软件测试复习文档

题型

- 选择 15*2
- 简答 5~10
- 解答 5~15

1. 选择

- 等价准则
- 等价类的特点
  - 完备性：划分出的各个等价类（子集）的并是输入/输出的子集。
  - 无冗余性：各个等价类是互不相交的一组子集。
  - 等价性：各个输入数据对于揭露程序中的错误都是等效的。
- 等价类分类
  - 有效等价类：对于程序的规格说明来说是合理的、有意义的输入数据构成的集合。
  - 无效等价类：对程序的规格说明来说是不合理的或无意义的输入数据所构成的集合。
- 正交表
- 逻辑覆盖由弱到强：
  1. 语句覆盖（最弱）
  2. 判定覆盖
  3. 条件覆盖
  4. 判定-条件覆盖
  5. 条件组合覆盖
  6. 路径覆盖（最强）
- 例题
  1. 侧重于观察资源耗尽情况下的软件表现的系统测试被称为 （ 压力测试 ）
  2. 在下面所列举的逻辑测试覆盖中，测试覆盖最弱的是（ 语句覆盖 ）
  3. 不属于白盒测试的技术是（ 边界值分析 ）
  4. 划分软件测试属于白盒测试还是黑盒测试的依据是 （ 是否能看到被测源程序 ）
  5. 单元测试一般以（ 白盒测试 ）为主。
  6. 某次程序调试没有出现预计的结果，下列（ 编写的语句书写格式不规范 ）**不可能**是导致出错的原因。
  7. **不属于**单元测试的内容是 （ 用户界面测试 ）
  8. 修复软件缺陷费用最高的是（ 发布 ）阶段
  9. （ 语句覆盖 ）是选择若干个测试用例，运行被测程序，使得程序中的每个可执行语句至少执行一次
  10. 软件测试是软件质量保证的重要手段，下述哪种测试是软件测试的最基础环节？（ 单元测试 ）
  11. 连接速度测试属于(性能测试)
  12. 软件测试的目的是（ 尽可能发现并排除软件开发中的错误，提升软件可靠性 ）。

13. 软件测试是软件质挝保证的重要手段，下述哪种测试是软件测试的最基础环节（单元测试）
14. 导致软件缺陷的最大原因是（软件需求说明书）
15. 单元测试中用来模拟实现**被测模块需调用的其他功能模块**的是（桩模块）。
16. 集成测试计划应该在（概要设计）阶段末提交。
17. 软件测试是采用（测试用例 ）执行软件的活动。
18. 在下列描述中，关于测试与调试的说法错误的是（测试必须在详细设计已经完成的情况下才能开始；没有详细设计的信息调试不可能进行）
19. （判定 - 条件覆盖）是设计足够多的测试用例，使得程序中每个判定包含的每个条件的所有情况（真/假）至少出现一次，并且每个判定本身的判定结果（真/假）也至少出现一次。
20. 单元测试的主要任务不包括（全局数据结构）
21. 下列关于程序效率的描述错误的是（ 源程序的效率与详细设计阶段确定的算法的效率无关）。
22. 下列（ 控制流图 ）是对程序流程图进行简化后得到的，它可以更加突出的表示程序控制流的结构，且不包含复合条件。
23. 自底向上增量式集成测试中，下面（ 父单元用测试过的子单元测试 ）描述是正确的。
24. 测试后程序中残存的错误数目与该程序中已发现的错误数目成（ 正比 ）。
25. 针对是否对无效数据进行测试，可以将等价类测试分为（ 1）标准(一般)等价类测试 2）健壮等价类测试 ）
26. 对于正交表，L次数（水平数因子数），设函数有三个参数x,y, z,分别取值为，(1,2,3),(1,2,3)(1,2,3,4),若完成正交表测试，至少需要的测试用例个数为 （12）
27. 软件测试用例主要由测试输入数据和(测试的预期结果)两部分组成。

2. 简答

2.1 软件质量的定义

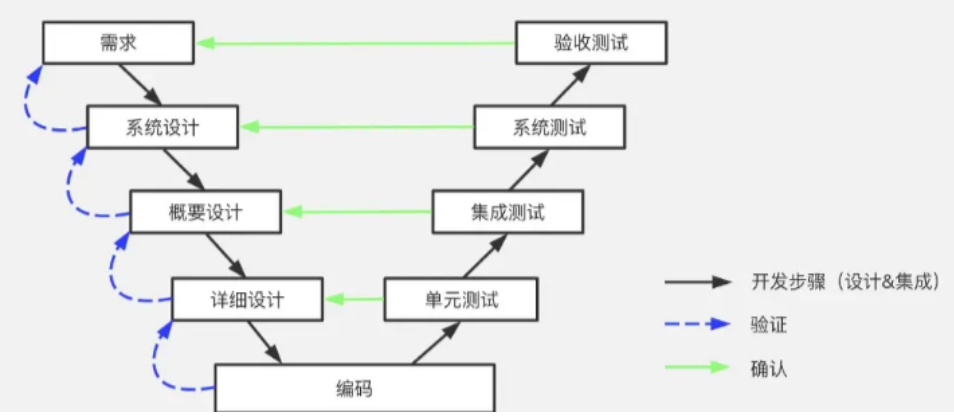
- 概括地说,软件质量是软件产品满足规定的和隐含的与需求能力有关的全部特征和特性。
- 包括功能性、可靠性、易使用性、效率、可维护性、可移植性和安全性等

