1. 试述软件的概念和特点？软件复用的含义？构件包括哪些？
2. 瀑布模型和螺旋模型的主要区别是什么？
3. 软件生存周期及其模型是什么？
4. 什么是软件测试？软件测试的目的与原则
5. 净室软件工程的策略是什么？
6. 软件配置管理的作用？软件配置包括什么？
7. 什么是软件质量？软件包是什么？
8. 目前主要的测试用例设计方法是什么？
9. 软件的安全性应从哪几个方面去测试？

参考答案：

1. 答案如下：
   1. 软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、文档的完整集合。
   2. 软件复用(Software Reuse)是将已有软件的各种有关知识用于建立新的软件,以缩减软件开发和维护的花费。软件复用是提高软件生产力和质量的一种重要技术。早期的软件复用主要是代码级复用，被复用的知识专指程序，后来扩大到包括领域知识、开发经验、设计决定、体系结构、需求、设计、代码和文档等一切有关方面。
   3. 可以被复用的软件成分一般称作可复用构件
2. 答案如下：
   1. 参照TP书上第六章45/46页的讲解，参考一下书上的说法进行对比即可。考虑弹性、风险、成本，等几个方面。
3. 答案如下：
   1. 软件生存周期是软件开发全部过程、活动和任务的结构框架，是从可行性研究到需求分析、软件设计、编码、测试、软件发布维护的过程。
   2. 在经历需求、分析、设计、实现、部署后，软件将被使用并进入维护阶段，直到最后由于缺少维护费用而逐渐消亡。这样的一个过程，称为“生命周期模型“（Life Cycle Model）。
4. 答案如下：
   1. 使用人工或自动手段，来运行或测试某个系统的过程。其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。
   2. 软件测试的目的：
      1. 测试是程序的执行过程，目的在于发现错误
      2. 一个成功的测试用例在于发现至今未发现的错误
      3. 一个成功的测试是发现了至今未发现的错误的测试
      4. 确保产品完成了它所承诺或公布的功能，并且用户可以访问到的功能都有明确的书面说明。
      5. 确保产品满足性能和效率的要求
      6. 确保产品是健壮的和适应用户环境的
   3. 软件测试的原则：

教材的说法：

* + 1. 软件测试应尽早执行，并贯穿于整个软件生命周期
    2. 软件测试应追溯需求
    3. 测试应由第三方来构造
    4. 穷举测试是不可能的,要遵循Good-enough原则
    5. 必须确定预期输出（或结果）
    6. 必须彻底检查每个测试结果
    7. 充分注意测试中的群集现象
    8. 缺陷的二八定理
    9. 严格执行测试计划，排除测试的随意性
    10. 注意合法合理的输入，也要注意非法的非预期的输入
    11. 检查程序是否是否做了不该做的
    12. 测试应从”小规模”开始，逐步转向”大规模”
    13. 反复使用同样的测试会使软件具有抵抗力
    14. 关注缺陷的修复

另一种说法：

* + 1. 应当把”尽早和不断地测试”作为开发者的座右铭。
    2. 程序员应该避免检查自己的程序，测试工作应该由独立的专业的软件测试机构来完成。
    3. 设计测试用例时，应该考虑到合法的输入和不合法的输入，以及各种边界条件，特殊情况下要制造极端状态和意外状态，比如网络异常中断、电源断电等情况。
    4. 一定要注意测试中的错误集中发生现象，这和程序员的编程水平和习惯有很大的关系。
    5. 对测试错误结果一定要有一个确认的过程。一般有A测试出来的错误，一定要有一个B来确认，严重的错误可以召开评审会进行讨论和分析。
    6. 制定严格的测试计划，并把测试时间安排得尽量宽松，不要希望在极短的时间内完成一个高水平的测试。
    7. 回归测试的关联性一定要引起充分的注意，修改一个错误而引起更多错误出现的现象并不少见。
    8. 妥善保存一切测试过程文档，意义是不言而喻的，测试的重现性往往要靠测试文档。

