附录A

小测验问题解答

第1章

1. 在千年虫例子中，Dave有错误吗？

如果Dave是个好的程序员，他应该对这个“显然的”疏忽产生疑问而不是仅仅将程序设计到只能有效工作到1999年。由于他没有这样做，软件测试员就应该测试并发现该缺陷，然后由开发小组确定是否修正。

2. **判断是非**：公司或者开发小组用来称呼软件问题的术语很重要。

错。这虽然不重要，但是使用什么术语常常反映了小组的个性及其寻找、报告和确定问题的方法。

3. 仅仅测试程序是否按预期方式运行有何问题？

这最多只能算测试问题的一半。用户不一定遵守规则，软件测试员需要证实不按操作有何后果。此外，如果测试员进行测试没有打破砂锅问到底的态度就会遗漏某些软件缺陷。

4. 产品发行后修复软件缺陷比项目开发早期这样做的费用要高出多少？

10~100倍，甚至更高。

5. 软件测试员的目标是什么？

软件测试员的目标是尽可能早一些找出软件缺陷，并确保其得以修复。

6. **判断是非**：好的测试员坚持不懈地追求完美。

错。好的测试员知道何时完美无法企及，何时达到“够好”。

7. 给出几个理由说明产品说明书为什么通常是软件产品中制造缺陷的最大来源。

产品说明书常常没写——不要忘了，说不出来就做不出来。其他原因是产品说明书虽然有，但是不完整，不停更改，或者产品说明书内容没有同开发小组其他成员沟通过。

第2章

1. 说出在程序员开始编写代码之前要完成哪些任务？

开发小组需要了解客户的要求，在产品说明书中定义功能特性。应该建立详细的进度，使小组成员知道哪些工作已经完成，哪些工作还要做。软件应该形成体系，经过设计，测试小组应该开始计划工作。

2. 正式并被锁定不能修改的产品说明书有何缺点？

如果软件开发过程中市场转移到不同的方向上或者客户要求改变，就没有调整软件的灵活性。

3. 软件开发大爆炸模式的最大优点是什么？

简单。仅此而已。

4. 采用边写边改模式时，如何得知软件发布的时间？

边写边改模式没有真正的退出标准，除非某人或者进度决定该结束了。

5. 瀑布模式为什么不好用？

像大马哈鱼一样，很难向上游。每一步都是跟着上一步的独立、离散的过程。如果走到头发现有些事情应该早些做时，想退回来就来不及了。

6. 软件测试员为什么最喜欢螺旋模式？

他们很早就参与开发过程，有机会尽早发现问题，为项目节省时间和金钱。

第3章

1. 假定无法完全测试某一程序，在决定是否应该停止测试时要考虑哪些问题？

终止测试没有一定的时间，每一个项目都会有所不同。决定时要考虑的因素有：仍然会发现大量软件缺陷？项目小组对已执行的测试满意吗？报告的软件缺陷是否经过评估定下来哪些修复，哪些不修复？产品按照客户的要求验证了吗？

2. 启动Windows计算器程序，输入5 000−5=（逗号不能少），观察结果。这是软件缺陷吗？为什么？

答案是0，而不是预期的4995。其原因是逗号（,）被自动转换为小数点（.）。于是算式变为了5.000−5=0，而不是5 000−5=4995。要确定这是否为软件缺陷，就需要根据产品说明书进行合法性检查，也许在产品说明书上声明逗号会被转换为小数点。还要对照用户需求进行验证，看大多数用户是接受这点还是产生迷惑。

3. 假如测试模拟飞行或模拟城市之类的模拟游戏，精确度和准确度哪一个更值得测试？

模拟游戏的目的是使游戏者置身于与现实情形接近的虚构环境中。在模拟器中飞行应该感觉像在真飞机上一样。城市模拟就应该反映真实城市的各种情况。最重要的是如何准确地模拟实际情形。飞机是像波音757一样还是像一只小鸟一样飞行？城市航线与实际路线相仿吗？软件有了准确性，才能谈到精确。这是关心建筑物中的窗户位置是否准确以及飞机的移动是否与游戏杆操作完全协调的第一点。

4. 有没有质量很高但可靠性很差的产品？请举例说明。

有可能，但是它取决于客户对质量的期望。不少人购买高性能跑车，认为提速、时速、式样、舒适度和装饰好就是高质量。此类汽车一般可靠性较差，经常抛锚，修理费用昂贵，而车主不把可靠性差当做严重的质量问题。