2.2 软件缺陷类型、等级

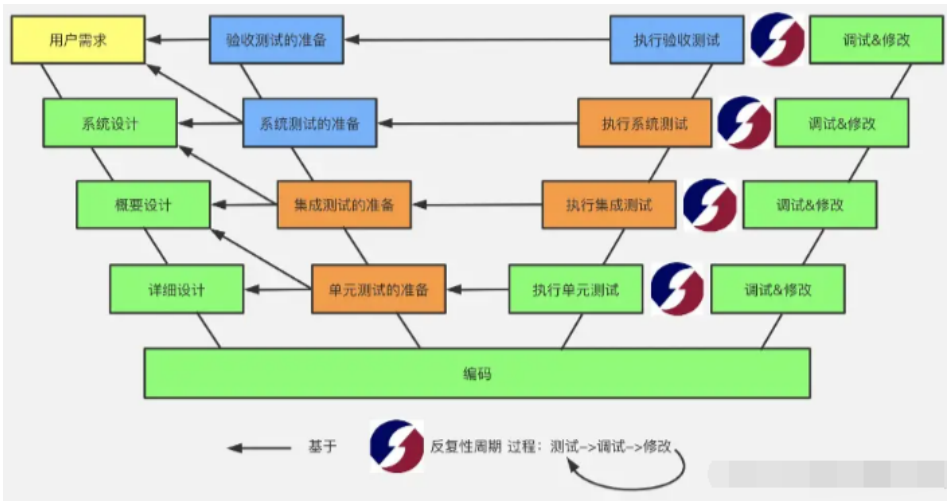
- 1）．致命错误，可能导致本模块以及其他相关模块异常，死机等问题；
  2）．严重错误，问题局限在本模块，导致模块功能失效或异常退出
  3）．一般错误，模块功能部分失效；
  4）．建议问题，由问题提出人对测试对象的改进意见；

2.3 软件测试模型，V/W模型的区别

- **V模型**
  -



- 特点
  1. 强调了在整个软件项目开发中需要经历的若干个测试级别，并与每一个开发级别对应；
  2. 既包括底层测试又包括了高层测试。低层测试是为了源代码的正确性，高层测试是为了使整个系统满足用户的需求。
- 局限性
  1. 把测试作为编码之后的最后一个活动，需求分析等前期产生的错误直到后期的验收测试才能发现；
  2. 将开发和测试过程划分为固定边界的不同阶段，使得相关人员很难跨过这些边界来采集测试所需要的信息；
  3. 容易让人形成“测试是开发之后的一个阶段”、“测试的对象就只是程序”等误解。

- **W模型**
  -


- 特点
  1. 测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、功能和设计同样要测试；
  2. 体现“尽早地和不断地进行软件测试”的原则；
  3. 在V模型中增加软件和开发阶段应同步进行的测试。
- 局限性
  1. 在W模型中，需求、设计、编码等活动被视为串行的，同时，测试和开发活动也保持着一种线性的前后关系，上一阶段完全结束，才可正是开始下一阶段工作，这就无法支持迭代、自发性以及变更调整。

- **区别**

1. 结构：V模型是简单的线性结构，而W模型是在V模型基础上扩展的结构，增加了评审和确认阶段，形成了W形的结构。
2. 测试活动：W模型比V模型更加全面，涵盖了额外的评审和确认活动。

3. 重点：W模型强调了评审和确认活动的重要性，以确保产品在进入下一个阶段之前符合质量标准和要求。

2.4 优秀的软件工程师应具有的素质

1. 技术知识：需掌握软件开发的基础知识和相关技术；
2. 分析能力：要能逻辑清晰地分析问题并找到解决方案；
3. 注意细节：必须具备发现软件微小缺陷的敏锐观察力；
4. 沟通能力：应能有效沟通，确保信息在团队中准确传达；
5. 学习能力：对新技术持续学习，适应不断变化的技术环境；
6. 团队合作：能够在团队中合作，共同推进项目进展；
7. 组织能力：需有良好的时间管理和资源协调能力；

2.5 自动化测试优缺点，列举5个自动化测试工具，简介1~3个

- 优点：
    1. 程序的回归测试更方便
    2. 可以进行更多更繁琐的测试
    3. 可以执行一些手工测试困难或不可能进行的测试
    4. 充分利用资源。
    5. 测试具有一致性和可重复性。
    6. 测试的复用性。
    7. 让产品更快面向市场。
    8. 增加软件信任度。
- 缺点
    1. 初始投资和维护成本可能较高
    2. 需要时间和培训来学习自动化工具
    3. 应用程序更新可能需要频繁更新测试脚本
    4. 对测试环境有特定要求
    5. 复杂应用可能难以自动化
- 5个自动化测试工具
    1. C++ Test
        - C++ Test是一个针对C/C++源代码进行自动化单元测试的工具。它可以自动测试任何C/C++函数、类或部件，减少编写测试用例、测试驱动程序或桩调用代码。C++ Test能够自动测试代码结构（白盒测试）、测试代码的功能性（黑盒测试）和维护代码的完整性（回归测试）。
    2. JUnit
        - JUnit是一个Java语言的单元测试框架，广泛用于开发期间的自动化测试。它简单易用，并与其他工具如Eclipse和IntelliJ IDEA集成良好。
    3. JMeter
        - JMeter是Apache提供的一款开源工具，用于性能测试和负载测试。它可以用来测试服务器、网络或对象的性能，并具有图形化用户界面。
    4. TestComplete
        - TestComplete是一个功能强大的桌面和移动应用自动化测试工具。它支持多种编程语言和操作系统，提供了一系列用于测试Web、桌面和移动应用程序的功能。
    5. Postman
        - Postman是一个流行的API开发工具，它提供了一个用户友好的界面来测试RESTful API。Postman支持测试脚本编写、环境变量配置和集合运行。

2.6 黑盒测试的优缺点

- 优点
    1. 简单，无需了解程序的内部代码实现。
    2. 从用户角出发，容易知道用户会使用到哪些功能，会遇到什么问题。
- 缺点
    1. 代码覆盖率较低，自动化测试覆盖率较低。

2.7 四大测试概念及目的

1. 单元测试
    - 检测最小的软件设计单元模块是否符合详细设计的要求，是否存在编码错误等，确保产生符合要求的、运行可靠的程序单元。
    - 单元测试是最低层次的测试，但确是最有效的测试，在性能价格比上最优。
2. 集成测试
    - 检测此前已经测试过的各个模块（单元）是否能够完好地结合在一起，是否在接口等方面存在错误，确保各单元（模块）以正确、稳定和一致的方式进行交互。
3. 系统测试
    - 测试已集成在一起的产品是否符合需求规格说明书的需求。
    - 主要验证系统的功能性需求和非功能性需求，为下一阶段的验收测试奠定基础。
4. 验收测试
    - 检测产品是否符合最终用户的要求，并在软件正式交货前确保系统能正常工作且可用。

2.8 系统测试中，α测试和β测试有什么区别?

