软件测试第一次作业

问题 I：正确理解原型方法对软件生命周期不同阶段的支持，分别给出：辅助或代替分析阶段；辅助设计阶段；代替分析与设计阶段；代替分析、设计和实现阶段；代替全部开发阶段所对应的开发活动执行时间顺序。

解答：

阶段	开发活动执行时间顺序
辅助或代替分析阶段	初步需求、分析、原型过程、需求说明、设计、设计说明、编码、程序系统、编码、软件产品 、运行维护
辅助设计阶段	初步需求、分析、需求说明、设计、原型过程、设计说明、编码、程序系统、编码、软件产品 、运行维护
代替分析与设计阶段	初步需求、分析、原型过程、设计说明、编码、程序系统、编码、软件产品 、运行维护
代替分析、设计和实现阶段	初步需求、分析、原型过程、程序系统、编码、软件产品 、运行维护
代替全部开发阶段	初步需求、分析、原型过程、软件产品 、运行维护

软件测试第二次作业

问题：在敏捷宣言遵循的12条原则中挑选1条你感兴趣的原则进行风险评估

解答：

对敏捷宣言中的第四条原则：

> **在整个项目过程中，业务人员与开发人员必须日常一起工作。**

进行风险评估。

- **需求变更风险**

  在一个现代软件项目中，需求的变更难以避免。项目实际开发过程中，业务人员与开发人员一同工作可以做到及时地发现需求变更，处理需求变更，减少客户需求随时变化对工程的影响。业务人员与客户进行沟通， 并及时将需求变更反映给开发人员，开发人员更新文档，调整开发，避免因需求变更的不及时传播而做无用功。

- **进度风险**

  业务人员与开发人员一起工作，积极并及时处理需求变更，不断修改、完善软件，开发计划频繁调整，将导致软件项目开发进度的不确定性和不可评估性。

- **预算风险**

  需求的频繁变更导致开发时间的延长，成本随开发时间而不断累积，经费一旦不足将导致项目的失败。业务人员与开发人员工作，及时了解开发进度，及时向客户反应开发进度，并争取更多的预算，减轻预算风险。

- **管理能力风险**

  开发人员和业务人员一同工作将增加管理难度，与此同时，随着开发的进行，软件的开发、测试也愈加复杂，管理难度增加，管理能力风险上升。

- **信息安全风险**

  需求变更导致频繁的软件调整，设计、开发、测试过程产生不一致性，影响安全模块的设计和软件的测试，导致产生更多的软件漏洞，信息安全风险升高。客户的重要数据可能会随之泄露。

- **应用技术风险**

  软件实现过程中的需求变更，可能会产生因当前技术栈无法处理的风险，导致必须更换应用技术，重新配置环境等，对软件开发产生重大影响。同时，过长的开发周期可能会面临外界技术的革新，导致软件产品不再具有优势。

- **质量控制风险**

  实现过程中的需求变更会导致软件质量控制的疏忽，软件测试滞后，软件产品质量难以保障。

- **软件设计与开发工具风险**

  对需求变更的响应可能会对软件设计产生大的影响，开发工具可能难以满足新需求的要求，导致需要更新或寻求新的开发工具，不仅增加了成本上的风险，还增加了时间上的风险。

- **员工技能风险**

  业务人员与开发人员一同工作，需要业务人员熟练掌握开发人员的语言，对业务人员的要求也更高。同时随着需求变更导致的开发时间的延长，开发愈发复杂，对业务和开发人员的要求也越来越高。

- **人力资源风险**

  业务人员和开发人员都是软件开发中重要的人力资源。开发周期中，员工的离职和流动会严重影响开发过程，新的人力资源需要时间来熟悉整个开发计划，并需要时间来磨合，人力资源流失直接导致开发周期进一步延长，使得项目难以准时交付。

- **政策风险**

  需求变更导致开发时间增长，社会、内部政策的变动将对软件开发产生或轻或重的影响。

- **市场风险**

  需求变更导致开发时间增长，市场的变化将进一步影响需求，软件开发过程需要了解市场的变化，提前或及时做出调整。

- **营销风险**

  好的营销的效果吸引投资，预算有保证，失败的营销降低客户的信心和预算投入，严重导致软件开发的停滞。营销提高用户对软件产品的期望，对高质量软件产品的需求对开发人员提出新的挑战。

软件测试第三次作业

问题：

1. **综述软件测试的16条公理**
2. **结合你所熟悉的一套软件，针对上述公理表述你的见解**

回答：

1. 软件测试的十六条公理由Paul Gerrard于2008年提出，它是软件测试的上下文无关规则（Context-neutral or Context-free），由4条利益相关者公理、6条测试设计公理和6条测试执行与交付公理
2. 以一个微信点单小程序为例，准对上述十六条公理进行分析。

