练习题

1. 什么是软件测试?

软件测试(英语：Software Testing)，描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。

经典定义：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

2. 软件测试涉及哪几个关键问题?

ppt上的：

- 软件测试是证伪而非证真
- 尽早地和不断地进行软件测试
- 重视无效数据和非预期使用习惯的测试
- 程序员应该避免检查自己的程序
- 充分注意测试中的群集现象
- 用例要定期评审

网上资料（至少五个方面）：

- 谁来执行测试，
- 测试什么
- 软件测试对象
- 什么时候进行测试
- 怎样进行测试
- 测试停止的标准是什么
- 软件测试误区

（最好把软件测试的原则也写上）

3. 简述软件测试的复杂性和经济性。

复杂性：

黑盒测试的复杂性：穷举输入测试：

> 输入量太大，输出太多，实现的途径太多，规格说明没有客观标准。

白盒测试的复杂性：穷举路径测试：

- 不能保证程序实现符合规格说明的要求。
- 不可能查出程序中因遗漏路径而出现的错误。
- 可能发现不了有关数据的故障。

经济性原则：

- 根据程序的重要性和一旦发生故障将造成的损失来确定它的测试等级；
- 认真研究测试策略，以便能开发出尽可能少的测试用例，发现尽可能多的软件故障。

4. 软件测试中检测到的错误都是由编码错误引起的吗？为什么？

不是，还会有

- 问题定义(需求分析)错误：问题定义不满足用户的要求而导致的错误。
- 规格说明错误：规格说明与问题定义不一致。（冗余，不一致，不完整，不可测试，不可行）
- 设计错误：系统的设计与需求规格说明中的功能说明不相符。（不完全，算法错误，接口错误，控制逻辑错误，数据结构错误）

测试过程中计划-需求-设计-编码，前面3个阶段也可能引进错误，不单单是由编码错误引起。

5. 简述软件测试的流程

单元-->集成-->系统-->验收-->文档测试/评审 -->s 缺陷跟踪

（注意和软件测试的**阶段**区分开来）

6. 静态测试和动态测试分别都应用在什么场景下?

静态测试：

不实际运行被测软件，而只是静态地检查程序代码、界面或文档中可能存在的错误过程。

- 代码测试：主要测试代码是否符合相应的**标准和规范。**
- 界面测试：主要测试软件的实际界面与需求中的说明是否相符。
- 文档测试：主要测试用户手册和需求说明是否真正符合用户的实际需求。

动态测试：

指实际运行被测程序，输入相应的测试数据，检查实际输出结果和预期结果是否相符的过程。

动态测试方法为**结构**和**正确性**测试。

7. 黑盒测试和白盒测试的区别是什么？可以同时使用这两种测试方法吗?

- 从定义上：
- 从测试目的上：

  黑盒测试的目的是检测是否有不正确或遗漏的功能；数据或者参数上，输入能否正确接收；是否有数据结构错误或外部信息访问错误；性能上是否能够满足要求；是否有初始化或终止性错误。

  白盒测试的目的是通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致，而不顾它的功能。
- 检测方式上

不可以同时使用，一般是先黑盒测试后白盒测试。

8. 黑盒测试中，测试人员和程序员应该相互独立。解释其合理性。

程序员应该避免检查自己的程序。程序员与软件产品有着直接的利益关系，有很多理由支持这个原则。测试工 作需要严格的作风，客观的态度和冷静的情绪。但是心理学告诉我们，人们具 有一种不愿意否定自己的自然性心理，这是做好软件测试的一大心理障碍。

9. 若测试机器学习程序，请设计出一些蜕变关系。

10. 程序变异的基本思想是什么?