- α测试：是在用户组织模拟软件系统的运行环境下的一种验收测试，由用户或第三方测试公司进行测试，模拟各类用户为对即将面世的软件产品进行测试，试图发现并修改错误。
- β测试：是用户公司组织各方面的典型终端用户在日常工作中实际使用beta版本，并要求用户报告异常情况，提出批评意见。
- 区别
    1. 测试的场所不同

       α测试是指把用户请到开发方的场所来测试，β测试是指在一个或多个用户的场所进行测试。
    2. 测试环境不同

       α测试的环境受开发方控制的用户的数量相对较少，时间比较集中。而β测试的环境是不受开发方控制的，谁也不知道用户如何折磨软件，用户数量相对较多，时间不集中。
    3. 测试周期不同

       一般地，α测试先于β测试执行。通用软件产品需要较大规模的β测试，测试周期较长。

2.9 面向对象的测试和传统测试的测试单元的不同?

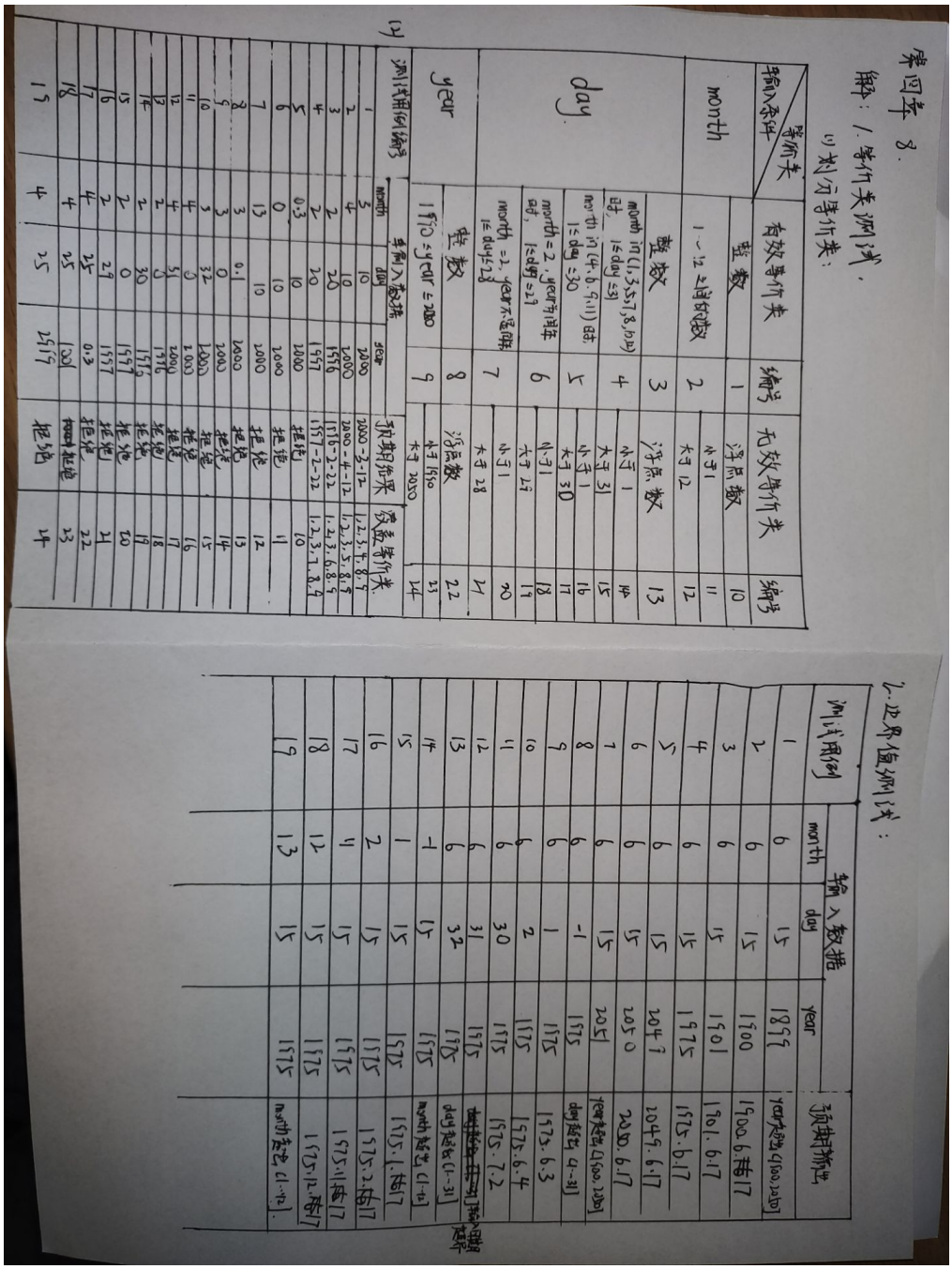
- 传统软件的基本构成单元为功能模块，每个功能模块一般能独立地完成一个特定的功能。
- 面向对象的软件中，基本单元是封装了数据和方法的类和对象。

3. 解答

3.1 黑盒测试

- 等价类划分
- 边界值测试

1. P75-8 程序有三个输入变量 month、 day、year(month、day 和 year均为整数值，并且满足：1<month<12, 1<day<31,1900<year<2050)，分别作为输入日期的月份、日、年份，通过程序可以输出该输入日期在日历上隔一天（第三天）的日期。例如，输入为 2005 年 11月29日，则该程序的输出为 2005年12月1日。请用等价类测试和边界测试法设计测试用例。

1. 