1. 第一条：测试需要利益相关者。作为一个软件产品，其利益相关者包括：产品经理、商铺相关业务人员、使用该小程序的顾客等。它们分别代表开发方、服务方、消费者的利益。在不同阶段的测试过程中，不同利益相关者的参与程度不同，例如在产品Beta测试阶段，需要消费者参与测试。
2. 第二条：测试的价值是为利益相关者提供决策依据。对于利益相关者，不仅需要考虑开发时间和开发成本，还需要考虑测试的结果和对应的价值，来作为放弃开发或者接受产品交付的决定。例如，服务商对页面的切换效果和支付延迟要求比较严格，测试员针对这些模块的测试对利益相关方来说就更有价值。
3. 第三条：如果我们不约束测试的范围，我们永远无法符合利益相关者的期望。测试人员需要与利益相关者进行协商，以划分测试人员的管理范围，例如要了解有哪些模块和系统的哪些范围需要进行测试，而那些模块由于不可抗拒因素不可测试。
4. 第四条：测试和验收的范围始终是妥协的结果。测试人员需要与利益相关方进行协商，双方需要在一定情境下做出妥协，例如对一个功能测试的完善程度，考虑成本与时间和安排的议程。同时，双方还需要协商讨论出接受系统的标准，来节省双方在开发过程中争论所消耗的时间。
5. 第五条：测试设计基于模型。对于一个点单小程序，针对其处理交易、库存、客服等等部分可能存在匹配的测试模型，开发人员会考虑使用这些模型的组合来进行软件项目的测试。但该软件项目可能会存在一些特殊的模块或加入了一些全新的功能，不满足现成模型的假设要求，故需要对模型进行评估和调整。
6. 第六条：测试人员需要知识来源以选择测试内容。测试人员需了解消费者的行为和点餐规则和商家制定的销售规则来选择、进行测试。例如点单杯数不能小于零，折扣仅限每周末细节等等。
7. 第七条：测试人员需要知识来源以评价被测对象的实际产出或行为。与上一条一样，测试人员需要了解，并且是准确地了解软件产品投放的行业的领域规则，来评价测试的结果是否符预期。
8. 第八条：测试需要一个或多个覆盖模型。在分析第五条时，我们了解到测试人员会考虑多个测试模型，本条公理意味着测试人员需要保证，所选用的模型集合的并集能够准确地覆盖所有测试需求，否则会出现缺漏。
9. 第九条：测试需要一种机制来排序测试优先级。测试覆盖软件产品的方方面面，在这之中，对不同模块的测试应有不同的优先级，不同模块的可能会存在耦合和依赖，使得被依赖的模块的测试优先级高于依赖该模块的其他模块，同时，需求方越重视的模块往往有越高的测试优先级。
10. 第十条：测试的知识来源是模糊且不完整的。测试所使用的判准的来源可能不够可信，即测试的上游所提供的资料未必完善，例如一份不完善或表达模糊的折扣机制，会影响测试的结果。
11. 第十一条：通过利益相关者做决策时的信心来衡量测试的价值高低。一个测试再这整个设计开发过程中越重要，测试通过会增加需求方的信心。需求方在做出决策时必然会考虑种种因素，一个让然需求方做出有底气的决定的测试往往越重要。
12. 第十二条：一些重复测试是不可避免的。重复的测试可能出现在需求的变化或代码的改动后，以往通过的测试已随代码的改变而不再有力，故必须重新进行测试来保证改动后没有产生新的问题。例如需求方要求加入会员折扣制度，在新的代码编写后需要对支付模块重新测试。
13. 第十三条：先执行最有价值的测试，否则可能没有时间执行它们。测试不能同时进行，测试也需要时间。开发过程紧张的，根据测试的重要程度来安排测试的顺序，优先测试重要的模块，能降低软件开发的风险。
14. 第十四条：测试执行需要明确、可控的测试环境。在测试的过程中，测试模块的切换，修改会导致测试环境的变更，因此划分出一个测试环境的并集很重要。对于点餐小程序：这个环境的并集就包含测试交易的环境，测试网络的环境·，测试UI效果的环境等等。
15. 第十五条：测试永远不会按计划进展，测试所得到的证据按照离散量子化的模式出现。需要承认测试结果并不总是符合要求的，测试过程也会因为一些因素而中断延期，例如点餐后台服务端的测试逻辑不合理，需要重新设计测试。陆陆续续的测试得到的便是不连续的证据，对这些离散的测试数据进行整理就显得尤为重要。
16. 第十六条：测试永远不会结束，只会停止。需求方和测试方，设计编码人员和测试人员都需要在交流与妥协中前进，在达到一个限度时结束测试，否则测试就会影响开发人员别的模块的开

发，也会延长软件的交付时间。比如界面效果的测试，如果一味追求完美，效果的提升并不会随着成本的增长而显著提高，盲目的坚持测试和修改反而会影响交付。

软件测试第四次作业

问题：

选择一个测试要素 (Lec.10, slide 9)，以你实现的或者熟悉的一个软件作品为例，分别讨论该要素在软件生命周期的需求、设计、编程、测试、安装、验收和维护各阶段的测试目标和内容