5. 为什么不可能完全测试程序？

除了极短小的简单程序，完全测试需要太多输入、输出和分支组合。此外，软件说明书也许不客观，可以用多种方式解释。

6. 假如周一测试软件的某一功能，每小时发现一个新的软件缺陷，你认为周二将会以什么样的频率发现软件缺陷？

这里有两个基本要素。首先——余下的软件缺陷与发现的软件缺陷成比例——意味着周二

不会比周一的情况好多少。其次，杀虫剂现象表明，除非增加新的测试，否则反复执行同样的测试，不会发现不同的新软件缺陷。综合这两个软件要素，可能发现软件缺陷的速度继续保持原有的频率，甚至更低。

第4章

1. 软件测试员可以根据产品说明书进行白盒测试吗？

是的，白盒测试就是使用如何设计影响如何测试的概念进行的。测试员可以参加焦点人群、易用性研究和市场会议，了解用于定义功能特性和整个产品的过程。但是这存在一定的风险，因为这些信息诱使测试员倾向于假定说明书是正确的。

2. 试举一些Mac或Windows标准规范的例子。

在Mac机上，删除的文件放在废纸箱；在Windows中，删除的文件放在回收站。

在Windows中，按F1总是显示软件的帮助，在Mac机上则是Command-？。

在Windows中，File菜单总是最左边的菜单选项。

在Windows中，选择Help菜单中About显示软件的版权、许可权和版本信息。

在Mac机上，Command-X执行剪切操作，Command-C执行复制操作，Command-V执行粘贴操作。

还有很多例子。

3. 指出下述产品说明中的错误：当用户选择Compact Memory 选项时，程序将使用Huffman 解析矩阵方法尽可能压缩邮件列表数据。

错误在于使用了“尽可能”的说法。这一点无法测试，因为该说法没有量化、不精确。说明书应该说明压缩究竟达到何种程度才行。

4. 解释软件测试员应该担心下述产品说明的哪些内容：尽管通常连接不超过一百万个，但是该软件允许多达一亿个并发的连接。

可测试性。典型应用只有一百万个倒无关紧要。如果产品说明书声明有一亿种可能性，那么，一亿个连接都要测试。测试员需要设法测试这么多可能性，或者让说明书作者把最大可能性降低到接近典型应用的数目。

第5章

1. **判断是非**：在没有产品说明书和需求文档的条件下可以进行动态黑盒测试。

对。该技术称为探索测试，基本上把软件用做产品说明书。这不是理想的过程，但是急了也能用。最大的风险是不知道特性是否被遗漏。

2. 如果测试程序向打印机输送打印内容，应该选用哪些通用的失效性测试用例？

可以尝试打印时不加纸，或者使其卡纸。可以脱机打印，拔掉电源，断开打印机电缆。可以尝试在墨粉不足的条件下打印，甚至不加墨盒。为了明确所有的可能性，可以查看打印机的操作手册，找出支持的错误处理，设法建立使用的错误情况。

3. 启动Windows写字板程序，并从File菜单选取Print命令，打开如图5-12所示的对话框。左下角显示的Print Range（打印区域）特性存在什么样的边界条件？

如果选择Page选项，From和To文本域就变为可用状态。明显的边界条件是0到99999，即文本域的最小值和最大值。增加测试254，255，256，1023，1024和1025等内部边界是明智的

做法。此外，还有其他的内部边界。试着从只有6页的文档打印第1~8页。注意在本例中，软件必须在打印完第六页之后停止，是因为数据没有了，而不是接到停止命令。这是一个不同的内部边界。看看是否还能想出别的。

4. 假设有一个文本框要求输入10个字符的邮政编码，如图5-13所示。对于该文本框应该进行怎样的等价划分？

至少应该有以下的等价划分，但是还可以想出更多：

- 合法的5位数字邮政编码。合法是指所有字符都是数值，不是指投入使用的现有邮政编码——但这可以构成另一个区间。
- 合法的9位数字（带连线的9位数字）邮政编码。
- 5位以下数字。例如只有4位数字。
- 9位以下数字。例如只有8位数字。
- 5位以上数字。例如不带连线的8位数字。这是否与9位以下数字区间相同呢？
- 9位以上数字，尽管不可能输入9位以上带连线的数字，但是测试员应该尝试一下。
- 10位数字，无连线。与9位以上数字区间稍有差别。
- 连线位置不对。
- 连线不只一条。
- 无数字和无连线。