2. P75-14 假设商店货品价格(R)都不大于 100 元（且为整数），若顾客付款(P)在 100元内，现有一个程序能在每位顾客付款后给出找零钱的最佳组合（找给顾客货币张数最少）。假定此商店的货币面值只包括 50 元(N50)、10元(N10)、5元(N5)、1元(NI)四种。请结合等价类划分法和边界值分析法为上述程序设计出相应的测试用例。

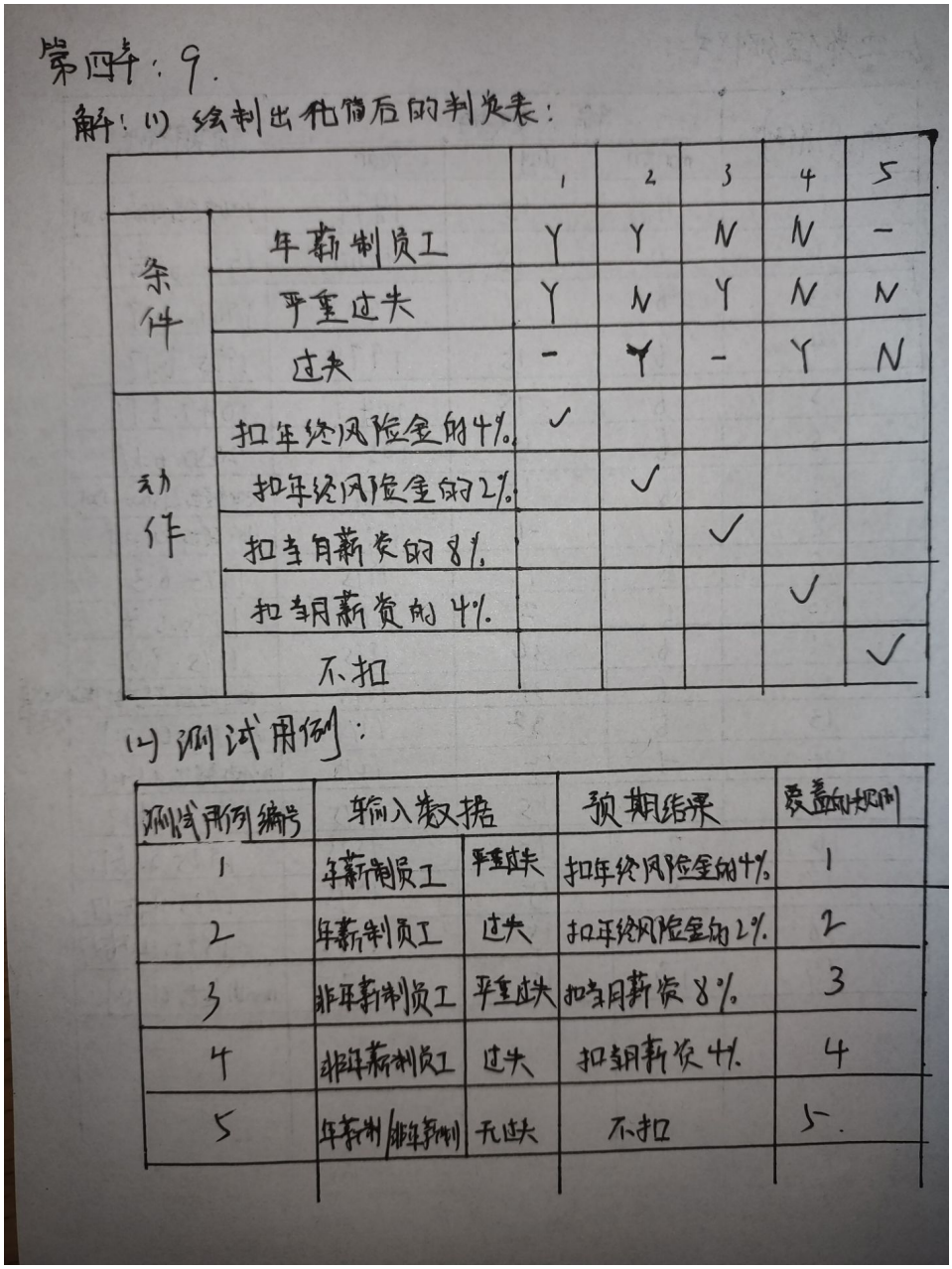
3. 等价类测试的优缺点

    - 优点：
        - **减少测试用例**：通过将输入数据分为有效等价类和无效等价类，可以减少需要测试的用例数量。

        - **覆盖关键场景**：确保了每个等价类至少被测试一次，覆盖了关键的输入场景。
        - **易于识别错误**：由于每个等价类代表一组相似的输入，如果发现错误，可以快速定位到问题所在。
    - 缺点：
        - **可能遗漏边界情况**：等价类测试可能不会覆盖到边界值或异常值，这些情况可能需要特别的关注。
        - **依赖需求文档**：如果需求文档不完整或不准确，等价类划分可能不全面。
4. 边界值测试的优缺点
    - 优点：
        - **发现边界错误**：边界值是程序逻辑中常见的错误点，边界值测试有助于发现这些错误。
        - **补充等价类测试**：可以作为等价类测试的补充，确保边界情况得到测试。
    - 缺点：
        - **测试用例可能较多**：边界值测试可能会生成更多的测试用例，特别是当输入域较大时。
        - **可能忽略非边界情况**：专注于边界值可能会忽略一些非边界但同样重要的输入情况。