回答：

在基于软件生命周期的软件测试方法中，考虑十五项测试要素和六个SDLC阶段。十五个测试要素分别是：可靠性、授权、文件完整性、进程追踪、系统运行的连续性、服务级别、存取控制、方法论、正确性、一致性、易用性、可维护性、可移植性、耦合性、性能和易操作性。软件生命周期中的六个阶段依次为：需求分析阶段、设计阶段、编程实现阶段、测试阶段、安装和验收阶段、维护阶段。

一个测试要素由若干个测试事件组成，用于验证该测试要素所描述的测试目标是否已经达成。一个测试事件描述了测试条件和可能发生的事件。不同的测试要素在SDLC的不同阶段有着不同的测试内容。

其中，正确性是指**数据输入、过程处理和输出的正确性。** 下面选择"**正确性**"来作为要讨论的设计要素，结合软件工程**初级实训所开发的C++项目Agenda**，说明"正确性"在软件生命周期的需求、设计、编程、测试、安装、验收和维护各阶段的测试目标和内容。

Agenda简介

Agenda是一个命令行界面的议程管理系统。系统提供用户登录，新用户注册，已注册用户登陆后可以注销 `delete` 当前用户账户，查询 `query` 用户名单，也可以添加 `add`、删除 `delete`、查询 `query` 系统中记录的会议安排等管理功能。

需求阶段

- "正确性"在需求阶段的测试目标有：
- 需求分析足够正确充分地反应了客户的需求；
- 需求文档、功能规格说明和产品规格说明正确完整。
- "正确性"在需求阶段的测试内容为：
- 定义功能规格说明，审查需求分析文档、规格说明书，确认定义符合要求。在这个测试过程中识别错误的需求，对测试中出现的问题应给出合理解释或正确有效的处理，从而减少后续追踪需求错误上的开销，规避因错误需求导致的系统错误风险；
- 分析测试要素，是否定义、预估了性能标准、差错的容忍度、存储控制的方式等，它们与"正确性"都是分不开的，因为这些都将作为衡量输入输出是否正确的基础；
- 验证活动：确定验证的方法、确定需求的充分程度、生成功能测试数据、确定与需求符合的设计；
- 总而言之，就是要对需求阶段的各类制品（主要是文档）进行测试，识别错误的需求和错误的判准。
- 结合Agenda项目的需求文档，可以给出以下例子：
- 例如Agenda的用户查询功能：用户的信息包括用户名、密码、邮箱和电话信息，而查询功能可以查看已注册该系统的所有用户的用户名、邮箱及电话信息。显然，查看用户的密码是错误的需求，如文档中出现"用户可以查看已注册该系统的所有用户的所有信息"，则必须修改为"用户可以查看已注册该系统的所有用户的用户名、邮箱及电话信息（不包括密码）"。（当然也不会出现如此低级的错误）；

- 再如Agenda的退出：程序的退出并保存是一个必要的功能。如果在需求分析中遗漏此功能，整个程序的执行流程就是不完整的，会导致数据写回硬盘产生错误，因而需要对需求进行测试来查改这些问题。

设计阶段

- “正确性”在设计阶段的测试目标有：
- 系统设计文档、程序设计流程图、数据流图正确无误
- 设计与需求保持一致，没有差错
- “正确性”在设计阶段的测试内容为：
- 在设计阶段，我们对需求进行细化会产生很多制品，包括UML类图、系统流程图、数据流图、输入输出说明、过程说明、文件说明、控制说明、硬件和软件的需求说明、操作手册说明书、数据字典等等。测试需要检查这一系列文档是否符合设计需求，在表达上清晰无误；
- 因为产生诸多文档，需要阐述测试方法和测试评估准则，编写测试计划，成立测试小组，安排具有里程碑的测试日程，测试过程不能出现遗漏；
- 设计阶段会考虑环境和开发工具，需要对开发工具进行测试，测试其可行性；
- 检查设计中遗漏的情况、错误的逻辑、不匹配的模块接口、不合理的数据结构、错误的 I/O 假定、不够充分的用户界面等等；
- 进行设计阶段的评审。
- 结合Agenda项目的需求文档，可以给出以下例子：
- 检查 Agenda 的UML类图：判断给出的函数参数、返回值是否无误，各个模块接口是否匹配；
- 设计使用三层构架，需要测试对各个类的划分是否合理，例如Date类、User类应该放在数据访问层，AgendaUI应该放在表现层。对类错误的分层会破坏整个架构；
- 对于各个类交互的基本逻辑是否合理，考虑User类，Meeting类是没有直接耦合的；AgendaService类只包含Storage类的调用，AgendaUI类只能操作AgendaService类的功能接口。如哦AgendaUI类直接访问数据访问层的类即可判断设计不正确；
- 检查AgendaUI类的流程图，用户界面很容易出现逻辑错误，输入非法的命令命令行可能导致程序异常退出。