变异测试的定义:

- 变异测试也称为"变异分析"，是一种对测试数据集的有效性、充分性进行评估的技术，能为研发人员开展需求设计、单元测试、集成测试提供有效的帮助。
- 变异测试通过对比源程序与变异程序在执行同一测试用例时差异来评价测试用例集的错误检测能力。
- 在变异测试过程中，一般利用与源程序差异极小的简单变异体来模拟程序中可能存在的各种缺陷。

给定一个程序p和一个测试数据集t，通过变异算子为p产生一组变异体Mi,对p和m都使用t进行测试运行，如果某mi在某个测试输入t上与p产生不同的结果，则该mi被杀死，若某mi在所有的测试数据集上都与p产生不同的结果，则该mi被杀死；若某mi在所有的测试数据集上都与p产生相同的结果，则称其为活的变异体。接下来对活的变异体进行分析，检查其是否等价于p；对不等价于p的变异体m进行进一步的测试，直到充分性度量达到满意的程度。

11. 什么是变异算子?

定义了从原有程序生成差别极小程序（即变体）的转换规则。

比如：运算符变异，数值变异，方法返回值变异，继承变异，多态变异，重载变异 。

12. 一阶变异和高阶变异有什么区别?

一阶变体:

在对程序进行测试的时候，仅经过一次变更而得到的变体称为一阶变体。

高阶变体:

同样地，二阶变体就是经过了两次简单变更而得到的变体，三阶变体就是经过了三次简单变更而得到的变体，依此类推。对一个一阶变体再进行一次简单变更就可以得到二阶变体，也就是说，一个n阶变体可以由一个(n-1)阶变体进行一次简单变更而得到。高于一阶的变体叫作高阶变体。

在实际中，使用最多的还是一阶变体，主要原因有两个:一个是在数量上，高阶变体的数量远多于一阶变体的数量，大量的变体会影响到充分性评价的可量测问题;另一个是涉及耦合效应。

13. Web 系统具有什么特征?

图形化，与平台无关，分布式的，动态的，交互的。

14. 简述性能测试的基本步骤和流程。

- **性能需求分析**:
    - 系统信息调研
    - 业务信息调研
    - 性能需求评估：业务角度，系统角度（架构，数据库，实时性，大数据）

    - 确定性能测试点：关键业务，日请求量，逻辑复杂度，运营推广活动。
    - 确定性能指标
- **性能测试准备：**

  测试环境准备，测试场景设计，性能工具准备（负载工具，监控工具），测试脚本准备，测试数据准备（负载测试数据，DB数据量大小）

- **性能测试执行：**

  人工边执行边分析，无人值守执行性能测试。

- **结果分析与调优**
- **测试报告与总结**

15. 构造下述三角形问题的弱健壮的等价类测试用例。

1、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。

答：

划分等价类：

假设三条边的长度分别为a，b，c，那么可以划分为4个等价类：

- R1 = {<a, b, c>：三条边分别为a，b，c的等边三角形}
- R2 = {<a, b, c>：三条边分别为a，b，c的等腰不等边三角形}
- R3 = {<a, b, c>：三条边分别为a，b，c的完全不等边三角形}
- R4 = {<a, b, c>：三条边分别为a，b，c，不能构成三角形}

弱健壮有效等价类测试用例覆盖（弱一般等价类测试用例）：

测试用例	a	b	c	预期输出
WN1	7	7	7	等边三角形
WN2	3	3	4	等腰不等边三角形
WN3	4	5	6	完全不等边三角形
WN4	7	1	4	不能构成三角形

弱健壮无效等价类测试用例覆盖：

测试用例	a	b	c	预期输出
WR1	0	7	7	a的值不在取值范围内
WR2	7	0	7	b的值不在取值范围内
WR3	7	7	0	c的值不在取值范围内
WR4	101	7	7	a的值不在取值范围内
WR5	7	101	7	b的值不在取值范围内
WR6	7	7	101	c的值不在取值范围内
WR7	1.1	7	7	a的值不在取值范围内
WR8	7	1.1	7	b的值不在取值范围内
WR9	7	7	1.1	c的值不在取值范围内

