盖程度最高的是：( )
A.语句覆盖B.判断覆盖 C.条件覆盖 D.判断－条件覆盖 E. 条件组合覆盖 F.路径覆盖

13.下面哪些不是排错时应该采用的方法策略？ ()

断点设置 可疑变量查看 SQL 语句执行检查 注意群集现象 为代码添加注释

14.下面哪些是用于Java程序单元测试的工具？ ()
A. Eclipse B.JUnit C.NetBean D.Struts

15.JUnit 属于哪类工具？ ()
A. 开放源码的工具 B.商业工具 C.功能测试工具 D.性能测试工具

16.在JUnit中所有测试用例类的父类是： ()
A.Test B.TestCase C.TestSuite D.TestManager

17.在JUnit中所有测试套件类的父类是： ()
A.Test B.TestCase C.TestSuite D.TestManager

18.在JUnit中如果你要测试多个类，你应该使用： ()
A.Test B.TestCase C.TestSuite D.TestManager

19.在JUnit中如果你要测试单个类，你应该使用： ()
A.Test B.TestCase C.TestSuite D.TestManager

20.性能测试的好处包括：()
A. 从用户的角度改进质量 B.减少变更的成本 C.增加利润 D.加快项目进度

21.负载分析的目标是： ()
A.确保被执行的测试代表了真实的用户活动
B.确保系统功能被实现
C.确保客户学会使用系统
D.确保系统没有逻辑错误

26.客户端兼容性测试不包括： ()
A. 视频设置 B.Modem/连接速率 C. 打印机 D.表单测试

多选题
1. 下列哪些叙述属于瀑布型开发的特点？ ()
A. 推迟关键风险决定的确认 B. 有利于时间进度和工作完成情况的预估
C. 推迟和集中的继承与测试 D. 排斥早期的部署

2. 在迭代开发中，每一个迭代都会产生一个可执行的版本。每个迭代都包括集成和测试。迭代可以为下列哪些方面带来帮助？ ()
A. 在投入大的成本之前解决主要的风险 B. 使早期的客观反馈变成可能
C. 进行持续的测试和集成 D. 关注项目长期的目标里程碑上

3. 为了帮助管理需求与源于这些需求的测试之间的关系，你可以在这些元素之间构建可跟踪的关系。可跟踪性可以帮助你做到下列哪些事情？ ()
A. 评估一个需求的变更对项目的影响 B. 评估在需求上测试失败
C. 提前交付项目 D. 验证所有系统需求完全被实现了 E. 管理变更 F. 管理项目范围

5. 基于组件的架构的目的()
A.促进软件的重用 B. 项目管理的基础 C. 管理复杂性 D.维护完整性

8.为什么要测试？ ()
A. 以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷
B. 实施测试收集到的测试结果数据为可靠性分析提供了依据
C. 它只能说明软件中存在错误 D. 保证软件开发团队的利益

9. 软件质量缺陷的原因:()
A. 缺乏或者没有进行沟通 B. 软件复杂度 C. 编程错误 D. 客户操作错误

10.下面哪些是软件测试的原则？ ()
A.尽早并持续的测试 B. 避免自检 C.严格执行测试计划 D.妥善保存测试产物

11.可测试性包括： ()
A.可操作性 B.可观察性 C.可控制性 D.可分解性 E. 可扩展行性

12.一个好的测试的属性包括： ()
A. 一个好的测试发现错误的可能性很高 B.一个好的测试并不冗余
C. 一个好的测试应该很复杂 D.一个好的测试应该比较简单

13.测试显示了哪些问题?()
A. 错误 B. 与需求的不一致 C. 性能问题 D.质量的迹象

14.测试是哪种角色的职责:()
A. 项目经理 B.客户 C. 测试人员 D.开发人员

15.白盒测试的方法有哪些？ ()
A. 语句覆盖方法 B. 分支覆盖 C. 逻辑覆盖 D. 循环测试

16.黑盒测试的目的是?()
A. 功能不对或遗漏 B. 界面错误 C. 数据结构或外部数据库访问错误 D.性能错误 E.代码错误

17.软件测试的特征是:()
A. 测试开始于模块层，然后“延伸”到整个基于计算机的系统集合中
B.不同的测试技术适用于不同的时间点
C.测试是由软件的开发人员和独立的测试组来管理的
D.测试和调试是不同的活动，但是调试必须能够适应任何的测试策略

18.集成测试的方法有哪些？ ()
A. 自顶向下集成 B.自底向上集成 C.随机集成 D.分对集成

19.确认测试的标准是什么？ ()
A.所有的功能需求都得到了满足 B.所有性能需求都达到了
C.文档是正确且合理的 D.项目经理的要求达到了

20.系统测试的种类有： ()
A. 恢复测试 B.安全测试 C.压力测试 D.性能测试

22.面向对象软件开发的特点： ()
A. 模型从需求开始被开发
B. 模型逐步演化为详细的类模型、类连接和关系、系统设计和分配、以及对象设计
C. 测试的目标是在错误传播到下一次递进前发现错误
D. 以功能进行模块划分

23.合格代码的特点是：()
A. 正确性 B.清晰性 C. 规范性 D. 一致性 E.高效性

24.单元测试分为： ()
A.人工静态检查 B.动态执行跟踪 C.压力测试

25.人工检查阶段的活动包括： ()
A.检查算法的逻辑正确性 B.模块接口的正确性检查
C. 输入参数有没有作正确性检查 D.调用其他方法接口的正确性

26.动态执行测试通常分为 ()
A.黑盒测试 B.白盒测试 C. 集成测试 D.系统测试

27.单元白盒测试的任务包括： ()
A. 对模块内所有独立的执行路径至少测试一次
B.对所有的逻辑判定，取“真”与“假”的两种情况都至少执行一次
C. 在循环的边界和运行界限内执行循环体
D.测试内部数据的有效性

28.单元白盒测试的步骤包括： ()
A.设计测试用例 B.设计测试类模块 C. 跟踪调试 D.测试功能

29.在JUnit中运行测试的方式可以使用： ()
A.junit.textui.TestRunner B.junit.awtui.TestRunner
C.junit.swingui.TestRunner D.直接使用 Java 命令运行测试类

30.为什么进行性能测试? ()
A.优化性能 B.最小化成本 C. 最小化风险 D.满足客户功能上的要求

31.基于硬件的测试的特点是： ()
A. 需要很多台实际的计算机 B. 需要一名测试人员 C.需要多名测试人员 D. 需要一台实际的计算机
注释：至少一台物理计算机模拟很多个用户

参考答案：单选BAAA BAD EFEBA BCCBA A C 多选1.ACD 8.AB 9.ABC 10.ACD 12.AB 13.ABC 14.ACD 15.ABCD 18.ABD 20.BCD 23.ABCDE 24.AB 26.AB 30.AC