5. **判断是非**：访问程序的所有状态也确保了遍历各种状态之间的转换。

错。想想游览遍布美国的50个不同城市。可以制定到达每个城市的旅游计划，但是不可能走遍所有城市之间的道路——这将是走遍美国的所有道路。

6. 绘制状态转换图有多种不同的方法，但是它们都具有相同的三个要素是什么？

- 软件可能处于的每一个状态。
- 从一个状态转移到另一个状态所需的输入和条件。
- 当进入和退出状态时产生的条件、变量和输出。

7. Windows计数器程序的初始状态变量有哪些？

初始显示值和内部中间值置为0。存储寄存器（MC,MR,MS和M+按钮）置为0。剪切板内容（暂存剪切、复制和粘贴数据）保持不变。

另一个初始状态变量是计算器启动时出现在屏幕上的位置。打开计算器程序的多个副本，注意其位置不一定相同（至少在Windows95/98中如此）。作为探索测试的一个练习，看能否找出计算器打开时出现位置的规则。

8. 当设法显露竞争条件软件缺陷时，要对软件进行何种操作？

尝试同时做几件事。它们可以是相关的，例如从一个应用程序同时向两台打印机输出打印；也可以是无关的，例如在计算机计算时按各种键。所做的目的是迫使软件执行某一功能时出现与自己竞争的状况。

9. **判断是非**：在进行压迫测试的同时进行重负测试是不合情理的。

错。任何测试都是合理的。软件测试员的任务是发现软件缺陷。但是，由于软件测试的实质，在这种情况下发现的任何缺可能都不会修复。

第6章

1. 说出进行静态白盒测试的几个好处。

静态白盒测试在开发过程早期发现软件缺陷，使修复的时间和费用大幅降低。软件测试员可以得到软件如何运做的信息，存在哪些弱点和危险，而且可以与程序员建立良好的伙伴关系。项目状态可以传达给参与测试的所有小组成员。

2. **判断是非**：静态白盒测试可以找出遗漏之处和问题。

对。遗漏的问题比普通的问题更重要，通过静态白盒测试可以发现。当根据公布的标准和规范检查代码，在正式审查中仔细分析时，遗漏的问题就显而易见了。

3. 正式审查由哪些关键要素组成？

过程。按照过程进行是正式检查和两个程序员之间互查代码的区别。

4. 除了更正式之外，检验与其他审查类型有什么重大差别？

主要区别是，检验时在场的不是代码的原创者。这迫使另一个人完全理解要检查的软件。这比让其他人只是审查软件寻找软件缺陷更有效。

5. 如果要求程序员在命名变量时只能使用8个字符并且首字母必须采用大写的形式，那么这是标准还是规范呢？

这应该算是标准，如果程序员被告知要用超过8个字符命名，那么，这就是规范了。

6. 你会采用本章的代码审查清单作为项目小组验证代码的标准吗？

不会。这只是用做一个通用的示例。其中虽然有一些好的测试用例，应该在测试代码时考虑，但是，应该研读其他公开的标准之后再采用自己的标准。

7. 缓冲区溢出错误作为一个常见的安全问题属于哪一级错误？是由什么原因引起的？

数据引用。它们是由于使用了未正确声明或未进行初始化的变量、常量、数组、字符串或记录。

第7章

1. 为什么了解了软件的工作方法会影响测试的方式和内容？

如果仅从黑盒子的角度测试软件，就无法知道测试用例是否足以覆盖软件的各个部分，以及测试用例是否多余。有经验的黑盒子测试员能够为程序设计相当有效的测试用例，但是没有白盒测试知识，他就不知道这一套测试的好坏程度。

2. 动态白盒测试和调试有何区别？

这两个过程存在交叉。但是动态白盒测试的目的是为了发现软件缺陷，而调试的目的是修复软件缺陷。在分离和查找软件缺陷原因时发生交叉。

3. 在大爆炸软件开发模式下几乎不可能进行测试的两个原因是什么？如何解决？

一股脑交付软件，即使能够找出软件出现缺陷的原因，也非常困难——这是大海捞针的问题。第2个原因是软件缺陷众多、相互隐藏。顾此失彼，即使发现了缺陷，还是会发现软件仍然不行。