编程阶段

- “正确性”在编程阶段的测试目标有：检查出代码的编译错误、运行错误和逻辑错误，单元测试各个方法判断输出是否正确。
- “正确性”在编程阶段的测试内容为：程序符合设计
- 测试编码与设计是否一致，代码是否正确实现了系统的规格说明；
- 审查代码：检查代码是否遵守编程规范，测试代码注释是否正确，可采用编码走查和检查、静态分析和动态测试等技术；
- 单元测试：为各个单元编写测试模块，测试各个单元是否能正常编译运行，测试输出结果是否满足要求
- 结合Agenda项目的需求文档，可以给出以下例子：
- 首先要对各个类及逆行单元检测，测试内容需要覆盖代码的各个路径，对于Date类，需要编写一个DateTester类测试日期的格式是否正确。测试属于白盒测试，测试输入包含非法日期（如2021年2月29日、2021年11月0日等等），非法格式（默认格式为`yyyy-mm-dd/hh:mm`）。测试Date类的运算符重载，例如Date类的等号，大于小于号是否满足要求；
- 在Agenda类中Storage类负责控制文件的读写，文件的读写是测试的重点，需要测试文件不存在、路径错误等情况程序对异常的处理是否正确，还需要测试文件读写是否会出现一些意料之外的错误，比如user和meeting文件的格式都是csv格式（用逗号划分），那么如名称、邮箱等出现逗号是否会影响文件的读写；
- Meeting类创建会议的时间安排逻辑比较复杂：创建会议时，不允许会议时间重合，即一个用户某场会议的开始时间和结束时间之间不能有别的会议，但是允许允许会议开始时间是另一个会议的结

束时间，即一个会议结束后马上开始另一个会议。对于这个会议的测试需要专门编写MeetingTest类来进行严格的测试，测试集必须覆盖条件分支的每一种情况；
- 代码得编写是否严格按照UML类图，例如UML中为`+`的是公共方法/属性、`-`的是私用方法/属性，在编写时是否遗漏了`private`、`public`的指定；方法传参是否忘记了引用符号，指针类型是否使用正确，这些都是要终点审查的细节。

测试阶段

- “正确性”在测试阶段的测试目标有：测试软件在不同环境和输入下，它的输入是否都是正确的。
- “正确性”在测试阶段的测试内容为：
- 白盒测试：包括了上一阶段的单元测试，为多个类建立小模块来进行测试；
- 黑盒测试：主要用于测试整个系统运行起来的效果。测试的输入应该包含合法的和非法的输入，甚至应包含极端的输入；测试界面显示是否有差错；
- 测试阶段分析缺陷：是否出现了缺陷，识别缺陷的类型、级别和原因；
- 进行第三方的正式确认测试：检验系统是否按照用户的要求运行；用户能够成功地安装一个新的应用系统来进行测试；使用测试工具进行自动化的测试。
- 结合Agenda项目的需求文档，可以给出以下例子：
- 使用 Google Test 来进行单元测试，使用SonarQube进行代码质量评估；
- 界面测试：测试界面是否满足需求，Agenda是一个命令行系统，故界面要求不高，但是还是有输出细节需要处理，例如要保证每一个需要换行的地方能够正确换行，检查错别字也是测试的内容；
- 文件读写测试：再一次测试文件读写，因为Agenda采用的是启动时将数据写入内存，修改在内存，只有在关闭时才会写入磁盘，因此异常关闭会导致新写入的数据清空，需要测试判断是否存在更恶劣的情况，例如在读入内存的过程中异常退出是否会影响原本的数据内容；
- 安全性的测试：用户的密码等隐私数据是否能已加密的方式存储在文件中，评估加密方式的可靠性。

安装阶段

- “正确性”在安装阶段的测试目标有：测试软件是否能正确安装。
- “正确性”在安装阶段的测试内容为：
- 正确的程序和数据的加入，充足的的内存空间和满足要求的的环境；
- 安装文件是否打包正常，安装文档和操作手册是否清晰无二义性。
- 结合Agenda项目的需求文档，可以给出以下例子：
- 在不同的环境中部署Agenda，检查安装情况，并对安装失败的情况进行分析整合（例如可能是CPP版本过低、文件夹权限不足等等）；
- 为安装失败的情况提供解决方案。

验收阶段

- “正确性”在验收阶段的测试目标有：测试验收文档和程序
- “正确性”在验收阶段的测试内容为：
- 检查编写验收计划、项目描述、用户职责、行政流程、验收活动描述等文档是否编写正确。
- 结合Agenda项目的需求文档，可以给出以下例子：
- 在不同的环境中部署Agenda，检查安装情况，并对安装失败的情况进行分析整合（例如可能是CPP版本过低、文件夹权限不足等等）；
- 为安装失败的情况提供解决方案。