1. 答案如下：
   1. 增量计划。开发一个采用增量策略的项目计划，建立每个增量的功能、它的项目大小、以及净室开发进度表。必须特别小心以保证通过认证的增量将被定时集成。
   2. 需求收集。使用类似于在第11 章引入的技术，为每个增量开发一个客户级需求的更详细的描述。
   3. 盒结构规约。使用一个运用盒结构的规约方法［HEV93］来描述功能规约。遵从操作分析原则，盒结构”在每一个精化级别上分离和分开行为、数据及过程的创造性定义”。
   4. 形式化设计。使用盒结构方法，净室设计是规约的自然的无缝的扩展。虽然，在两个活动间可进行清楚的区分，但是，规约(称为”黑盒”)是被递进地求精(在一个增量内)以成为类似于体系结构的和过程的设计(分别称为”状态盒”和”清晰盒”)。
   5. 正确性验证。净室小组对设计及代码进行一系列严格的正确性验证活动。验证从最高层次的盒结构(规约)开始，然后移向设计细节和代码。正确性验证的第一层次通过应用一组”正确性问题”［LIN88］来进行，如果这没有证明规约是正确的，则使用更形式化的(数过学的)验证方法。
   6. 代码生成、检查和验证。以某种专门语言表示的盒结构规约被转换为合适的程序设计语言。然后，使用标准的走查或检查技术(第8 章)来保证代码和盒结构的语义相符性，以及代码的语法正确性。然后，对源代码进行正确性验证。
   7. 统计性测试计划。分析软件的项目级使用情况，计划和设计一组执行用途的”概率分布”的测试用例(25.4 节)。如图25－1 所示，这个净室活动是和规约、验证及代码生成并行进行的。
   8. 统计性使用测试。记住，对计算机软件进行彻底测试是不可能的，因此，总需要设计有限数量的测试用例。统计性使用技术［POO88］执行一系列由特定对象的所有用户的所有可能的程序执行的统计样本(上面提到的概率分布)所导出的测试。认证。一旦完成验证、检查和使用测试(并且所有错误被修正)，则开始进行增量集成前的认证工作。
2. 答案如下：
   1. 软件配置管理作为软件开发过程的必要环节和软件开发管理的基础，贯穿整个软件生命周期，同时对软件开发过程的宏观管理即项目管理也有重要的支持作用。一个软件开发组织真正有效的实施软件配置管理，将会使软件开发过程有更好的可预测性，使系统具有可重复性，大大提高软件组织的竞争力。
   2. 软件配置包括如下内容：
      1. 配置项识别
      2. 工作空间管理
      3. 版本控制
      4. 变更控制
      5. 状态报告
      6. 配置审计
3. 答案如下：
   1. 简单的说：软件质量：软件产品的特性可以满足用户的功能、性能需求的能力。  
      比较长的说法：  
      现代质量管理认为，质量是客户要求或者期望的有关产品或者服务的一组特性，落实到软件上，这些特性可以是软件的功能、性能和安全性等等。这些特性决定了软件产品保证客户满意的能力，并且，这些特性应该是可以度量的。  
      我们还可以从另一个角度，即软件产品是如何生产出来的，来间接的推断软件质量。我们称之为软件的流程质量，以有别于前面所说的软件产品质量。所谓流程，我们可以将其理解为一个活动序列和与此相关的输入、输出、约束条件、实现方法、辅助工具等等因素共同组成的系统。ISO9001 和SW-CMM 都主要是从流程角度来探讨软件质量和质量改进的。   
        
      当然，我们还能从其它角度，比如软件的生产者-人的素质，来诠释软件质量，但不管怎样，软件的产品质量是最终的检验标准，而最终的检验者就是客户。从这个意义上说，软件质量就是客户满意度。
   2. 软件包(Software Package)是指具有特定的功能，用来完成特定任务的一个程序或一组程序。可分为应用软件包和系统软件包两大类。应用软件包与特定的应用领域有关，又可分为通用包及专用包两类。通用软件包根据社会的一些共同需求开发，专用软件包则是生产者根据用户的具体需求定制的，可以为适合其特殊需要进行修改或变更。
4. 答案如下：
   1. 白盒测试：
      1. 逻辑覆盖
      2. 循环覆盖
      3. 基本路径覆盖
   2. 黑盒测试：
      1. 边界值分析法
      2. 等价类划分
      3. 错误猜测法
      4. 因果图法
      5. 状态图法
      6. 测试大纲法
      7. 随机测试
      8. 场景法
5. 答案如下：