像构建软件时那样有步骤和条理地集成、测试模块，可以在软件缺陷相互重叠、隐藏之前将其找出。

4. **判断是非**：如果匆忙开发产品，就可以跳过模块测试而直接进行集成测试。

错。软件测试员可以这样做，但是不应该这样做。这样做的结果是会遗漏应该在早期发现的软件缺陷。跳过或推迟测试一般都会使项目完成时间延长、费用增加。

5. 测试桩和测试驱动有何差别？

测试桩用于自顶向下的测试。它用自己替换低级模块。其对于要测试的高级代码，外表和行为就像低级模块一样。

测试驱动和测试桩相反，用于自底向上的测试。它是代替高级软件，更有效地运行低级模块的测试代码。

6. **判断是非**：总是首先设计黑盒测试用例。

对。基于对软件行为操作的认识程度来设计测试用例，然后利用白盒测试技术进行检查使其更加有效。

7. 在本章描叙的三种代码覆盖中，哪一种最好？为什么？

条件覆盖是最好的。因为它综合了分支覆盖和语句覆盖，它保证决策逻辑中的所有条件(例如if-then语句等)，以及来自这些语句的所有分支和代码行都得到验证。

8. 静态和动态白盒测试最大的问题是什么？

很容易形成偏见。看到代码可能会说："啊，我知道了，不要测试这个案例，代码处理是对的。"实际上这是被表面蒙蔽了，去掉了必要的测试用例，一定要小心。

第8章

1. 部件和外设有何区别？

组件一般是指PC机内部的硬件设备；外设是PC机外部的。但是，根据硬件的类型，这个界限也可以突破。

2. 如何辨别发现的软件缺陷是普通问题还是特定的配置问题？

在几个不同的配置中重新运行暴露软件缺陷的相同步骤。如果软件缺陷不出现了，就可能是配置缺陷了。如果在不同的配置中都出现，就可能是通用的问题。但是一定要注意。配置问题可以跨越整个等价划分。例如，软件缺陷仅在激光打印机上出现，而在喷墨打印机上表现不出来。

3. 如何保证软件永远不会有配置问题？

这是一种技巧性问题。需要把硬件和软件打成一个包，软件只能在该硬件上运行，硬件必须完全密封，没有连接到外界的单独接口。

4. **判断是非**：选择测试的配置的时候只需考虑一种翻版的声卡。

视情况而定。翻版的硬件设备和原版完全一致，只是名称和包装盒不同。一般这两者功能是100%等价的，但是，有些设备的固件或驱动程序不同，比原版多出一些支持和附加特性。在这种声卡的测试上，需要在确定等价划分前计划的测试中了解这些声卡的不同和相同之处。

5. 除了年头和流行程度，对于配置测试，配置测试中用于等价划分硬件的其他原则是什么？

地区和国家是选择对象，因为某些硬件，例如DVD播放器只能播放当地使用的DVD。另一种可选对象是客户或者业务。某些硬件针对一些客户或者业务是需要的，而对其他不适用，把其他应用该软件的情况考虑在内。

6. 能够发布具有配置缺陷的软件产品吗？

可以。永远不可能把软件缺陷全部修复。在所有测试中，任何处理都是有风险的。测试员和测试小组需要决定哪些软件缺陷需要修复，哪些不需要修复。决定留下仅在少见的硬件中出现的不太重要的软件缺陷很容易，除此之外就没那么容易了。

第9章

1. **判断是非**：所有软件必须进行某种程度的兼容性测试。

错。有极少数独立使用、专用、不与任何外界打交道的软件不需要进行兼容性测试。但是，除此之外99%的软件都必须进行某种程度的兼容性测试。

2. **判断是非**：兼容性是一种产品特性，可以有不同程度的符合标准。

对。软件的兼容性取决于客户的要求。某个字处理程序与其竞争对手产品的文件格式不兼容或者新操作系统不支持某一游戏软件，都是正常的。软件测试员应该通过确定兼容性检查的工作量大小，为兼容性测试的决定提供依据。

3. 如果受命对产品的数据文件格式进行兼容性测试，应该如何完成任务？

研究接受测试的程序文件是否要求符合已有的标准。如果是这样，要测试程序确实遵循这些标准。对可能读写程序文件的程序进行等价划分。设计测试文档，使其包含程序能够读写的数据类型的典型范例。测试这些文件在接受测试的程序和其他程序之间是否能正确传输。

4. 如何进行向前兼容性测试？

向前兼容性测试不容易——对现在仍然见不到的东西进行测试毕竟难以实现。解决问题的方法是，完整细致地将测试定义在可以作为标准之处，从而使该标准成为判定向前兼容性测试的手段。

第10章

1. 翻译和本地化有何区别？

翻译只考虑语言的方面——翻译词语。本地化要照顾到地区和国家的习惯、风俗和文化。

2. 要了解他国语言才能测试本地化产品吗？

不必，但是，测试小组中要有人熟练掌握该语言。不懂该语言者可以测试与该语言无关的软件部分，但是懂一点外语可以促进测试。

3. 什么是文本扩展，由此可能导致什么样的常见软件缺陷？

当文本被翻译成其他语言时会出现文本扩展。文本字符串长度可能增加 1 倍或更长。原来在屏幕上适应对话框、按钮等的文本不再适应，甚至可能导致软件崩溃，因为变长的文本在为该字符保留的内存空间放不下，会覆盖其他内存空间。