16. 对 NextDate 示例，运用等价类划分法给出测试用例

2、NextDate函数问题

NextDate函数包含3个变量，即月份（month）、日期（day）和年份（year），函数的输出为输入日期的后一天。例如，输入为2017年4月25日，则函数的输出为2017年4月26日。函数要求输入变量均为整数值，并且满足下列条件：

（1）1≤month≤12；
（2）1≤day≤31；
（3）1912≤year≤2050。

使用简单等价类划分方法，可以划分为以下有效等价类：

M1 = {month:1≤month≤12}
D1 = {day:1≤day≤31}
Y1 = {year:1912≤year≤2050}

若M1、D1、Y1这3个条件中的任意一个无效，那么NextDate函数都会产生一个输出，其无效等价类：

M2 = {month:month<1}　　D3 = {day:day>31}
M3 = {month:month>12}　　Y2 = {year:year<1912}
D2 = {day:day<1}　　Y3 = {year:year>2050}

NextDate函数的一般等价类测试用例

测试用例	输入			期望输出
	day	*month*	*year*	
TestCase1	25	4	2017	2017年4月26日

NextDate函数的弱健壮等价类测试用例

测试用例	输入			期望输出
	day	*month*	*year*	
TestCase1	25	4	2017	2017年4月26日
TestCase2	25	0	2017	*month*不在1～12中
TestCase3	25	13	2017	*month*不在1～12中
TestCase4	0	4	2017	*day*不在1～31中
TestCase5	32	4	2017	*day*不在1～31中
TestCase6	25	4	1911	*year*不在1912～2050中
TestCase7	25	4	2051	*year*不在1912～2050中

弱健壮等价类测试中的无效测试用例则只包含一个无效值，其他都是有效值，即含有单软件缺陷假设。

NextDate函数的强健壮等价类测试用例

测试用例	输入			期望输出
	day	*month*	*year*	
TestCase1	25	−1	2017	*month*不在1～12中
TestCase2	−25	4	2017	*day*不在1～31中
TestCase3	25	4	1900	*year*不在1912～2050中
TestCase4	−1	−4	2017	变量*day*、*month*无效，变量*year*有效
TestCase5	−1	4	1900	变量*day*、*year*无效，变量*month*有效
TestCase6	25	−4	1911	变量*month*、*year*无效，变量*day*有效
TestCase7	−25	−4	2051	变量*day*、*month*、*year*无效

强健壮等价类测试考虑了更多的无效值情况。强健壮等价类测试中的无效测试用例可以包含多个无效值，即包含多个软件缺陷假设。因为NextDate函数有3个变量，所以对应的强健壮等价类测试用例可以包含一个无效值，2个无效值或3个无效值。

若测试机器学习程序，请设计出一些蜕变关系。

蜕变测试是一种特殊的黑盒测试方法，蜕变测试依据被测软件的领域知识和软件的实现方法建立蜕变关系(Metamorphic Relation, MR)，利用蜕变关系来生成新的测试用例，通过验证蜕变关系是否被保持来决定测试是否通过。

蜕变关系(Metamorphic Relation, MR) 是指多次执行目标程序时，输入与输出之间期望遵循的关系。蜕变测试依据蜕变关系生成更多的后续测试用例，测试时就会多次执行目标程序时，使得程序可以进一步被验证。

简述数据库应用软件与数据库管理系统软件的区别与联系。

区别：

- 数据库应用软件就是实现把用户意义下抽象的逻辑数据处理，转换成为计算机中具体的物理数据处理的软件。
- 数据库应用系统软件是在数据库管理系统（DBMS）支持下建立的一种计算机应用系统软件。

联系:

- 功能都是对数据库进行管理。
- 数据库应用系统软件的组成成分中都包含有数据库应用软件，这两者都是通过数据库管理系统来实现对数据库的管理和操控。

如何对数据库应用软件中设计的数据库模式的好坏进行验证?