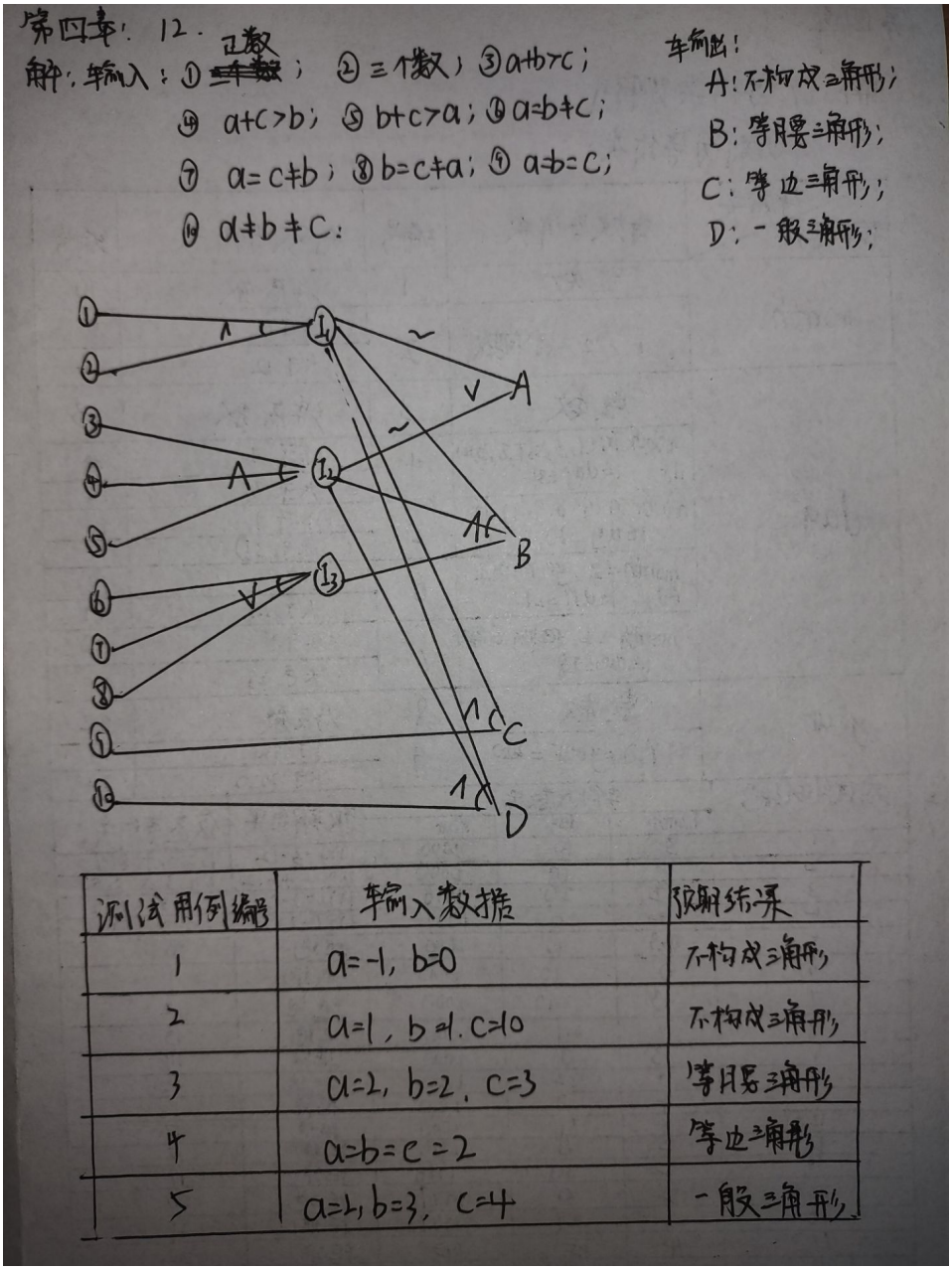
- 判定表测试

1. P75-9 某软件的一个模块的需求规格说明书中描述：
   (1) 年薪制员工：严重过失，扣年终风险金的4%；过失，扣年终风险金的2%。
   (2) 非年薪制员工：严重过失，扣当月薪资的8%；过失，扣当月薪资的4%。
   请绘制出判定表，并设计相应的测试用例。

1. 

2. P75-10 某公司折扣政策：年交易额在10万元以下的，无折扣；在10万元以上的并且近三个月无欠款的，折扣率 10%；在10万元以上，虽然近三个月有欠款，但是与公司交易在10年以上的，折扣率8%；在10万元以上，近三个月有欠款，且交易在10年以下的折扣率5%；请用判定表来描述该公司的折扣政策。

- 因果图测试
  1. P75-12 请使用因果图法为三角形问题设计测试用例。
     1.



2. 判定表测试的优缺点

- 优点：
  - **系统化**：通过表格形式系统化地展示了输入条件与预期结果之间的关系，便于理解和分析。
  - **覆盖全面**：可以确保所有可能的输入条件组合都被测试到，有助于发现组合逻辑错误。
  - **易于维护**：当需求变更时，判定表可以方便地进行更新和维护。
- 缺点：
  - **创建复杂**：对于具有大量条件和动作的系统，创建判定表可能非常复杂和耗时。
  - **可能冗余**：某些条件组合可能在实际中不会发生，但判定表测试可能会包含这些冗余情况。
  - **测试用例生成**：需要从判定表中正确地生成测试用例，这可能需要额外的分析和判断。

3. 因果图测试的优缺点

- 优点：
  - **图形化表示**：使用图形化的方法表示条件和结果，直观易懂。

        - **易于识别因果关系**：有助于识别和理解条件之间的因果关系。
        - **减少测试用例**：通过识别条件的独立性，可以减少不必要的测试用例。
    - 缺点：
        - **可能遗漏组合**：在转换因果图为判定表的过程中，可能会遗漏某些条件组合。
        - **难以处理复杂逻辑**：对于具有复杂逻辑和多个条件的系统，因果图可能难以表示和理解。
        - **转换过程可能出错**：将因果图转换为判定表或测试用例时，可能会出现错误，需要仔细检查。
- 正交表
    - 它是一种特殊的矩阵，用于系统地探索多个测试条件的组合。正交表的主要目的是减少需要测试的案例数量，同时确保测试覆盖了所有重要的条件组合。