4. 指出扩展字符可能导致问题的一些领域。

经过排序或者按字母排序的字词次序混乱，大小写转换出错，以及常见的显示和打印问题。

5. 使文本字符串与代码脱离为什么重要？

如果进行本地化的人只用修改文字而不必修改程序代码，工作就会简单多了。这样还会简化测试工作，因为已经知道软件的本地化版本中代码不变。

6. 说出在本地化程序之间可能变化的一些数据格式类型。

度量单位，例如磅、英寸和公升。24小时制或12小时制。最近随着欧洲一些国家成立欧盟，货币成为一个重要问题。如此等等。

第11章

1. **判断是非**：所有软件都有一个用户界面，因此必须测试易用性。

对。即使嵌入再深的软件终将以某种形式显露在用户面前。不要忘记UI可以简单到一个开关和一个灯泡，也可以复杂到飞行模拟器，即使软件在代码库中只有一个代码模块,其接口也要以变量和参数的形式显露在可以作为用户的程序员面前。

2. 用户界面设计是一门科学还是一门艺术?

两者兼而有之。许多在实验室经过严格研究、完全测试的用户界面，投入市场却是完全失败的。

3. 既然用户界面没有明确的对与错，怎样测试呢?

软件测试员应当检查其是否符合7个重要原则：符合标准和规范、直观、一致、灵活、舒适、正确和实用。

4. 列举熟悉的产品中设计低劣或者不一致的UI例子。

虽然答案根据具体使用的软件不能确定，但是考虑一下这个问题：试着调整汽车收音机时钟的时间——不看手册能行吗?

一些对话框OK按钮在左边，Cancel按钮在右边，而另一些对话框Cancel按钮在左边，OK按钮在右边。假如习惯了其中一种布局，在单击时看也不看，就会丢失工作成果!

在按听筒使用呼叫等待或者会议呼叫时，是否意外地挂断某人的电话?

然而，最好的例子始终是亲自找到的。

5. 哪4种残疾会影响软件的易用性?

视力、听力、运动和认知障碍。

6. 如果测试将启用辅助选项的软件，哪些领域需要特别注意?

处理键盘、鼠标、声音和显示的部分。如果为支持辅助选项的流行平台编写软件，就比完全从头编制辅助特性的测试工作要容易一些。

第12章

1. 启动Windows画图程序（见图12-4），找出应该测试的文档例子。应该找什么?

以下是几个例子：翻滚帮助——当鼠标停在某个画图程序绘图工具上方看到的弹出式描述。从Help菜单选择About命令显示版权和许可协议窗口。按F1启动联机帮助，阅读手册、按索引选择或者输入关键词搜索。还有功能帮助——例如，从Color 菜单中选择Edit Color命令，在标题栏单击“？”按钮，然后单击单其中的一种颜色，就会得到选择和创建颜色的帮助。

2. Windows画图程序帮助索引包含200多个条目，从airbrush tool到zooming in or out。是否要测试每一个条目看能够到达正确的帮助主题吗?假如有 10 000个索引条目呢?

任何一个测试都存在风险问题。如果有充足的时间测试所有索引项，就应该这样做。如果无法进行完全测试，就必须对认为有必要检查的对象建立等价划分。可以根据程序员关于索引系统工作方式的介绍来做决定；可以向文档作者请教索引项是如何生成的。可以尝试每一个开头字母，或者第1个，第2个，第4个，第8个，第16个……直至最后一个。甚至可以等到读完第15章“自动测试和测试工具”之后进行。

3. **判断是非**：测试错误提示信息属于文档测试范围。

对。但这不仅仅是文档测试。信息的内容需要作为文档测试；而强制信息显示和保证显示信息准确无误是代码测试的任务。

4. 好的文档以哪3种方式确保产品的整体质量?