软件安全性测试包括程序、数据库安全性测试。根据系统安全指标不同测试策略也不同。

* 1. 用户认证安全的测试要考虑问题：
     1. 明确区分系统中不同用户权限
     2. 系统中会不会出现用户冲突
     3. 系统会不会因用户的权限的改变造成混乱
     4. 用户登陆密码是否是可见、可复制
     5. 是否可以通过绝对途径登陆系统（拷贝用户登陆后的链接直接进入系统）
     6. 用户退出系统后是否删除了所有鉴权标记，是否可以使用后退键而不通过输入口令进入系统
  2. 系统网络安全的测试要考虑问题
     1. 测试采取的防护措施是否正确装配好，有关系统的补丁是否打上
     2. 模拟非授权攻击，看防护系统是否坚固
     3. 采用成熟的网络漏洞检查工具检查系统相关漏洞（即用最专业的黑客攻击工具攻击试一下，现在最常用的是 NBSI 系列和 IPhacker IP ）
     4. 采用各种木马检查工具检查系统木马情况
     5. 采用各种防外挂工具检查系统各组程序的外挂漏洞
  3. 数据库安全考虑问题：
     1. 系统数据是否机密（比如对银行系统，这一点就特别重要，一般的网站就没有太高要求）
     2. 系统数据的完整性（我刚刚结束的企业实名核查服务系统中就曾存在数据的不完整，对于这个系统的功能实现有了障碍）
     3. 系统数据可管理性
     4. 系统数据的独立性
     5. 系统数据可备份和恢复能力（数据备份是否完整，可否恢复，恢复是否可以完整）

1. 什么是测试用例 什么是测试脚本 两者的关系是什么？
2. 简述什么是静态测试、动态测试、黑盒测试、白盒测试、α测试  
   β测试
3. 软件质量保证体系是什么 国家标准中与质量保证管理相关的几个标准是什么?他们的编号和全称是什么?
4. 软件产品质量特性是什么?
5. 软件测试的原则与策略是什么?
6. 结构化系统测试和功能性系统测试分别采用了哪些方法和技术?
7. 软件测试分为几个阶段 各阶段的测试策略和要求是什么?
8. 面向对象的测试用例设计有几种方法？如何实现?
9. 在软件测试各个阶段通常完成什么工作？各个阶段的结果文件是什么？包括什么内容？

参考答案：

1. 答案如下：
   1. 为实施测试而向被测试系统提供的输入数据、操作或各种环境设置以及期望结果的一个特定的集合。
   2. 测试脚本是为了进行自动化测试而编写的脚本。
   3. 测试脚本的编写必须对应相应的测试用例，
2. 答案如下：
   1. 静态测试是不运行程序本身而寻找程序代码中可能存在的错误或评估程序代码的过程。
   2. 动态测试是实际运行被测程序，输入相应的测试实例，检查运行结果与预期结果的差异，判定执行结果是否符合要求，从而检验程序的正确性、可靠性和有效性，并分析系统运行效率和健壮性等性能。
   3. 黑盒测试一般用来确认软件功能的正确性和可操作性,目的是检测软件的各个功能是否能得以实现,把被测试的程序当作一个黑盒,不考虑其内部结构,在知道该程序的输入和输出之间的关系或程序功能的情况下,依靠软件规格说明书来确定测试用例和推断测试结果的正确性。
   4. 白盒测试根据软件内部的逻辑结构分析来进行测试,是基于代码的测试，测试人员通过阅读程序代码或者通过使用开发工具中的单步调试来判断软件的质量，一般黑盒测试由项目经理在程序员开发中来实现。
   5. α测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的受控测试，Alpha测试不能由程序员或测试员完成。
   6. β测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，Beta测试不能由程序员或测试员完成。
3. 答案如下：
   1. 来自Wikipedia对SQA的定义，软件质量保证（SQA）：

Software Quality Assurance (SQA) consists of the software engineering processes and methods used to ensure quality. SQA encompasses the entire software development process, which may include processes such as reviewing requirements documents, source code control, code reviews, change management, configuration management, release management and of course, software testing.

SQA由一套软件工程过程和方法组成，以保证（软件的）质量。SQA贯穿整个软件开发过程，(它)应包括需求文档评审、代码控制、代码评审、变更管理、配置管理、版本管理和软件测试。

1. 国家标准：
   1. GB/T 8567-2006 计算机软件文档编制规范
   2. GB/T 11457-2006 信息技术 软件工程术语
   3. GB/T 16260.1-2006 软件工程 产品质量 第1部分：质量模型
   4. GB/T 16260.2-2006 软件工程 产品质量 第2部分：外部度量
   5. GB/T 16260.3-2006 软件工程 产品质量 第3部分：内部度量
   6. GB/T 16260.4-2006 软件工程 产品质量 第4部分：使用质量的度量
   7. GB/Z 20156-2006 软件工程 软件生成周期过程 用于项目管理的指南
   8. GB/T 20157-2006 信息技术 软件维护
   9. GB/T 20158-2006 信息技术 软件生成周期过程 配置管理
2. 答案如下：
   1. 功能性：适应性、准确性、互操作性、依从性、安全性。
   2. 可靠性：成熟性、容错性、以恢复性。
   3. 可使用性：易理解性、易学习性、易操作性。
   4. 效率：时间特性、资源特性。
   5. 可维护性：易分析性、易变更性、稳定性、易测试性。
   6. 可移植性： 适应性、易安装性、遵循性、易替换性。
3. 答案如下：
   1. 软件测试的原则：

