组长：陈豪

组员：蔡文怿 丁晓安 刁琳琳 林浩 罗汉祥

11.1

答：软件测试的目的是发现软件中的错误和缺陷，并加以纠正。

11.2

答：白盒测试又称结构测试，这种方法把测试对象看作一个透明的盒子，测试人员根据程序内部的逻辑结构及有关信息设计测试用例，检查程序中所有逻辑路径是否都按预定的要求正确地工作。白盒测试主要用于对程序模块的测试。包括：

* 程序模块中的所有独立路径至少执行一次。
* 对所有逻辑判定的取值（“真”与“假”）都至少测试一次。
* 在上下边界及可操作范围内运行所有循环。
* 测试内部数据结构的有效性等。

黑盒测试又称行为测试，这种方法把测试对象看作一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符号它的功能需求。黑盒测试可用于各种测试，它试图发现以下类型的错误：

* 不正确或遗漏的功能
* 接口错误，如输入输出参数的个数、类型等。
* 数据结构错误或外部信息（如外部数据库）访问错误。
* 性能错误。
* 初始化和终止错误。

11.3

解：判定覆盖：（1）X=85,Y=85

（2）X=70，Y=95

（3）X=30，Y=95

条件覆盖：（1）X=85，X=85

（2）X=70，Y=75

（3）X=95，Y=50

（4）X=50，Y=95

（5）X=40，Y=40

判定条件覆盖：（1）X=85，X=85

（2）X=70，Y=75

（3）X=95，Y=50

（4）X=50，Y=95

（5）X=40，Y=40

（6）X=20，Y=95

（7）X=95，Y=20

条件组合覆盖：（1）X=85，X=85

（2）X=65，Y=85

（3）X=85，Y=65

（4）X=70，Y=75

（5）X=95，Y=50

（6）X=50，Y=95

（7）X=40，Y=40

路径覆盖：（1）X=85,Y=85

（2）X=70，Y=95

（3）X=30，Y=70

**11.5分别简述单元测试、集成测试、确认测试和系统测试的任务。**

单元测试：

又称模块测试，着重对软件设计的最小单元——软件构件或模块进行验证。

单元测试根据设计描述，对重要的控制路径进行测试，已发现构建或模块内部的错误，通常采用白盒测试，并且多个构件或模块可以并行测试。

单元测试的主要内容：接口、局部数据结构、边界条件、独立路径和错误处理路径。

集成测试：

集成测试，也叫组装测试或联合测试。在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。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。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。

目的：是确保各单元组合在一起后能够按既定意图协作运行，并确保增量的行为正确。它所测试的内容包括单元间的接口以及集成后的功能。使用[黑盒测试](http://baike.baidu.com/view/51274.htm)方法测试集成的功能。并且对以前的集成进行[回归测试](http://baike.baidu.com/view/106720.htm)。

确认测试：

确认测试的目的是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是确认测试的任务，即软件的功能和性能如同用户所合理期待的那样。

系统测试：

将已经确认的[软件](http://baike.baidu.com/view/37.htm)、计算机硬件、外设、网络等其他元素结合在一起，进行信息系统的各种组装测试和确认测试，系统测试是针对整个产品系统进行的测试，目的是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。系统测试发现问题之后要经过调试找出错误原因和位置，然后进行改正。是基于系统整体需求说明书的黑盒类测试，应覆盖系统所有联合的部件。对象不仅仅包括需测试的软件，还要包含软件所依赖的硬件、外设甚至包括某些数据、某些支持软件及其接口等。

**11.6什么是α测试？什么是β测试？**

α测试：

α测试是由一个用户在开发者的场所进行的测试，软件在开发者对用户的“指导下”进行测试。经过α测试后的软件称为β测试。

β测试：

β测试是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。

**11．7什么是回归测试？**

回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。自动回归测试将大幅降低系统测试、维护升级等阶段的成本。回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。在渐进和快速迭代开发中，新版本的连续发布使回归测试进行的更加频繁，而在极端编程方法中，更是要求每天都进行若干次回归测试。因此，通过选择正确的回归测试策略来改进回归测试的效率和有效性是非常有意义的。

* 观念：

1.回归测试是指重复以前的全部或部分的相同测试。

2.新加入测试的模组，可能对其他模组产生副作用，故须进行某些程度的回归测试。

3.回归测试的重心，以关键性模组为核心。

* 测试策略：

对于一个软件开发项目来说，项目的测试组在实施测试的过程中会将所开发的[测试用例](http://baike.baidu.com/view/106882.htm)保存到“测试用例库”中，并对其进行维护和管理。当得到一个软件的[基线](http://baike.baidu.com/view/350200.htm)版本时，用于基线版本测试的所有测试用例就形成了基线测试用例库。在需要进行回归测试的时候，就可以根据所选择的回归测试策略，从基线测试用例库中提取合适的测试用例组成回归测试包，通过运行回归测试包来实现回归测试。保存在基线测试用例库中的测试用例可能是自动[测试脚本](http://baike.baidu.com/view/651490.htm)，也有可能是测试用例的手工实现过程。

回归测试需要时间、经费和人力来计划、实施和管理。为了在给定的预算和进度下，尽可能有效率和有效力地进行回归测试，需要对测试用例库进行维护并依据一定的策略选择相应的回归测试包。

* 测试过程

有了测试用例库的维护方法和回归测试包的选择策略，回归测试可遵循下述基本过程进行:

(1). 识别出软件中被修改的部分；

(2). 从原基线测试用例库T中，排除所有不再适用的测试用例，确定那些对新的软件版本依然有效的测试用例，其结果是建立一个新的基线测试用例库T0。

(3). 依据一定的策略从T0中选择测试用例测试被修改的软件。

(4). 如果必要，生成新的测试用例集T1，用于测试T0无法充分测试的软件部分。

(5). 用T1执行修改后的软件。

第(2)和第(3)步测试验证修改是否破坏了现有的功能，第(4)和第(5)步测试验证 修改工作本身。

**11．8简述边界值分析方法的作用**

长期的测试工作经验告诉我们，大量的错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

使用边界值分析方法设计测试用例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。

边界分析是指对输入或输出的边界值进行测试的一种测试方法。所说的边界值是指输入等价类和输出等价类的边界值。

经验证明大量的程序错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

使用边界值分析方法设计测试用例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。

**13.1请讨论使软件维护成本居高不下的因素。如何尽可能降低这些因素的影响？**

软件维护的代价是生产率惊人下降。维护费用只不过是软件及维护最明显的代价，起一些隐性的代价将更为人们关注。

软件维护除费用外的无形代价包括：

1.维护活动占用了其他软件开发可用的资源，使资源的利用率降低

2.一些修复或修改请求得不到及时安排，使得客户满意率下降

3.维护的结果把一些新的潜在的错误引入软件，降低了软件质量

4.将软件人员抽调到维护工作中，使得其它软件开发过程受到干扰

影响维护工作量的因素主要有以下六种：

1.系统的规模：系统规模越大，其功能就越复杂，软件维护的工作量也随之增大

2.程序设计语言：使用强功能的程序设计语言可以控制程序的规模。语言的功能越强，生成程序的模块化和结构化程度越高，所需的指令数就越少，程序的可读性也越好

3.系统年龄：老系统比新系统需要更多的维护工作量。

4.数据库技术的应用：使用数据库，可以简单而有效地管理和存储用户程序中的数据，还可以减少生成用户报表应用软件的维护工作量

5.先进的软件开发技术：在软件开发过程中，如果采用先进的分析设计技术和程序设计技术，如面向对象技术、复用技术等，可减少大量的维护工作量

6.其它一些因素：如应用的类型、数学模型、任务的难度、IF嵌套深度、索引或下标数等，对维护工作量也有影响

为了有效的进行软件维护，尽可能降低这些因素的影响，应事先就开始做组织工作：

1.首先建立维护的机构

2.申明提出维护申请报告的过程及评价的过程

3.为每一个维护申请规定标准的处理步骤

4.建立维护活动的登记制度以及规定评价和评审的标准

**13.3软件维护的过程是如何进行的？为什么要进行软件可维护性分析？**

1.对于非纠错性维护，则首先判断维护类型，对适应性维护，按照评估后得到的优先级放入队列

2.对于改善性维护，则还要考虑是否采取行动，如果接受申请，则同样按照评估后得到的优先级放入队列，如果拒绝申请，则通知请求者，并说明原因

3.对于工作安排队列中的任务，由修改负责人依次从队列中取出任务，按照软件工程方法学规划、组织、实施工程。

4.每种维护请求都要进行同样的一系列技术工作：修改软件需求说明、修改软件设计、设计评审、必要时重新编码、单元测试、集成测试( 包括回归测试)、确认测试等

5.维护工作最后一步是复审

维护过程图：

**维护请求**

**类型**

**类型**

**严重性**

**评估后按优先**

**级在队列排队**

**救火行动，当**

**排在队列之首**

**评估后分类**

**评估后按优先**

**级在队列排队**

**采取的行动**

**通知请求者**

**并说明原因**

**按优先级在**

**队列中排队**

**从维护请求队列之首取出一任务**

**按SE方法学规划、组织、实施工程**

**队列中还有维护请求吗？**

**资源用于开发新的软件。**

**y**

**n**

**纠错性维护**

**其他**

**改善性维护**

**适应性维护**

**拒绝**

**接受**

**并不严重**

**非常严重**

进行可维护性分析的原因：

软件维护是指软件系统交付使用以后，为了改正错误或满足新的需求而修改软件的过程。一个中等规模的软件，如果其开发过程需要一二年时间，则它投入使用以后，其运行时间可能持续5～10年之久。在这个维护阶段中，人们需要着手解决开发阶段尚未解决的问题，同时，还解决维护工作本身所产生的问题。做好软件的维护工作不仅能够排除软件中存在的错误，使它能够正常工作，而且还可以使它扩充功能，提高性能，为用户带来新的效益。软件的可维护性是指维护人员为纠正软件系统出现的错误或缺陷，以及为满足新的要求而理解、修改和完善软件系统的难易程度。可维护性是所有软件系统都应具备的特点。在软件工程的每一阶段都应该努力提高系统的可维护性，在每个阶段结束前的审查和复审中，应着重对可维护性进行复审。维护阶段的花费约占整个软件生命期花费的67%。因此，应充分认识到维护现有软件的重要意义。  
**3.7在重构和正向工程之间存在的细微不同是什么？**

当某应用的基本体系结构是坚固的时候发生重构，即使技术的内部细节需要修改。当软件的大部分是有用的，仅仅部分模块和数据需要扩展性修改时，启动重构活动。正向工程是通过到实现语言的映射而把模型转换为代码的过程。正向工程过程应用软件工程的原理、概念、技术和方法来重新开发某个现有的应用系统。