3.2 白盒测试

- 基路径测试方法

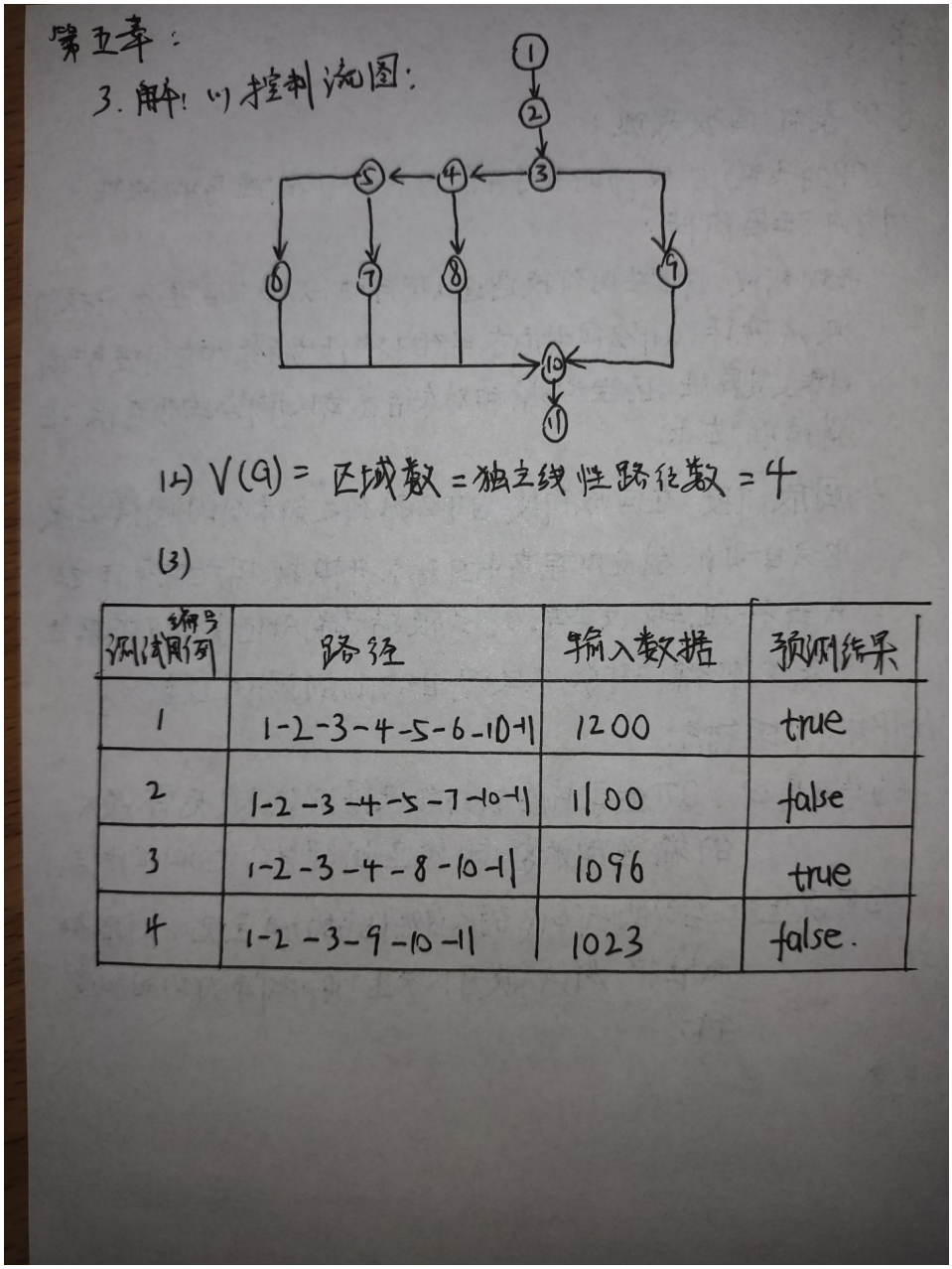
1. P106-3 以下代码由Java语言书写，用于判断闰年。请按要求回答问题

```java
    public boolean isLeap(int year) {
        boolean leap;
        if (year % 4 == 0) {
            if (year % 100 == 0) {
                if (year % 400 == 0) {
                    leap = true;
                } else {
                    leap = false;
                }
            } else {
                leap = true;
            }
        } else {
            leap = false;
        }
        return leap;
    }
```