教材的说法：

1. 软件测试应尽早执行，并贯穿于整个软件生命周期
2. 软件测试应追溯需求
3. 测试应由第三方来构造
4. 穷举测试是不可能的,要遵循Good-enough原则
5. 必须确定预期输出（或结果）
6. 必须彻底检查每个测试结果
7. 充分注意测试中的群集现象
8. 缺陷的二八定理
9. 严格执行测试计划，排除测试的随意性
10. 注意合法合理的输入，也要注意非法的非预期的输入
11. 检查程序是否是否做了不该做的
12. 测试应从”小规模”开始，逐步转向”大规模”
13. 反复使用同样的测试会使软件具有抵抗力
14. 关注缺陷的修复

另一种说法：

1. 应当把”尽早和不断地测试”作为开发者的座右铭。
2. 程序员应该避免检查自己的程序，测试工作应该由独立的专业的软件测试机构来完成。
3. 设计测试用例时，应该考虑到合法的输入和不合法的输入，以及各种边界条件，特殊情况下要制造极端状态和意外状态，比如网络异常中断、电源断电等情况。
4. 一定要注意测试中的错误集中发生现象，这和程序员的编程水平和习惯有很大的关系。
5. 对测试错误结果一定要有一个确认的过程。一般有A测试出来的错误，一定要有一个B来确认，严重的错误可以召开评审会进行讨论和分析。
6. 制定严格的测试计划，并把测试时间安排得尽量宽松，不要希望在极短的时间内完成一个高水平的测试。
7. 回归测试的关联性一定要引起充分的注意，修改一个错误而引起更多错误出现的现象并不少见。
8. 妥善保存一切测试过程文档，意义是不言而喻的，测试的重现性往往要靠测试文档。
9. 软件测试策略：在一定的软件测试标准、测试规范的指导下，依据测试项目的特定环境约束而规定的软件测试的原则、方式、方法的集合。
10. 答案如下：
    1. 结构化系统测试技术：用于验证所开发的系统及程序的运行情况。目标是要确保产品设计在结构上合理，功能上正确。为确定实现的配置及其各功能共同作用以完成特定任务提供了一种机制。结构化测试技术由以下几种：
       1. 1）压力测试：确定系统以期望的容量执行。  
          压力测试技术用于检查系统面对意外情况下的大数据量时是否可以正常运行。所涉及的方面包括输入事务、内部表、磁盘空间、输出、通信、计算机容量以及人机交互等。  
          当应用系统所能正常处理的工作量并不确定时需要使用压力测试。压力测试意图通过对系统施加超负载事务量来达到破坏系统的目的。弱点在于准备测试的时间与在测试的实际执行过程中所消耗的资源数量都非常之大，通常在应用程序投入使用之前这种技术是无法进行的。
       2. 执行测试：系统能达到期望的熟练性。  
          举例：事务轮转时间充分；软硬件使用良好。  
          执行测试技术用于检查系统是否达到了预期在产品状态下的成熟度。执行测试可以验证系统的响应时间、轮转时间及设计性能。  
          在开发过程的早期就应该进行执行测试，尽早制定已经完成的系统没有达到性能指标是非常有价值的。在关键时间点进行。关键时间点指的是当前的结果会影响甚至改变系统结构的时间点。
       3. 恢复测试：系统失效之后可以恢复到可操作状态。  
          举例：引入失败；评估备份数据的充分性。  
          恢复测试技术用于确保系统在经历灾难后可以继续正常运行，它不仅可以验证恢复过程，而且可以验证过程各组件的有效性。  
          当用户认为系统操作的连续性对于其所涉及领域的某些功能至关重要时，需要进行恢复测试。
       4. 操作测试：系统以正常操作状态执行。  
          举例：确定系统可以依据文档进行运行；JCL（工作控制语言）充分。  
          操作测试技术主要用于检查系统在正常的操作状态下是否可以执行。操作测试可以与其它测试联合执行。  
          任何应用程序在成为产品之前都应进行操作测试。
       5. （与过程的）一致性测试：系统的开发与标准和规程相一致。  
          举例：按标准执行；文档完整。  
          一致性测试技术用于验证应用程序的开发是否与信息技术指标、过程及准则相一致。一致性测试最有效的方法是过程审查。  
          系统开发标准和过程的一致性程度依赖于管理层对于所需遵循的特定过程和执行标准的重视程度。
       6. 安全性测试：根据组织的重要性对系统进行保护。  
          举例：访问拒绝；规程适当。  
          安全性测试技术用于评价保护性程序及安全对策的充分性。安全性缺陷不如其它类型的缺陷那么明显。安全性测试是测试过程中高度专业化的部分。分物理安全性（针对利用物理方法收集信息的手段）和逻辑安全性（针对使用计算机处理和通信能力进行非法活动信息的手段）。  
          当系统保护信息和资产对于组织来说意义重大时，需要进行安全性测试。
    2. 功能性系统测试用于确保系统需求与定义都得到了满足。该过程通常包含创建用于评价应用程序正确性的测试条件。用于执行功能测试的几种测试技术包括：
       1. 需求测试：系统按制定方式执行。  
          举例：证明系统需求；与政策、规则相一致。  
          需求测试技术验证系统是否正确执行其功能，并且能保证在相当长的一段时间内保持其正确性。需求测试的执行主要通过执行创建的测试条件以及功能检查单来完成，通过需求得到测试条件，然后以类似于SDLC这种特定的方式表现，生成用于评价实现的应用系统的测试数据。  
          任何应用程序都应该对需求进行测试，此过程应该开始于需求阶段，并一直持续到系统运行和维护阶段。
       2. 回归测试：验证系统中没有改变的部分仍能正确运行。  
          举例：未变更的部分正常运行；未变更的人工规程正确。  