模式（schema） 是称逻辑模式，是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。又称概念模式或逻辑模式。

是对所有用户数据逻辑结构和特征的所有描述。主要由数据库设计者进行DDL语言进行描述和定义。体现了数据库的整体观。

编号	数据库设计步骤和输出成果	数据库设计验证观点
1	需求分析 输出：数据字典和数据流图	- 是否反映所有的用户需求 - 是否充分考虑系统的扩充和改变
2	概念结构设计 输出：ER 图 (Entity-Relation 图， 关系实体图)	- 是否涵盖了系统涉及所有的实体和属性 - 实体与属性的划分是否正确 - 实体之间的联系与约束刻画是否准确全面 - 不同子 ER 图中是否存在命名、结构等冲突 - 不同子 ER 图中是否存在冗余
3	逻辑结构设计 输出：库表结构等	- 是否符合关系数据库设计的范式理论，通常应达到 3NF 或者 BCNF - 是否考虑该系统的性能需求等进行去范式化 - 是否考虑该系统的性能要求进行分库分表
4	物理结构设计 输出：存储、索引等	- 索引设计(聚簇索引、唯一索引等)是否合理 - 存储结构(关系、索引、日志，备份等)是否合理

简述 MySQL 的 SQL 功能回归测试过程。

功能测试的内容：

- 功能测试，在每一个开发阶段，去验证在每个业务功能操作上都和设计文件（内部和外部）中规定的一样。
- 开始点：功能测试开始于成功完成单元测试后，和功能测试计划已经被有关各方已批准，并且在基线控制之下。
- 结束点：功能测试结束于执行完所有的计划的测试用例，结果中没严重性为1或者2的缺陷。如果未被测试的用例应该被记录下来，并标明原因；所有测试应该被记录下来；有相应的测试报告和总结。

回归测试的内容有：

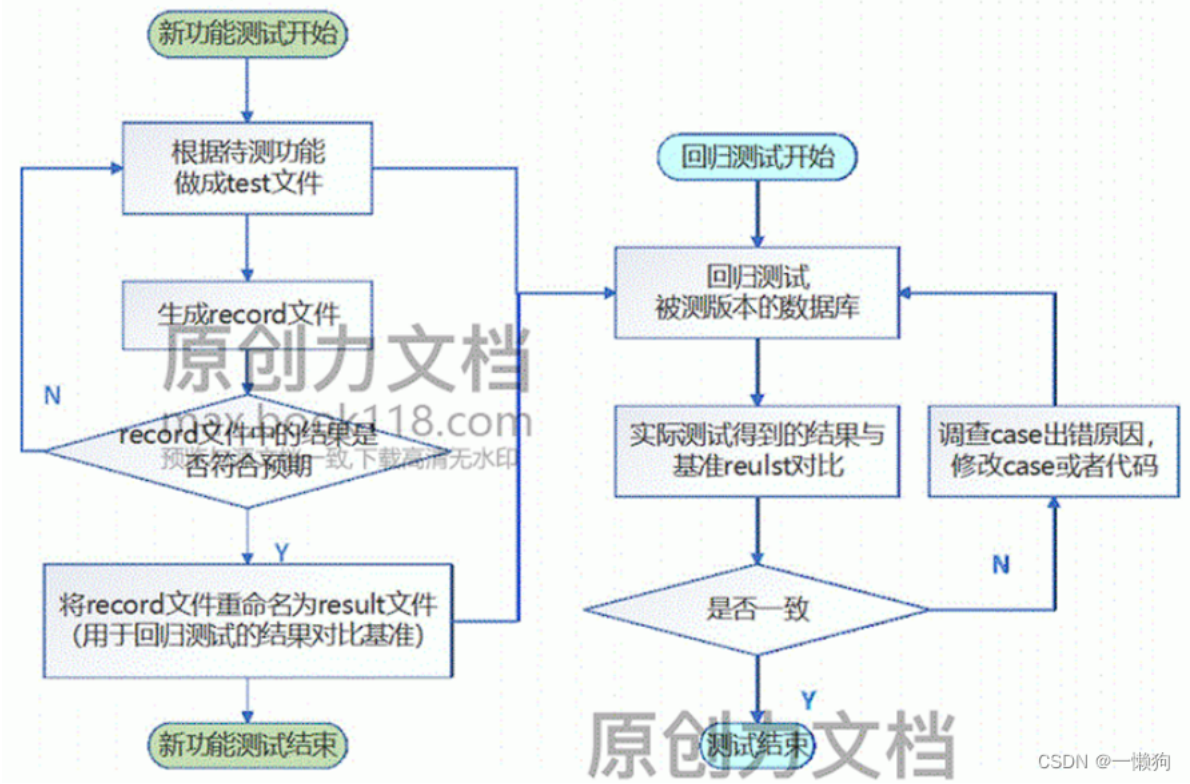
回归测试验证，当系统某部分修改（增加新的功能或者修改bug）后，去验证这部分是否被成功修改和其他部分是否会被这部分的修改所影响。要执行回归测试，应用程序必须运行相同的测试用例通过至少两次。第一次测试是修改前应用程序的特定部分是否正确响应。第一次测试获得的应用程序的正确反应做为第二次运行后判定程序是否正确运行的判定标准。