（1） 请画出以上代码的控制流图。

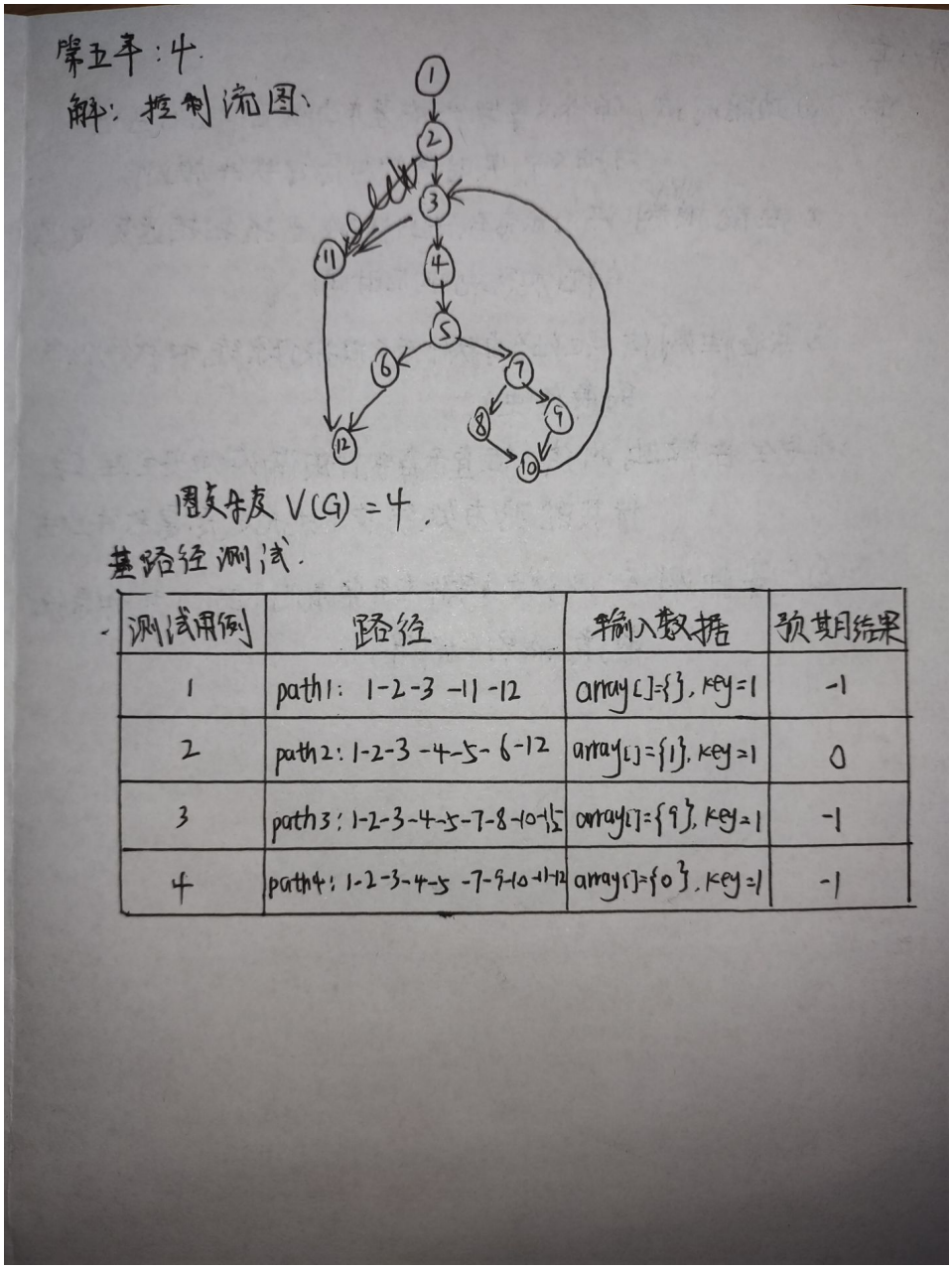
（2） 请计算上述控制流图的圈复杂度V(G)（独立线性路径数）。

（3） 假设输入的取值范围是 0 < year < 2010，请使用基路径测试法为变量year设计测试用例，使其满足基本路径覆盖的要求。



2. P106-4 请用逻辑覆盖和基路径测试方法对下面的Java代码进行测试。代码的功能是：用折半查找法在元素呈升序排列的数组中查找值为key的元素。

```java
public int binSearch(int array[], int key) {
    int mid, low, high;
    low = 0;
    high = array.length - 1;
    while (low <= high) {
        mid = (low + high) / 2;
        if (key == array[mid]) {
            return mid;
        } else if (key < array[mid]) {
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    return -1;
}
```



3. P107-6 请用逻辑覆盖和基路径测试方法对下面的Java代码进行测试。代码的功能是：输入三个数，判断它们是否为有效的日期（其中年 < 2050）

```
    public boolean isDate(int year, int month, int day) {
        boolean flag = true;
        if ((year < 0) || (year > 2050)) {
            flag = false;
        }
        if ((month < 1) || (month > 12)) {
            flag = false;
        } else {
            switch (month) {
                case 1:
                case 3:
                case 5:
                case 7:
                case 8:
                case 10:
                case 12:
                    if ((day > 31) || (day < 1)) {
                        flag = false;
```

```
                    }
                    break;
                case 4:
                case 6:
                case 9:
                case 11:
                    if ((day > 30) || (day < 1)) {
                        flag = false;
                    }
                    break;
                case 2:
                    if (isLeap(year)) {
                        if ((day > 29) || (day < 1)) {
                            flag = false;
                        }
                    } else {
                        if ((day > 28) || (day < 1)) {
                            flag = false;
                        }
                    }
                    break;
                default:
                    break;
            }
        }
        return flag;
    }
```