          回归测试技术对已经测试过的部分进行重新测试，以保证它们在应用程序其它部分发生变更之后仍能正常运行。  
          当变更会对应用程序中没有变更的部分产生高风险的影响时需要进行回归测试。
       3. 错误处理测试：错误可以得到防止或检测，并被修复。  
          举例：将错误引入测试；错误的再次注入。  
          人工系统与自动系统之间差别的特点之一就是预定义的错误处理特性。错误处理测试技术用于检查应用系统正确处理发生异常的能力。错误处理测试需要一组知识丰富的人员来预见应用系统可能发生的错误。它是测试错误的引入、错误的处理，控制条件以及条件的再次正确输入。  
          在系统整个生命周期中都应该进行错误测试。在开发过程中，应该识别错误带来的问题并且采取相应的措施将错误减少到可以接受的程度。
       4. 人工支持测试：人机交互有效。  
          举例：具备人工规程；人员接受过培训。  
          人工支持测试技术主要包括人员在准备数据以及使用来源于自动程序数据的过程中执行所有功能。  
          在生命周期的全过程都应该验证人工系统功能的正确性。
       5. 系统间测试：数据可以正确地在系统间传递。  
          举例：系统间参数变化；系统间文档更新。  
          系统间测试技术用于保证应用程序间相互管理的正确性。系统间测试的一个最好的工具是集成测试工具，它允许在产品环境下进行测试，可以以最小的代价测试系统间的耦合性。  
          在应用系统间的参数发生变更时需要进行系统间的测试。测试的程度和类型依赖于与出错的参数相关联的风险情况。
       6. 控制测试：将系统风险控制降低到可以接受的级别。  
          举例：文件一致性规程正常；人工控制正确。  
          控制测试技术包括数据确认、文件完整性控制、评审追踪、备份和恢复、文档，以及与系统完整性相关的其它方面。主要用于确保对系统特定功能的检查。可以用于控制测试的一个方法是生成风险矩阵。  
          控制测试是系统测试中的一个完整的部分，占测试时间的很大比例。
       7. 平行测试：发现原系统与新系统之间的意外差异。  
          举例：原系统与新系统一致；原系统仍然可以工作。  
          平行测试技术用于检查新应用程序的结果是否与原来的应用程序或者上一版本应用程序的处理相一致。它执行冗余处理以保证新版本或者新应用程序执行的正确性；给出同一应用程序不同版本之间一致的和不一致的地方。平行测试可以对整个应用程序进行，也可对应用程序的一部分进行。  
          当不能确定新应用程序处理的正确性，或者当新旧版本的应用程序非常类似时，需要进行平行测试。
11. 答案如下：
    1. 软件测试按阶段划分可以分为单元测试、集成测试、系统测试和<验收测试>（不一定有）几个阶段
    2. 单元测试测试策略：
       1. 自顶向下的单元测试策略  
          方法：先对最顶层的基本单元进行测试，把所有调用的单元做成桩模块。然后再对第二层的基本单元进行测试，使用上面已测试的单元做驱动模块。依此类推直到测试完所有基本单元。  
          优点：在集成测试前提供早期的集成途径。在执行上和详细设计的顺序一致。不需要开发驱动模块。  
          缺点：随着测试的进行，测试过程越来越复杂，开发和维护成本增加。  
          总结：比孤立单元测试的成本高很多，不是单元测试的一个好的选择。
       2. 自底向上的单元测试策略  
          方法：先对最底层的基本单元进行测试，模拟调用该单元的单元做驱动模块。然后再对上面一层进行测试，用下面已被测试过的单元做桩模块。依此类推，直到测试完所有单元。  
          优点：在集成测试前提供系统早期的集成途径。不需要开发桩模块。  
          缺点：随着测试的进行，测试过程越来越复杂。  
          总结：比较合理的单元测试策略，但测试周期较长。
       3. 孤立单元测试策略  
          方法：不考虑每个单元与其它单元之间的关系，为每个单元设计桩模块或驱动模块。每个模块进行独立的单元测试。  
          优点：简单、容易操作，可达到高的结构覆盖率。  
          缺点：不提供一种系统早期的集成途径。  
          总结：最好的单元测试策略。
    3. 集成测试的测试策略：
       1. 大爆炸集成  
          优点：可以迅速完成集成测试；并且只要极少数的驱动和桩模块；用例也是最少的；简单；资源利用率高  
          缺点：一次试运行成功的可能性不大，问题定位和修改比较困难，许多接口错误很容易躲过测试。  
          适应于一个维护型项目或被测试系统较小
       2. 自顶向下集成  
          优点：较早地验证了主要控制和判断点；按深度优先可以首先实现和验证一个完整的软件功能；功能较早证实，带来信心；只需一个驱动，减少驱动器开发的费用；支持故障隔离。  
          缺点：柱的开发量大；底层验证被推迟；底层组件测试不充分。  
          适应于产品控制结构比较清晰和稳定；高层接口变化较小；底层接口未定义或经常可能被修改；产口控制组件具有较大的技术风险，需要尽早被验证；希望尽早能看到产品的系统功能行为。
       3. 自底向上集成  
          优点：对底层组件行为较早验证；工作最初可以并行集成，比自顶向下效率高；减少了桩的工作量；支持故障隔离。  
          缺点：驱动的开发工作量大；对高层的验证被推迟，设计上的错误不能被及时发现。  