维护阶段

- ”正确性”在维护阶段的测试目标有：测试维护阶段软件的正确性，为程序扩展提供正确性测试。
- “正确性”在维护阶段的测试内容为：
- 接收反馈修改需求，打补丁或升级，对提供的新功能进行测试；对于错误反馈进行修改，对于建议反馈加入讨论；
- 系统重新测试以确定改变的部分和未改变的部分能够继续工作；
- 需要对在原来的测试文档的基础上进行修改，进行新的测试。
- 结合Agenda项目的需求文档，可以给出以下例子：
- 对于Agenda基本实现结束后，我们在其基础上加入了若干扩展，例如日志功能、异常检测功能等。对于加入的新功能，我们需要按照需求—设计—编码—测试的流程，来确保新功能不会引入新的错误和异常；
- 例如对于日志系统，我们需要分析并设计日志的格式、文件读写方式、日志记录的实际，然后进行编码，进行白盒测试，在和源程序放在一起执行黑盒测试，测试通过后即可打包，依次进入安装、验收阶段，更新相关文档后完成新功能的加入。

软件测试第五次作业

问题

计算下列代码片段的 Halstead 复杂度的11项内容：

```
if (month < 3) {
    month = month + 12;
    year = year - 1;
}
return dayray((int)(day + (month + 1) * 26/10 + year +
        year / 4 + 6 * (year/100) + year / 400) % 7);
```

回答

- Halstead 复杂度根据程序源代码中语句行的操作符和操作数的数量计算程序复杂性。
    - 程序源代码中操作符和操作数的量越大，程序难度就越大。
    - **操作符**统计范围通常包括**语言保留字**、**函数调用**、**运算符**，也可以包括有关的**分隔符**（在C语言中，分隔符有括号、花括号、逗号、空白符、分号和冒号）等。
    - **操作数**统计范围可以是**常量**和**变量**等标识符。
- **注：在以下的计算中，我们不考虑任何分隔符，即在操作符的统计时忽略分隔符 。**

首先计算 Halstead 复杂度度量：

1. 程序中不同的操作符的个数：$n_1 = 11$，它们分别是：
    - 语言保留字：`if`、`return`、`int`；
    - 函数调用：`dayray`；
    - 运算符：`<`、`=`、`+`、`-`、`*`、`/`、`%`。
2. 程序中不同操作数的个数：$n_2 = 13$ ，它们分别是：
    - 常量：`3`、`12`、`1`、`26`、`10`、`4`、`6`、`100`、`400`、`7`；
    - 变量：`month`、`year`、`day`。
3. 程序中出现的操作符总数：$N_1 = 21$;
4. 程序中出现的操作数总数：$N_2 = 22$;

以下是 Halstead 复杂度的11项内容计算结果：

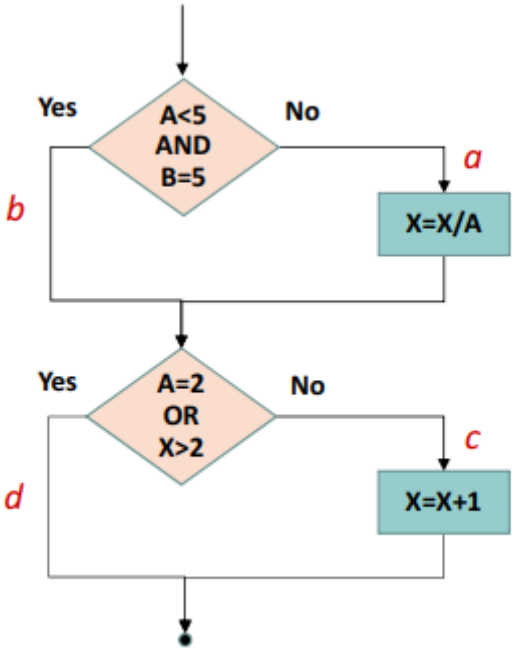
1. 程序词汇表总长度：$n = n_1 + n_2 = 24$；
2. 程序长度：$N = N_1 + N_2 = 43$；
3. 程序的预测长度：$N^{\wedge} = n_1 \log_2 n_1 + n_2 \log_2 n_2 \approx 83$；
4. 程序的体积：$V = N \log_2(n) \approx 211$；
5. 程序的级别：$L^{\wedge} = (2/n_1) \times (n_2/N_2) \approx 0.107$；
6. 程序的难度：$D = 1/L^{\wedge} \approx 9.308$
7. 编程工作量：$E = V \times D \approx 1964$；
8. 语言级别：$L' = L^{\wedge} \times L^{\wedge} \times V \approx 2.4$
9. 编程时间：$T^{\wedge} = E/(S \times f) \approx 0.03$（这里$S = 60 \times 60, f = 18$）；
10. 平均语句大小：$N/\text{语句数} = 10.75$（这里语句数为4，即代码第1、2、3、5行）
11. 程序中错误数预测值：$B = V/3000 \approx 0.07$。