增强易用性、提高可靠性、降低支持费用。

第13章

1. 在电影《WarGames》中攻破北美防空联合司令部（NORAD）的计算机系统背后的动机是什么？

使用和征服。可能还有一些挑战和树立威信的动机。

2. **判断是非**：威胁模型分析是一个由软件测试员执行的正式过程，用以确定在哪些地方最适合进行针对安全漏洞的测试。

错。这是由整个项目小组执行的正式过程。

3. JPEG病毒是由一个缓冲区溢出缺陷产生的。回到第6章的通用代码审核检查表，哪两类检查最能描述出这个溢出发生的原因？

计算错误——预计结果值只能为正数。当结果值变成负数时，会变成一个巨大的正数。数据引用错误——因为当数据变成一个巨大的正数后，目标缓冲区的大小就没有限制在注明的大小（65533字节）范围内。

4. 当尝试在标准的Windows应用程序中打开一个文件时出现最近使用过的文件清单，可能是安全漏洞中的哪一类数据的例子？

潜在数据。

5. 哪两类额外和潜在不安全的数据在文件保存到磁盘上时被无意识地写入磁盘？

RAM损耗和磁盘损耗。

第14章

1. 使用黑盒测试技术，网页的哪些基本元素可以轻易地测试到？

与多媒体光盘软件中的元素类似——文本、图像和超级链接。

2. 什么是灰盒测试？

灰盒测试是可以边看着代码、边利用代码的信息帮助测试。它不像白盒测试一样详细地检查代码。代码用来协助测试，但是测试并不完全基于代码。

3. 为什么网站测试可以使用灰盒测试？

因为很多网站主要由易于查看的HTML标记语言，而不是可执行程序构成。所以可以既快又轻松地看看网页的构成，然后依据此设计出测试。

4. 为什么不能依赖拼写检查工具来检查网页的拼写？

因为拼写检查器只能检查普通文本，不能检查图形化了的字母和随时间或每次查看时都改变的动态生成的文字。

5. 列出在进行网站兼容性测试和配置测试时需要考虑到的一些方面。

硬件平台、操作系统、Web浏览器、浏览器插件、浏览器选项和设置，视频分辨率和颜色深度、文字大小和调制解调器速度。

6. Jakob Neilsen的10个常见网站错误中哪几个会导致兼容性和配置缺陷？

盲目使用不成熟的新技术。现有硬件和软件第 1 次应用新技术时都容易出问题。这有一点技巧问题——本章没有讲，但愿在应用本书第三部分“运用测试技术”所学的内容之后能够找到答案。

第15章

1. 说出使用软件测试工具和自动化的一些好处。

它们可以加快执行测试用例的时间；能够提高软件测试员的效率，从而留出更多的时间进行测试计划和测试用例开发。它们精确且不会懈怠。

2. 在决定使用软件测试工具和自动化时，要考虑哪些缺点或者注意事项？

因为软件在产品开发过程中会变化，测试工具也要随着变化。测试员可能会陷入陷阱，花费太多时间去设计测试工具和自动化，而忽视了实际测试。容易过分依赖自动化。自己动手测试是无可替代的。

3. 工具和自动化之间有何差别？

测试工具有助于测试，简化手工完全测试任务。自动化也是一种工具，但是它的执行不需要人工干预。想一想在木匠呼呼大睡时电锯和钉锤能盖好房子吗？

4. 查看工具和注入工具有何异同？

这两种类型的工具都可以深入到软件中一般用户在正常情况下无法访问的地方。查看工具是非入侵式，只允许查看发生了什么。注入工具是入侵式的——不仅允许查看发生了什么，还可以操纵。利用这些工具，可以尝试在普通用户级层次难以执行或不可能执行到的测试用例。

5. **判断是非**：入侵式工具是最佳类型，因为其操作与测试的软件最贴近。

错。入侵式工具在一些情形下可以提供更好的信息和控制，但是它具有可能影响软件和测试结果的不利一面。最好是仔细评估每种情况，选择最适用的工具，且副作用最小。

6. 最简单但很有效的测试自动化类型是什么？

记录和回放测试用例，只需要手工执行测试一次，这是非常有效的。它把测试员从单调的重复性操作中解放出来，给测试员更多的时间用来寻找难以发现的软件缺陷。

7. 说出可以增加到问题6中所述的测试自动化中，使其更有效的一些特性。

简单编写一些步骤，而不只是捕捉步骤。暂停或等待软件对操作响应的能力。使宏知道软件缺陷是否出现的一些简单验证。

8. 聪明猴子比宏和笨拙的猴子有什么优点？

它们几乎都有自知能力，知道软件的状态图表，知道自己在哪里，能做什么。

第16章

1. 描述杀虫剂怪现象，如何找到新人查看软件来解决它？

杀虫剂现象（在第3章“软件测试的实质”中描述）是指不停地用同样的测试用例或者同一个人测试软件时出现的情形。最终，软件似乎对测试具有了免疫力，因为找不到新的软件缺陷。如果改变测试或者换新人测试，就会找出新的软件缺陷。软件缺陷在哪没有变，只是采用新的方法使其暴露出来。

2. 对软件进行beta测试程序有哪些正面作用？

由更多的人来检查软件。这是发现配置和兼容性问题的好方法。

3. 对于beta测试程序有哪些注意事项？

beta测试不能代替有组织、有计划、有条理的测试方法——在通常意义上的软件缺陷寻找

方面没有优势。测试员应该知道beta测试者的经验水平、设备，并要确保从测试中得到预期的结果。

4. 如果正在为小型软件公司测试，为什么外包配置测试是个好主意?