          适应于底层接口比较稳定；高层接口变化比较频繁；底层组件较早被完成。
       4. 三明治集成  
          优点：集合了自顶向下和自底向上两种策略的优点  
          缺点：中间层测试不充分  
          适应于大部分软件开发项目
       5. 基干集成  
          优点：具有三明治集成的优点，更适合于大型复杂项目的集成。  
          缺点：必须对系统的结构和相互依存性进行仔细的分析；驱动和桩开发量大；局部采用了大爆炸的策略，有些接口可能测试不充分。  
          嵌入式系统中常用
       6. 分层集成  
          适应于有明显层次关系的系统
       7. 基于功能的集成  
          优点：优先验证关键功能的正确性；减少驱动的开发；进度要快。  
          缺点：对接口测试不充分；有较大的冗余测试。
       8. 基于消息的集成  
          优点：优先验证关键消息的正确性；减少驱动的开发；进度要快。  
          缺点：对接口测试不充分；有较大的冗余测试。
       9. 基于风险的集成  
          优点：最具有风险的组件最早进地验证，有助于系统的快速稳定。  
          缺点：需要对各组件的风险有一个清晰的分析。
       10. 基于进度的集成  
           优点：具有较高的并行度；能够有效缩短项目的开发进度。  
           缺点：桩和驱动工作量较大；有些接口测试不充分；有些测试重复和浪费。
    4. 系统测试的测试策略：
       1. 数据和数据库完整性测试
       2. 功能测试
       3. 用户界面测试
       4. 性能评测
       5. 负载测试
       6. 强度测试
       7. 容量测试
       8. 安全性和访问控制测试
       9. 故障转移和恢复测试
       10. 配置测试
       11. 安装测试
       12. 加密测试
       13. 可用性测试
       14. 版本验证测试
       15. 文档测试
12. 答案如下：
    1. Berard提出了一些测试用例的设计方法，主要原则包括:
       1. 每个测试用例应当给予特殊的标识，并且还应当与测试的类有明确的联系。
       2. 测试目的应当明确。
       3. 应当为每个测试用例开发一个测试步骤列表。这个列表应包含以下一些内容：
          1. 列出所要测试对象的专门说明。
          2. 列出将要作为测试结果运行的消息和操作。
          3. 列出测试对象可能发生的例外情况。
          4. 列出外部条件（即为了正确对软件进行测试所必须有的外部环境的变化）。
          5. 列出为了帮助理解和实现测试所需要的附加信息。
    2. 主要方法：
       1. 基于故障的测试   
          基于故障测试也可以用于组装测试，组装测试可以发现消息联系中”可能的故障”。   
          “可能的故障”一般为意料之外的结果、错误地使用了操作/消息、不正确引用等。为了确定由操作（功能）引起的可能故障必须检查操作的行为。  
          这种方法除用于操作测试外，还可用于属性测试，用以确定其对于不同类型的对象行为是否赋予了正确的属性值。因为一个对象的”属性”是由其赋予属性的值定义的。
       2. 基于脚本的测试  
          基于脚本的测试主要关注用户需要做什么，而不是产品能做什么，即从用户任务（使用用例）中找出用户要做什么及去执行。  
          这种基于脚本的测试有助于在一个单元测试情况下检查多重系统。所以基于脚本测试用例测试比基于故障测试不仅更实际（接近用户），而且更复杂一点。
       3. OO类的随机测试  
          如果一个类有多个操作（功能），这些操作（功能）序列有多种排列。而这种不变化的操作序列可随机产生，用这种可随机排列的序列来检查不同类实例的生存史，就叫随机测试。
       4. 类层次的分割测试  
          这种测试可以减少用完全相同的方式检查类测试用例的数目。这很像传统软件测试中的等价类划分测试。分割测试又可分三种。
          1. 基于状态的分割。按类操作是否改变类的状态来分割（归类）。
          2. 基于属性的分割。按类操作所用到的属性来分割（归类）。
          3. 基于类型的分割。按完成的功能分割（归类）。
       5. 由行为模型（状态、活动、顺序和合作图）导出的测试  
          状态转换图（STD）可以用来帮助导出类的动态行为的测试序列，以及这些类与之合作的类的动态行为测试序列。
13. 答案如下：
    1. 单元测试阶段。各独立单元模块在与系统地其他部分相隔离的情况下进行测试，单元测试针对每一个程序模块进行正确性校验，检查各个程序模块是否正确地实现了规定的功能。生成单元测试报告，提交缺陷报告。
    2. 集成测试阶段。集成测试是在单元测试的基础上，测试在将所有的软件单元按照概要设计规格说明的要求组装成模块、子系统或系统的过程中各部分工作是否达到或实现相应技术指标及要求的活动。该阶段生成集成测试报告，提交缺陷报告。
    3. 系统测试阶段。将通过确认测试的软件，作为整个给予计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行全面的功能覆盖。该阶段需要提交测试总结和缺陷报告。