- 开始点：因为增加新功能或者修改缺陷而对代码进行的修改后开始回归测试。
- 结束点：回归测试结束于成功的执行相关的回归测试用例，并且修改后的程序相关部分没还未解决的缺陷。

手动测试：根据SQL手册，设计测试用例并手动执行SQL。

自动测试：MySQLTest自动化测试框架。

MySQL的SQL功能回归测试可以使用MySQLTest这个测试框架进行，具体回归测试过程如下，首先需要按照新功能测试步骤生成基准文件，其次按照回归测试流程测试，将测试结果与基准文件进行自动对比。



请举例说明如何测试 DBMS 的事务特性。

自动测试：

MySQLTest自动化测试框架事务（数据操作的最小逻辑单元）特性：

- ACID 原子性(Atomic)：该事物中的SQL语句或者全部执行，或者全部不执行。
- 一致性(Consistent)：一个事务在执行之前和执行之后，数据库的数据都必须处于一致的状态
- 隔离性(Isolation)：在并发环境中，并发的事务时相互隔离的，一个事务的执行不能被其他事务干扰.
- 持久性(Duration)：一旦事务提交，事物对数据库中的对应数据的状态变更就会永久保存到数据库中。

测试方法：同时开启多个客户端连接，设置不同隔离级别，测试不同的并发操作。

如何对 DBMS 进行性能测试？性能测试中重点需要考虑哪些内容？

DBMS的基准性能测试：

- OLTP（Online Transaction Processing）联机业务处理测试标准：TPC-C、TPC-E（最新）
- OLAP（Online Analysis Processing）决策支持/大数据测试标准：TPC-H、TPC-DS（最新）

常用工具：Sysbench、BechmarkSQL(TPC-C)等

性能测试中重点需要考虑的因素：

1）性能测试工具

2）性能度量指标

3）性能测试场景（基准或自定义场景）

4）性能测试数据量与数据分布

5）被测DBMS的架构特征

数据库性能测试需要如下几个**步骤：**

- 明确测试目的
- 设计测试模型 （即压力模型）
- 准备测试集群环境
- 准备压力测试工具或者编写压力测试脚本
- 明确性能指标并加监控
- 根据2设计的测试模型准备测试数据
- 测试执行
- 测试分析

DBMS 的高可用性测试的主要挑战有哪些?