配备和管理一个配置测试实验室的开销和负担很重，小公司或者项目几乎无法承受。

第17章

1. 测试计划的目的是什么?

为了解释IEEE 829的定义，测试计划的目的是定义测试活动的范围、方法、资源和进度，明确要测试的条目，要测试的功能特性，要执行的测试任务，每个任务的负责人，以及与计划相关的风险，简而言之，使项目小组其他成员在测试小组如何测试软件上取得一致。

2. 为什么创建计划的过程是关键，而不是计划本身?

因为测试计划中定义的所有问题和其他项目功能小组或者小组成员相互之间存在影响。让所有人了解和接受计划的内容是关键所在。私自建立书面文档并束之高阁不仅浪费时间，而且对项目形成危害。

3. 为什么定义软件的质量和可靠性目标是测试计划的重要部分?

因为如果顺其自然的话，每个人都会有自己对质量和可靠性的看法。由于看法各不相同，因此全部达到是不可能的。

4. 什么是进入和退出规则?

从一个测试阶段转移到另一个阶段的要求必须满足。一个阶段满足退出规则才会结束，新的阶段满足进入规则才会开始。

5. 列出在测试计划时应该考虑的一些常用测试资源。

人员、设备、办公场所和实验室、软件、外包公司和其他供给。

6. **判断是非**：制订进度要符合固定日期，以使测试任务或者测试阶段何时开始，何时结束没有异议。

错。因为测试对项目的其他方面非常依赖（例如，代码编制没有完成就不能测试），所以测试进度最好根据交付日期来制定。

第18章

1. 测试用例计划的4个理由是什么?

组织性、重复性、跟踪和测试证实。

2. 什么是特别测试?

特别测试是没有计划的测试。它很容易，也很有趣，但是没有组织性、无法重复，也无法跟踪，完成后，无法证实曾经执行过。

3. 测试设计说明的目的是什么?

测试设计说明的目的是组织和描述针对某种功能特性要实施的测试。它列举了要测试的功能特性和要用的方法。它明确了测试用例，但是不指明具体是什么，也不说明通过／失败的原则是什么。

4. 什么是测试用例说明?

该文档定义测试的实际输入值和预期输入结果，还指明具体的环境要求、程序要求和测

试用例之间的依赖性。

5. 除了传统的文档，可以用什么方式表述测试用例?

表格、真值表、列表和示意图——能最有效地给自己、其他测试员和项目小组其他成员表示测试用例的任何形式。

6. 测试程序说明的目的是什么?

测试程序说明的目的是明确执行测试用例所需的全部步骤，包括如何设置、启动、执行和关闭测试用例。它还解释了测试未按计划进行时应该怎么做。

7. 编写测试程序应该达到何种详细程度?

这个问题没有指定的答案，很大程度上取决于谁来使用程序。细节太少会导致程序不明确以及变化不定。细节太多会拖延测试进度。详细程度应该由行业、公司、项目和测试小组来设定。

第19章

1. 说出软件缺陷可能不修复的几个原因。

进度中没有安排足够的时间，或不是软件缺陷，或修复风险太大不值得修复，以及软件缺陷没有正确报告。

2. 哪些基本原则可能应用于软件缺陷报告，使软件缺陷获得最大的修复机会?

尽早记录。有效描述软件缺陷，确保其最小化、单一、明显、全面、可以再现。在进行过程中不掺杂个人看法。在软件缺陷的整个生命周期中跟踪报告。

3. 描述分离和再现软件的一些技术。

记录所做的操作，并仔细审查。利用白盒测试技术寻找竞争条件、边界条件、内存泄露和其他类似问题。看软件缺陷是否与状态相关，例如依赖初始状态或其后状态。考虑导致软件缺陷的资源依赖性，甚至硬件问题。

4. 假设正在Windows计算器上执行测试，发现1+1=2，2+2=5，3+3=6，4+4=9，5+5=10，6+6=13。写一个软件缺陷标题和有效描述该问题的软件缺陷描述。

标题：偶数之间的加法得到的结果比实际值大 1 。

描述：

测试用例：*简单加法*

设置步骤：*启动计算器程序1.0版*

再现步骤：*尝试两个偶数相加，例如2＋2，4＋4，6＋6*

预期结果：*所有数字对相加得到正确结果——2+2＝4，4+4＝8，……*

实际结果：*两个偶数相加，答案比正确值大1——2＋2＝5，4＋4＝9，6＋6＝13，依此类推*

5. 在软件启动画面上公司徽标中的错别字应该有什么样的严重性和优先级?