1.

判断题

1.

软件测试就是为了验证软件功能实现的是否正确，是否完成既定目标的活动，所以软件测试在软件工程的后期才开始具体的工作。

2.

发现错误多的模块，可能残留在模块中的错误也多。

3.

测试人员在测试过程中发现一处问题，如果问题影响不大，而自己又可以修改，应立即将此问题正确修改，以加快、提高开发的进程。

4.

单元测试通常应该先进行”代码走查”，再以白盒法为主，辅以黑盒法进行动态测试。

5.

功能测试是系统测试的主要内容，检查系统的功能、性能是否与需求规格说明相同。

6.

软件质量管理即QM由QA和QC构成，软件测试属于QC的核心工作内容。

7.

软件测试只能发现错误，但不能保证测试后的软件没有错误。

8.

软件就是程序。

9.

测试只要做到语句覆盖和分支覆盖，就可以发现程序中的所有错误。

10.

I18N测试是指对产品做出具有国际化，而L10N测试则是指对软件做出符合本地化需求更改工作。

2.

选择题

1.

进行软件质量管理的重要性有：（）

A、维护降低成本 B、法律上的要求 C、市场竞争的需要

D、质量标准化的趋势 E、软件工程的需要 F、CMM过程的一部分

G、方便与客户进一步沟通为后期的实施打好基础

2.

以测试的形态分测试可以分为：（）

A、建构性测试 B、系统测试 C、专项测试

D、单元测试 E、组件测试 F、集成测试

3.

选出属于黑盒测试方法的选项（）

A、测试用例覆盖 B、输入覆盖 C、输出覆盖

D、分支覆盖 E、语句覆盖 F、条件覆盖

4.

编写测试计划的目的是：（）

A、使测试工作顺利进行 B、使项目参与人员沟通更舒畅 C、使测试工作更加系统化

D、软件工程以及软件过程的需要 E、软件过程规范化的要求 F、控制软件质量

5.

依存关系有4种分别是：（）

A、开始－结束 B、开始－开始 C、结束－开始

D、结束－结束 E、开始－实施－结束 F、结束－审核－开始

6.

软件质量管理（QM）应有质量保证（QA）和质量控制（QC）组成，下面的选项属于QC的是：（）

A、测试 B、跟踪 C、监督

D、制定计划 E、需求审查 F、程序代码审查

7.