- 数据流测试
  - 变量的定义和使用

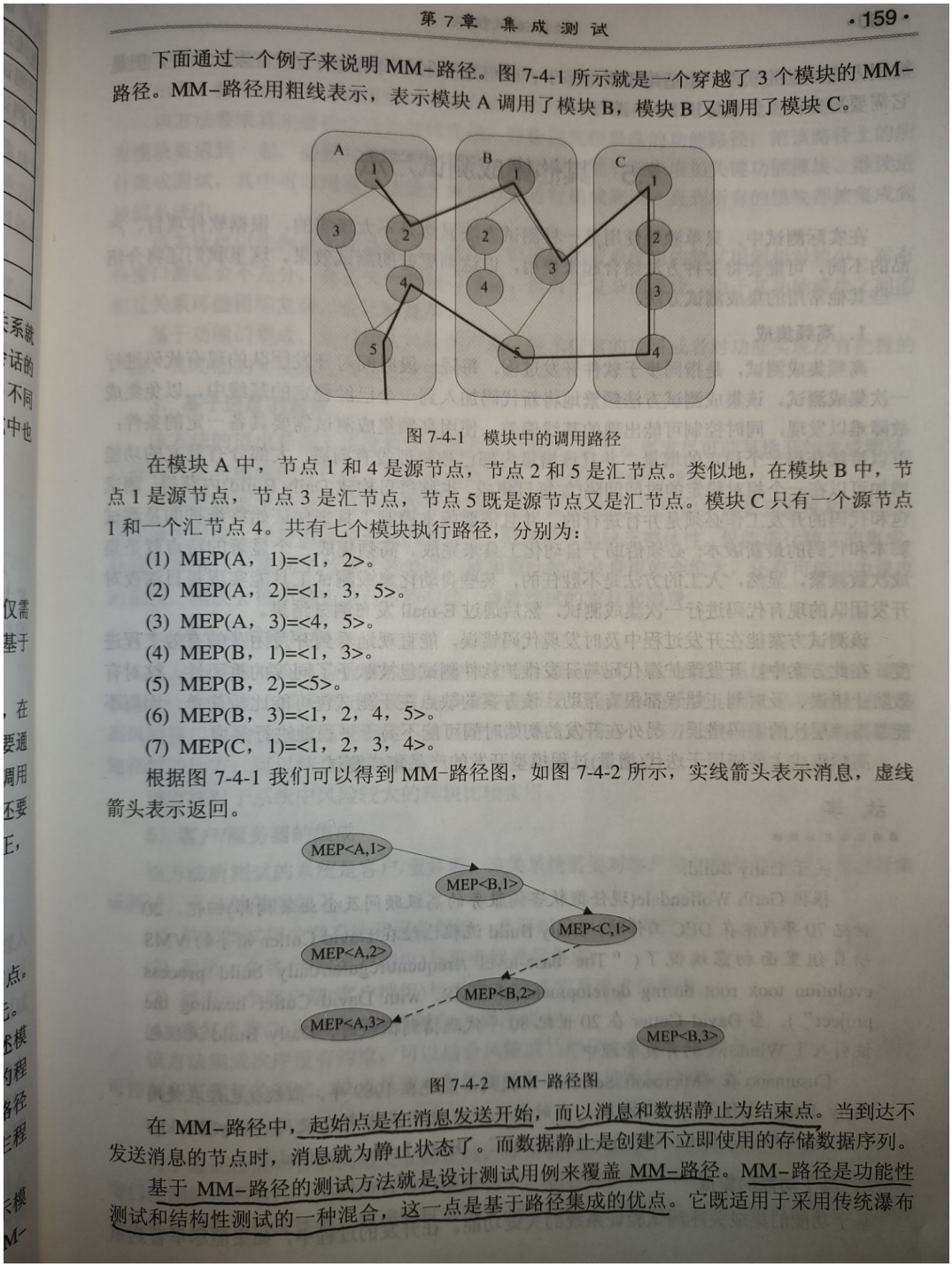
3.3 单元测试

- 会写桩、驱动、测试脚本程序

3.4 集成测试

- M-M路径测试，MEP构成M-M测试
  - 基于MM路径（Main Menu Path）的集成测试是一种软件测试方法，通常用于测试软件系统中的多个组件或模块在协同工作时的交互和功能。MM路径指的是从主菜单（Main Menu）开始，用户通过选择不同的菜单项进行操作，形成的操作路径。
  - 基于MM路径的集成测试通常在软件开发的后期进行，目的是在软件发布前确保所有功能模块能够无缝集成，提供稳定、可靠的用户体验。这种测试可以手动执行，也可以通过自动化测试工具来实现。
  - 这种测试方法主要关注以下几个方面：
    - **用户交互流程**：测试用户通过主菜单进行操作时，各个菜单项之间的导航是否流畅，用户界面是否友好。
    - **功能集成**：确保各个功能模块在集成后能够正常工作，没有功能冲突或数据不一致的问题。
    - **数据流**：检查数据在不同模块间传递时的准确性和完整性。

- **异常处理**：测试系统在遇到错误或异常输入时的响应和恢复能力。
- **性能测试**：评估系统在处理多个模块交互时的性能，包括响应时间和资源消耗。
- **安全性测试**：确保在用户通过主菜单进行操作时，系统的安全性没有被破坏，例如权限控制是否得当。
- **用户体验**：评估用户在使用集成后的系统时的整体体验，包括易用性、可访问性和满意度。



第7章 集成测试 ·159·

下面通过一个例子来说明 MM-路径。图 7-4-1 所示就是一个穿越了 3 个模块的 MM-路径。MM-路径用粗线表示，表示模块 A 调用了模块 B，模块 B 又调用了模块 C。

图 7-4-1 模块中的调用路径

在模块 A 中，节点 1 和 4 是源节点，节点 2 和 5 是汇节点。类似地，在模块 B 中，节点 1 是源节点，节点 3 是汇节点，节点 5 既是源节点又是汇节点。模块 C 只有一个源节点 1 和一个汇节点 4。共有七个模块执行路径，分别为：

(1) MEP(A，1)=<1，2>。
(2) MEP(A，2)=<1，3，5>。
(3) MEP(A，3)=<4，5>。
(4) MEP(B，1)=<1，3>。
(5) MEP(B，2)=<5>。
(6) MEP(B，3)=<1，2，4，5>。
(7) MEP(C，1)=<1，2，3，4>。

根据图 7-4-1 我们可以得到 MM-路径图，如图 7-4-2 所示，实线箭头表示消息，虚线箭头表示返回。

图 7-4-2 MM-路径图

在 MM-路径中，起始点是在消息发送开始，而以消息和数据静止为结束点。当到达不发送消息的节点时，消息就为静止状态了。而数据静止是创建不立即使用的存储数据序列。

基于 MM-路径的测试方法就是设计测试用例来覆盖 MM-路径。MM-路径是功能性测试和结构性测试的一种混合，这一点是基于路径集成的优点。它既适用于采用传统瀑布

3.5 系统测试

1. P207-1 针对某杀毒软件（如瑞星），考虑其需要测试的内容。
    1. **功能测试**：确保杀毒软件的基本功能正常运行，包括扫描文件、实时保护和恶意软件检测；
    2. **性能测试**：评估杀毒软件的性能，包括扫描速度、资源消耗和系统响应时间；
    3. **兼容性测试**：验证杀毒软件与不同操作系统和软件版本的兼容性；
    4. **安全性测试**：检查杀毒软件的漏洞和安全性，确保其能够有效防护系统免受恶意软件攻击；
    5. **可靠性测试**：测试杀毒软件在异常情况下的行为，如系统崩溃、网络中断等。
2. P207-2 针对某论坛（如www.tianya.cn），考虑其需要测试的内容。

1. **功能测试**：确保主要功能（注册、登录、发帖、回帖等）正常运作；
2. **性能测试**：评估网站的加载速度和响应时间，尤其是主页、帖子页面和搜索功能；
3. **安全性测试**：检查网站对常见攻击（如SQL注入、XSS攻击）的防护能力，确保用户数据安全；
4. **兼容性测试**：验证网站在不同操作系统和浏览器下的兼容性，以及在不同设备上的显示效果；
5. **用户体验测试**：测试网站的用户界面设计、导航结构和搜索功能，确保用户能够轻松使用论坛。

3.6 自动化测试

1. P254-5 简述QTP的录制/回放原理。分析QTY脚本的特点。

3.7 考察方式

- 对一个案例用不同的方法测试，再回答哪种方法更好，为什么