可能是严重性 3 （小问题），优先级2（必须在发布之前修复）。

6. 什么是软件缺陷生命周期的3个基本状态和2个附加状态?

基本状态为打开、解决和关闭。两个附加状态是审查和推迟。

7. 列举数据库软件缺陷跟踪系统比纸张系统有用得多的一些原因。

可以一眼看出处于软件缺陷生命周期的哪个状态——即使对于复杂的软件缺陷也不例外。

软件缺陷不会被轻易遗漏或者忽略。项目统计数据可以很快获得。

第20章

1. 如果使用源自软件缺陷跟踪数据库数据的度量来评估进展或者测试成效，为什么只计算每天发现的软件缺陷数目或者平均发现速度是不充分的？

这没有说明问题的全部。测试员有可能正在测试软件最复杂的部分。测试区域可能是由最富有经验的程序员编写的，也可能是由最没有经验的程序员编写的。测试的代码可能已经测试过，也可能是全新的。

2. 根据问题1的答案，列举可以更精确、更准确评估个人测试进展或者测试成效的一些其他软件度量。

每天平均发现的软件缺陷数目。目前发现的软件缺陷总数。修复的缺陷和所有发现的缺陷的比例。严重性1或优先级1的软件缺陷与全部发现的软件缺陷的比例。从解决状态到关闭状态的平均时间。

3. 从Calc-U-Lot v3.0项目中，提取出提交给Terry的所有已解决软件缺陷，其数据库查询是什么样的（任何需要的格式）？

Product EQUALS Calc-U-Lot AND
Version EQUALS 3.0 AND
Status EQUALS Resolved AND
Assign TO EQUALS Terry

4. 如果某项目中软件缺陷发现速度如图20-8所示的那样下降，全体人员都对项目即将关闭准备发布表示兴奋，请问可能有哪两个原因会导致这种受数据欺骗的假像？

可能是软件要进入发布测试阶段，然而软件的所有部分并未全部被测试——仅仅在当前阶段曲线是平坦的。测试员可能忙于回归测试和关闭软件缺陷，而无暇寻找新的软件缺陷。还可能是在温暖的周末，或者测试员出去度假了。

第21章

1. 测试费用为什么与一致性的费用相关？

因为无论开发过程多好，都要进行一次测试，根据产品说明书验证产品，根据用户需求进行合法性检查。如果没有发现软件缺陷，太好了，但是计划、开发和执行测试的费用都要算进一致性费用中。

2. **判断是非**：测试小组负责质量。

错。测试的目的是发现软件缺陷。测试员不会在产品中放进软件缺陷，也不保证完成测试时不再有软件缺陷。

3. 为什么要获得QA工程师的称号是难以做到的？

因为这意味着你要保证产品质量。准备好承担这个责任了吗？

4. 为什么测试小组或者质量保证小组独立向高级管理员报告好？

如果他们向开发经理或者项目经理报告，就会在寻找软件缺陷和编制软件或者进度会议之间产生利益冲突。

5. 如果某公司在软件中编入ISO 9000-3标准，那么它的CMM级是多少？为什么？

它们可能处于CMM3级，也许达到CMM4级的某些要求。它们的CMM级不是2级，因为2级只到项目级。3级达到整个组织或者公司级。4级开始运用统计控制。

第22章

1. 在因特网上搜索软件测试职位时，应该使用什么样的搜索关键字？

由于软件测试员的工作名称和描述是变化的，因此应该搜索software test 、software testing、 quality assurance和QA。

2. 说出在软件公开发布之前参加测试的三条途径。

beta测试和易用性测试是两个好的方法。另外一个方法是测试开源代码的软件并使用开源的测试工具。甚至可以从找缺陷中获得回报。

3. 软件测试员的目标是什么？

软件测试员的目标是尽可能早一些找出软件缺陷，并确保其得以修复——当然要耐心地掌握本书所讲的软件测试的所有实质。祝大家好运，加油去寻找软件缺陷吧!