软件测试第六次作业

问题

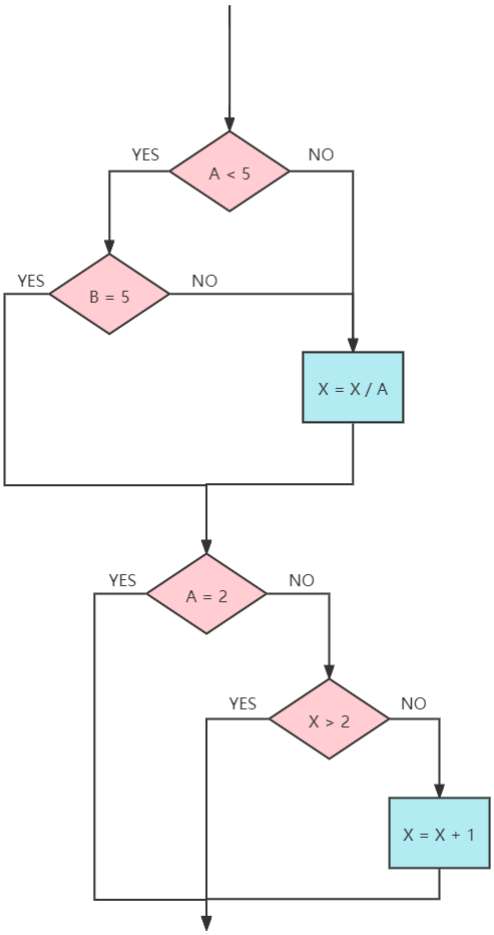
根据下侧的程序流程图，完成：

1. **转换单条件判定结构；**
2. **画出相应的程序控制流图；**
3. **给出控制流图的邻接矩阵；**
4. **计算 McCabe 环形复杂度；**
5. **找出程序的一个独立路径集合。**

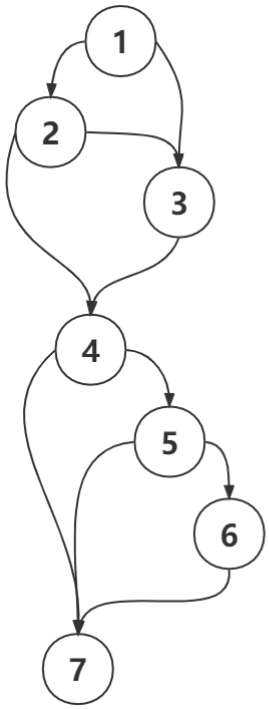


回答

1. **转换单条件判定结构：**



2. **画出相应的程序控制流图：**



1: A < 5
2: B = 5
3: X = X / A
4: A = 2
5: X > 2
6: X = X+1
7: end

3. **给出控制流图的邻接矩阵：**

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4. **计算 McCabe 环形复杂度：**McCabe 环路复杂度为程序逻辑复杂性提供定量测度。该度量用于计算程序的基本独立路径数目，也即是确保所有语句至少执行一次的起码测试数量。

   上述程序控制流图有7个结点和10条边，有环形复杂度如下

   $$V(G) = m - n + 2 = 10 - 7 + 2 = 5$$

5. **找出程序的一个独立路径集合：** 独立路径指至少沿一条新的边移动的路径。以下是一组独立的基本路径

编号	独立路径
1	1 - 2 - 4 - 7
2	1 - 3 - 4 - 7
3	1 - 2 - 3 - 4 - 7
4	1 - 2 - 4 - 5 - 7
5	1 - 2 - 4 - 5 - 6 - 7

软件测试第七次作业

问题

构造下述三角形问题的**弱健壮**的**等价类测试用例。**

- **三角形问题**：输入三个不超过100的正整数作为三角形的三条边，判断三角形是等边三角形、等腰不等边三角形、完全不等边三角形还是不能构成三角形。

回答

相关概念

- **等价类划分**是一种典型的**黑盒测试方法**，其完全不考虑被测程序的内部结构，只依据程序的规格说明来设计测试用例。它把所有可能的输入数据划分成若干部分，然后从每一部分中选取少数有代表性的数据做为测试用例。
- 等价类划分有两种不同的情况：
  - **有效等价类**：对于程序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**：对于程序规格说明来说是不合理的、无意义的输入数据构成的集合。
- **弱/强**、**一般/健壮**的等价类测试分类：
  - 弱等价类测试：针对单缺陷的等价类测试用例设计。

- 强等价类测试：针对多缺陷的等价类测试用例设计。
- 一般等价类测试：只覆盖有效等价类的测试用例设计。
- 健壮等价类测试：同时覆盖有效等价类和无效等价类的测试用例设计。