高可用是分布式系统架构设计中必须考虑的因素之一。它通常是指，通过设计减少系统不能提供服务的时间。如果系统能一直提供服务，我们就说该系统的可用性是100%。

挑战举例：搭建各种高可用架构，模拟高可用场景的复杂故障等

DBMS的高可用性测试：

● 高可用：指在DBMS发生故障时恢复业务运行的能力。

● 通常采用高可用架构：例如一主N备；三地两中心等

● 高可用测试：通过故障模拟（混沌测试），例如利用工具Jepsen、ChaosBlade等模拟网络故障，CPU超载，服务器宕机，内存不足等，测试在这些极端条件下系统的处理。

简述第十一章所讲的几种智能搜索算法的优劣性。

目前软件工程领域中用到的搜索方法分为3大类：

- 第1类是基于微积分的搜索方法
- 第2类是带有向导的随机搜索方法
- 第3类是枚举方法

1.遗传算法

优点：搜索从群体出发，存在潜在的并行性，可以和多个个体同时比较；搜索使用评价函数启发， 过程简单；具有可拓展性，容易和其他算法结合

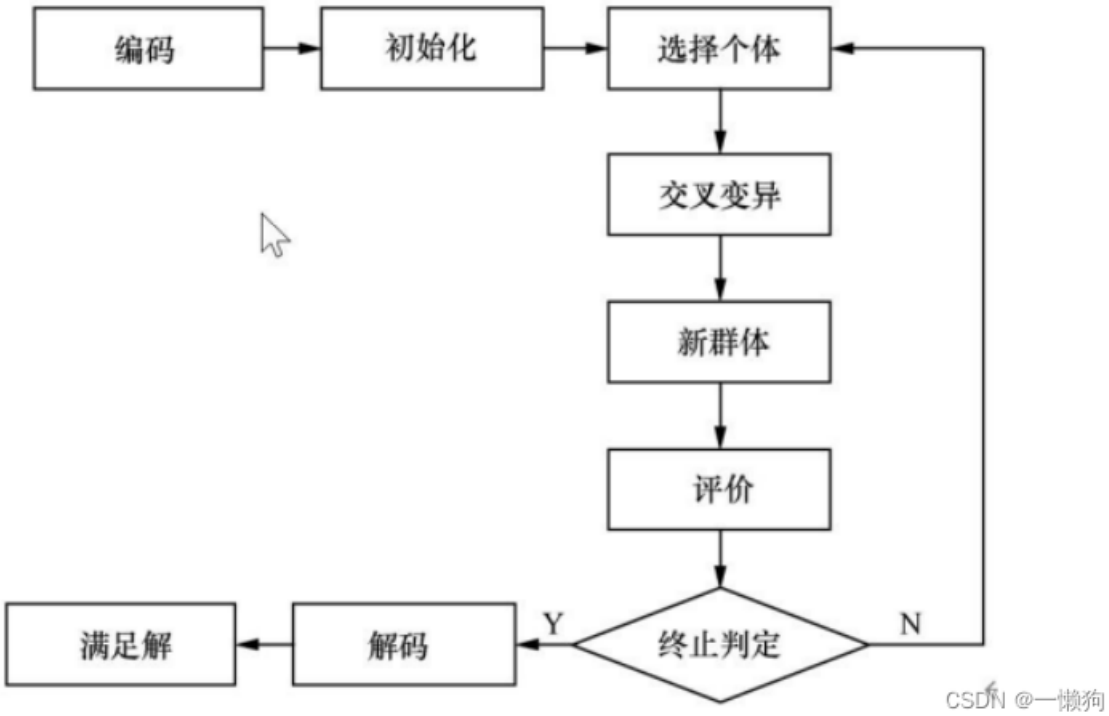
缺点：遗传算法的编程实现复杂，找到最优解以后还需要对问题进行编码，且因为不能及时利用网络的反馈信息，导致算法的搜索速度较慢。

2.蚁群算法

优点：采用正反馈机制，使得搜索过程不断收敛，最终逼近最优结果；搜索过程采用分布式计算方式，多个个体同时进行并行计算，大大提高了算法的计算能力和运行效率。

缺点：如果多样性过剩，系统过于活跃，会导致过多的随机运动，陷入混沌状态。如果多样性不够，正反馈过强，会导致僵化，当环境变化时蚁群不能相应调整。

简述遗传算法的过程。



云测试有哪两层含义?