实施缺陷跟踪的目的是：（）

A、软件质量无法控制 B、问题无法量化 C、重复问题接连产生

D、解决问题的知识无法保留 E、确保缺陷得到解决

F、使问题形成完整的闭环处理

8.

使用软件测试工具的目的：（）

A、帮助测试寻找问题 B、协助问题的诊断 C、节省测试时间

D、提高Bug的发现率 E、更好的控制缺陷提高软件质量

F、更好的协助开发人员

9.

典型的瀑布模型的四个阶段是：（）

A、分析 B、设计 C、编码

D、测试 E、需求调研 F、实施

10.

PSP是指个人软件过程 ，是一种可用于（）、（）和（）个人软件工作方式的自我改善过程。

A、控制 B、管理 C、改进

D、高效 E、充分 F、适宜

3.

问答题

1.

测试人员在软件开发过程中的任务是什么？

2.

在您以往的工作中，一条软件缺陷（或者叫Bug）记录都包含了哪些内容？如何提交高质量的软件缺陷（Bug）记录？

3.

黑盒测试和白盒测试是软件测试的两种基本方法，请分别说明各自的优点和缺点！

四、 设计题

1.

如何测试一个纸杯？

2.

测试notepad的文件保存功能，就是file/save弹出对话框的功能，从那几个方面写测试用例。

参考答案：

1.

答案如下

1.

错

2.

对

3.

错

4.

对

5.

对

6.

对

7.

对

8.

错

9.

错

10.

对

2.

答案如下

1.

ABCDEFG

2.

ABC

3.

ABC

4.

ABC

5.

ABCD

6.

ABC

7.

ABCD

8.

ABC

9.

ABCD

10.

ABC

3.

答案如下

1.

1、寻找Bug；

2、避免软件开发过程中的缺陷；

3、衡量软件的品质；

4、关注用户的需求。

总的目标是：确保软件的质量。

2.

一条Bug记录最基本应包含：编号、Bug所属模块、Bug描述、Bug级别、发现日期、发现人、修改日期、修改人、修改方法、回归结果等等；要有效的发现Bug需参考需求以及详细设计等前期文档设计出高效的测试用例，然后严格执行测试用例，对发现的问题要充分确认肯定，然后再向外发布如此才能提高提交Bug的质量。

3.

黑盒测试的优点有：

1.

比较简单，不需要了解程序内部的代码及实现；

2.

与软件的内部实现无关；

3.

从用户角度出发，能很容易的知道用户会用到哪些功能，会遇到哪些问题；

4.

基于软件开发文档，所以也能知道软件实现了文档中的哪些功能；

5.

在做软件自动化测试时较为方便。

黑盒测试的缺点有：

1.

不可能覆盖所有的代码，覆盖率较低，大概只能达到总代码量的30%；

2.

自动化测试的复用性较低。

白盒测试的优点有：

1.

帮助软件测试人员增大代码的覆盖率，提高代码的质量，发现代码中隐藏的问题。

白盒测试的缺点有：

1.

程序运行会有很多不同的路径，不可能测试所有的运行路径；

2.

测试基于代码，只能测试开发人员做的对不对，而不能知道设计的正确与否，可能会漏掉一些功能需求；

3.

系统庞大时，测试开销会非常大。

4.

答案如下

1.

答案如下：

1.

功能度：用水杯装水看漏不漏；水能不能被喝到

2.

安全性：杯子有没有毒或细菌

3.

可靠性：杯子从不同高度落下的损坏程度

4.

可移植性：杯子在不同的地方、温度等环境下是否都可以正常使用

5.

兼容性：杯子是否能够容纳果汁、白水、酒精、汽油等

6.

易用性：杯子是否烫手、是否有防滑措施、是否方便饮用

7.

用户文档：使用手册是否对杯子的用法、限制、使用条件等有详细描述

8.

疲劳测试：将杯子盛上水（案例一）放24小时检查泄漏时间和情况；盛上汽油（案例二）放24小时检查泄漏时间和情况等

9.

压力测试：用根针并在针上面不断加重量，看压强多大时会穿透

2.

答案如下：

1.

对文件名长度进行上下边界的测试

2.

对文件名进行等价类划分，测试文件名合法和非法的情况，测试文件名里包含有保留字的情况。

3.

测试文件保存的两种类型。

4.

测试文件名和文件保存类型组合的情况。

5.

根据文件系统的软件故障模型，设计测试用例。

6.

测试File菜单是否存在界面问题。(保存对话框调用的系统对话框，此处无需检查)

7.

热键和快捷键