构造测试用例

令三角形三条边边长分别是：a, b, c，四种有效输出分别是三角形为等边三角形、等腰不等边三角形、完全不等边三角形和不能构成三角形。

从而得到四个**有效等价类**，**满足**a, b, c**都是不超过100的正整数**$(a, b, c \in Z \cap (0, 100])$:

- $R1 = \{< a, b, c >: 三边分别为a, b, c的三角形是等边三角形\}$
- $R2 = \{< a, b, c >: 三边分别为a, b, c的三角形是等腰不等边三角形\}$
- $R3 = \{< a, b, c >: 三边分别为a, b, c的三角形是完全不等边三角形\}$
- $R4 = \{< a, b, c >: 三边a, b, c不能构成三角形\}$

分别对这四个有效等价类构造构造测试用例，由于各个等价类互补重合，故构造四个测试案例分别对应到每一个有效等价类。

测试用例编号	a	b	c	预期输出
WN1	50	50	50	等边三角形
WN2	40	40	60	等腰不等边三角形
WN3	40	50	60	完全不等边三角形
WN4	10	20	80	不能构成三角形

再考虑若九个无效等价类，分别针对一种单缺陷：包括$a \leq 0, b \leq 0, c \leq 0$，
$a > 100, b > 100, c > 100$，$a, b, c$**非整数**共九种情况，对九个无效等价类构造测试用例：

测试用例编号	a	b	c	预期输出
WR1	**0**	50	50	$a \leq 0$，不在合法范围中
WR2	**200**	50	50	$a > 100$，不在合法范围中
WR3	**aaa**	50	50	a 非整数，不在合法范围中
WR4	50	**0**	50	$b \leq 0$，不在合法范围中
WR5	50	**200**	50	$b > 100$，不在合法范围中
WR6	50	**b5**	50	b 非整数，不在合法范围中
WR7	50	50	**0**	$c \leq 0$，不在合法范围中
WR8	50	50	**200**	$c > 100$，不在合法范围中
WR9	50	50	**50.0**	c 非整数，不在合法范围中

故给出的弱健壮等价类测试用例共13个，如下表所示：

测试用例编号	a	b	c	预期输出
WN1	50	50	50	等边三角形
WN2	40	40	60	等腰不等边三角形
WN3	40	50	60	完全不等边三角形
WN4	10	20	80	不能构成三角形
WR1	**0**	50	50	$a \leq 0$，不在合法范围中
WR2	**200**	50	50	$a > 100$，不在合法范围中
WR3	**aaa**	50	50	a 非整数，不在合法范围中
WR4	50	**0**	50	$b \leq 0$，不在合法范围中
WR5	50	**200**	50	$b > 100$，不在合法范围中
WR6	50	**b5**	50	b 非整数，不在合法范围中
WR7	50	50	**0**	$c \leq 0$，不在合法范围中
WR8	50	50	**200**	$c > 100$，不在合法范围中
WR9	50	50	**50.0**	c 非整数，不在合法范围中

软件测试第八次作业

问题

分析 Chap.5 (Lec.19) 自动售货机软件例子生成的判定表图例的第6列和第23列，分别给出：

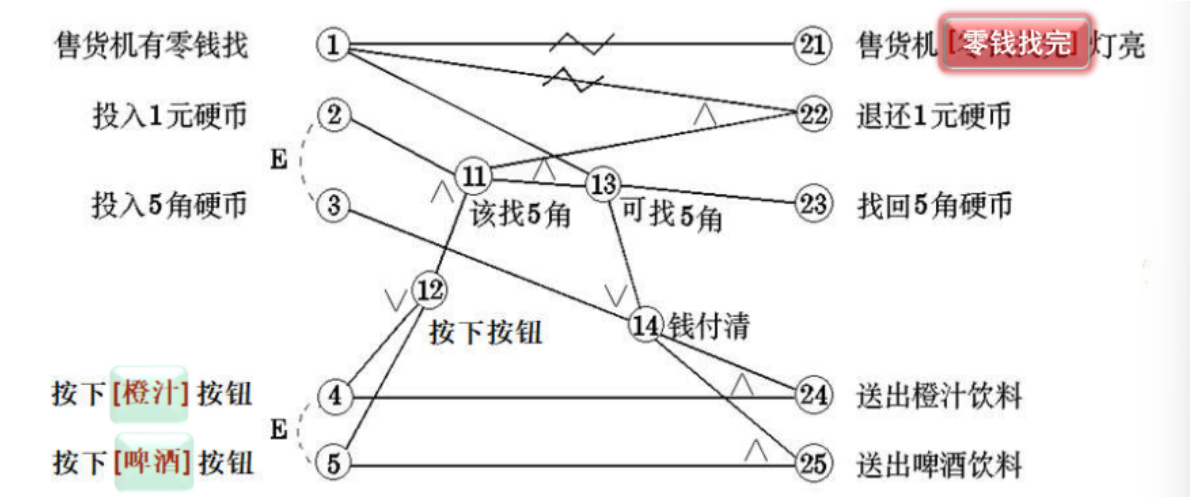
- **输入条件的自然语义陈述；**
- **输出结果的自然语义陈述；**
- **用命题逻辑形式描述实现上述输入-输出过程所应用的判定规则，并写出获得输出结果的推理演算过程。**