第一种是有效利用云计算环境资源对其他软件进行测试，即基于云计算的测试。

第二种是针对部署在“云”中的软件进行测试，即面向云计算的测试。

针对云软件质量属性定量化描述的问题，最直接的思路有哪两种?

（1）通过有效的定义软件失效和选取恰当的工作负载度量方式，定义软件可靠性。

（2）通过诊断软件故障信息，结合软件内部度量，提供其他产品质量度量。

请列举几种云安全技术。

云计算工作负载保护平台、云访问安全代理（CASB）、托管检测和响应（MDR）、微分区等。

云安全主要考虑的**关键技术**有：数据安全、应用安全、虚拟化安全。

云计算安全与传统软件安全有哪些区别?

1、安全边界模糊化

2、数据安全要求更高

3、责任共担模型

4、新环境中的错误配置

5、合规性要求升级

安全测试包括哪些主要的挑战?

1.测试理论很难适用于安全领域；

2.安全测试基础理论薄弱,当前测试方法缺少理论指导，也缺乏技术产品工具 。

简述 SOFIA 检测 SQL 注入的过程。

1.对sql语句进行解析。

2.裁剪，替换所有常量数值和字符串。

3.计算树编辑距离，利用approxlib 工具，该工具实现了计算树编辑距离的算法，利用树编辑距离 进行分类，往往是不够的。

4.聚类，利用K-中心算法进行分类。

企业的测试策略体现在几个方面?

1、测试的对象和范围是什么（测试什么东西，哪些不需要测试）。

2、测试目标是什么（为了让产品完全符合商业化的标准，还是小范围适用等）。

3、测试的重点和难点有哪些（测试难点在哪里，需要什么样的支持）

4、如何安排各类测试活动（先测试什么再测试什么，什么时候集成测试等）

5、资源投入情况（测试时长、人员配置、环境等）

为什么要制订测试计划?

- 制定测试计划能够把自身知识和经验直接转化为执行任务的具体方法，在方法中更能体现出制定者的自身素质以及能力。
- 制定测试计划可以促进团队间关于测试任务和过程的交流，增强团队之间的默契，可以高效率、高配合、高质量的完成测试任务。
- 制定测试计划可以为组织、安排和管理测试项目提供一个整体框架，对项目执行过程中的风险进行分析，并制定相关的应对策略。
- 制定测试计划中对人员的合理安排化，通过对每个员工的专长来对应分配难易任务，整个项目的进行就会显得合理化、层次化、条理化。同时将职责清晰地具体划分到个人身上，也有利于日后的纠错，即使发现哪个环节出现问题，及时弥补。

简述基于 CMMI 的测试流程和传统测试流程的区别。

> CMMI（Capability Maturity Model Integration）即能力成熟度模型集成，其前身为CMM，原本是当年美国军方为了评估自己的软件产品供应商的过程质量水平，而委托美国卡内基梅隆大学软件工程学院（SEI）开发的一套过程评估体系，后来又被推广到全世界，成为全球软件企业重要的过程改进方法之一，后因其衍生品的派系林立（如：SW-CMM、SE-CMM、IPT-CMM等等），SEI决定在SW-CMM、SE-CMM的基础上，加入委外采购、IPPD等相关内容，并融合ISO9000部分理念，形成了今天我们看到的CMMI，这也就是CMMI中的“I”的由来。

基于CMMI 的测试流程建立了一个自动的、可扩展的框架。因而能够从总体上改进组织的质量和效率。CMMI主要关注点就是**成本效益**、**明确重点**、**过程集中**和**灵活性**4个方面。

了解当前互联网公司是如何将 DevOps 部署到企业的软件质量保障流程中的

通过CI/CD持续部署，让开发运维成为一个统一的生态系统，并进行各个环境部署