判定表图例：

序 号		1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
条件	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
	3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
	4	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
	5	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
中间结果	11						1	1	0		0	0	0		0	0	0						1	1	0		0	0	0		0	0	0
	12						1	1	0		1	1	0		1	1	0						1	1	0		1	1	0		1	1	0
	13						1	1	0		0	0	0		0	0	0						0	0	0		0	0	0		0	0	0
	14						1	1	0		1	1	1		0	0	0						0	0	0		1	1	1		0	0	0
结果	21						0	0	0		0	0	0		0	0	0						1	1	1		1	1	1		1	1	1
	22						0	0	0		0	0	0		0	0	0						1	1	0		0	0	0		0	0	0
	23						1	1	0		0	0	0		0	0	0						0	0	0		0	0	0		0	0	0
	24						1	0	0		1	0	0		0	0	0						0	0	0		1	0	0		0	0	0
	25						0	1	0		0	1	0		0	0	0						0	0	0		0	1	0		0	0	0
测试用例							Y	Y	Y		Y	Y	Y		Y	Y							Y	Y	Y		Y	Y	Y		Y	Y	

条件	中间结果	结果
原因清单 (输入条件) C1 售货机可找零 C2 投入1元硬币 C3 投入5角硬币 C4 按下 橙汁 按钮 C5 按下 啤酒 按钮	建立中间结点，表示处理的中间状态 T11 投入1元硬币且按下饮料按钮 T12 按下 橙汁 或 啤酒 按钮 T13 应当找5角零钱并且售货机有零钱找 T14 钱已付清	结果清单 (输出结果) E21 零钱找完 灯亮 E22 退还1元硬币 E23 退还5角硬币 E24 送出橙汁饮料 E25 送出啤酒饮料

对应因果图：



回答

判定表图例的第6列：

- 输入条件的自然语义陈述：在售货机可以找零时，操作者投入1元硬币并按下”橙汁“按钮。
- 输出结果的自然语义陈述：售货机送出橙汁饮料，退还5角硬币。
- 根据因果图，给出判定规则：

$$C4 \vee C5 \Rightarrow T12$$
$$C2 \wedge T12 \Rightarrow T11$$
$$T11 \wedge C1 \Rightarrow T13$$
$$T13 \vee C3 \Rightarrow T14$$
$$T13 \Rightarrow E23$$
$$T14 \wedge C4 \Rightarrow E24$$

- 推理演算过程：证明 $\therefore C1 \wedge C2 \wedge C4 \Rightarrow E23 \wedge E24$

$$\because C4$$
$$C4 \vee C5 \Rightarrow T12$$
$$\therefore T12$$
$$\because C2$$
$$C2 \wedge T12 \Rightarrow T11$$
$$\therefore T11$$
$$\because C1$$
$$T11 \wedge C1 \Rightarrow T13$$
$$\therefore T13$$
$$\because T13 \vee C3 \Rightarrow T14$$
$$\therefore T14$$
$$\because T14 \wedge C4 \Rightarrow E24$$
$$\therefore E24$$

$$\because T13$$
$$T13 \Rightarrow E23$$
$$\therefore E23$$

$$\therefore C1 \wedge C2 \wedge C4 \Rightarrow E23 \wedge E24$$

判定表图例的第23列：

- 输入条件的自然语义陈述：在售货机不可以找零时，操作者投入1元硬币并按下”啤酒“按钮。
- 输出结果的自然语义陈述：售货机的“零钱找完”灯亮起，退还1元硬币。
- 根据因果图，给出判定规则：

$$\neg C1 \Rightarrow E21$$
$$C4 \vee C5 \Rightarrow T12$$
$$C2 \wedge T12 \Rightarrow T11$$
$$\neg C1 \wedge T11 \Rightarrow E22$$

- 推理演算过程：证明 $\neg C1 \wedge C2 \wedge C5 \Rightarrow E21 \wedge E22$

$$\because \neg C1$$
$$\neg C1 \Rightarrow E21$$
$$\therefore E21$$

$$\because C5$$
$$C4 \vee C5 \Rightarrow T12$$
$$\therefore T12$$
$$\because C2$$
$$C2 \wedge T12 \Rightarrow T11$$
$$\therefore T11$$
$$\because T11 \wedge \neg C1 \Rightarrow E22$$
$$\therefore E22$$

$$\therefore \neg C1 \wedge C2 \wedge C5 \Rightarrow E21 \wedge E22$$