目录

一. 测试基础

1. 如何制定测试计划

参考答案：

测试计划包括测试目标、测试范围、测试环境的说明、测试类型的说明（功能，安全，性能，稳定性）、测试工具、模块的划分、测试负责人、测试执行轮次的时间安排、相关文档在文档管理库中的位置、测试的风险 。其中模块划分需要根据测试人员对于业务的熟悉程度及个人能力进行分配，工作量的估算需要根据以往测试时的经验，结合本次需求的修改，可以大致估算出测试量

2. 在项目中如何保证软件质量

参考答案：

项目质量不仅仅是某个人或某个团队来保障的，而是整个团队一起努力的结果，在公 司级别需要有一个规范的项目流程

1. 产品，保证迭代过程中的产品逻辑，对于可能的兼容，升级做出预判，并给出方案
2. 设计，满足产品表达的同时，保证设计的延续性
3. 开发，产品细节的保证，技术方案选择要严谨，考虑兼容，性能，开发完成后要充分自测，严格遵循开发规范操作
4. 测试，验证产品逻辑，站在用户角度对交互设计进行系统验证，尽可能多的使用技术手段保证测试质量

3. 功能测试用例一般包含哪些内容？

参考答案：

要素一般包括：用例编号、用例优先级、测试目的、所属模块、前提条件、测试环境、

输入数据、测试步骤、预期结果、测试脚本等

核心要素：用例优先级、测试目的、预期结果

4. 黑盒（或功能）测试用例设计方法有哪些？

参考答案：

等价类划分方法：等价类划分法将程序所有可能的输入数据（有效的和无效的）划分成若干个等价类。

边界值方法：边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试方法。

错误推测方法：在测试程序时，人们可以根据经验或直觉推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的测试用例的方法。

因果图方法：因果图法是一种适合于描述对于多种输入条件组合的测试方法，根据输入条件的组合、约束关系和输出条件的因果关系，分析输入条件的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件涉及的各种组合情况。

判定表驱动分析方法：判定表是黑盒测试的方法之一，判定表是把作为条件的所有输入的各种组合值以及对应输出值都罗列出来而形成的表格。

正交分解法：是研究多因素多水平的一种设计方法，它是根据正交性，由试验因素的全部水平组合中挑选出部分有代表性的点进行试验，通过对这部分试验结果的分析了解全面试验的情况，找出最优的水平组合。

场景分析法：分析软件应用的场景，从用户的角度出发，从场景的角度来设计测试用例，是一种面向用户的测试用例设计方法。关心用户做什么，而不是关心产品做什么。

全局探索式测试方法：测试人员根据应用程序所提供的信息自由发挥，不受限制，不受任何约束的探索程序的各种功能。

5. APP 测试和 web 测试有什么区别

Web 端测试和移动端测试类型基本相似，都需要进行功能测试、性能测试、安全性测试，他们主要区分 web 端一般都是 b/s 架构，基于浏览器的，app 是 c/s 架构，是有客户端的。

(1) 从系统架构来看的话：web 测试只要更新了服务器端，客户端就会同步更新；而如果是 app 端下修改了服务端，意味着客户端用户所有使用的核心版本都需要进行回归测试一遍。

(2) 客户端性能方面：Web 端可能只会关注响应时间；App 则还要关心流量、电量、cpu、等；

(3) 兼容方面：Web 是基于浏览器的，所以更倾向于浏览器（IE、Chrome、firefox）和电脑硬件，电脑系统方向的兼容；App 测试则必须依赖于手机或者 pad，不仅要看分辨率、频目尺寸、重要看设备系统。

6. 发现一个 bug，怎么定位是 APP 端还是服务端的问题

1. 抓包分析 通过对客户端进行抓包，分析服务端返回的数据是否符合预期，如果服务端数据是正确的，那就是客户端的问题

2. 日志分析 可以通过查看客户端/服务端的日志，分析有没有异常的日志信息，

从而确定具体原因

7. 针对 App 的安装功能，写出测试点？

1. 正常安装测试，检查是否安装成功。
2. APP 版本覆盖测试。例如：先安装一个 1.0 版本的 APP,再安装一个高版本(1.1 版本)的 APP，检查是否被覆盖。
3. 回退版本测试。例如：先装一个 2.0 版本的 APP,再安装一个 1.0 版本的 APP,正常情况下版本是可以回退的。
4. 安装时内存不足，弹出提示。
5. 根据安装手册操作，是否正确安装。
6. 安装过程中的意外情况（强行断电、断网、来电话了、查看信息）等等，检查会发生的情况。
7. 通过'同步软件'，检查安装时是否同步安装了一些文件。
8. 在不同型号、系统、屏幕大小、分辨率上的手机进行安装。
9. 安装时是否识别有 SD 卡，并默认安装到 sd 卡中。
10. 安装完成后，能否正常启动应用程序。
11. 安装完成后，重启手机能否正常启动应用程序。
12. 安装完成后，是否对其他应用程序造成影响。
13. 安装完成后，能否添加快捷方式。
14. 安装完成后，杀毒软件是否会对其当做病毒处理。
15. 多进程进行安装，是否安装成功。
16. 在安装过程中，所有的提示信息必须是英文或者中文，提示信息中不能出现代码、符号、乱码等。
17. 安装之后，是否自动启动程序。
18. 是否支持第三方安装。
19. 在安装中点击取消。

8. 持续集成的目的是什么

持续集成指的是，频繁地（一天多次）将代码集成到主干。
它的好处主要有两个：
（1）快速发现错误。每完成一点更新，就集成到主干，可以快速发现错误，定位错误也比较容易。
（2）防止分支大幅偏离主干。如果不是经常集成，主干又在不断更新，会导致以后集成的难度变大，甚至难以集成。
持续集成的目的，就是让产品可以快速迭代，同时还能保持高质量。它的核心措施是，代码集成到主干之前，必须通过自动化测试，只要有一个测试用例失败，就不能集成。

9. 如何测试纸杯

功能度：纸杯容量（空杯、满杯升数、半杯升数）；水能不能被喝到；纸杯形状（正圆柱、

上宽下窄圆柱、上窄下宽圆柱、其他形状）、纸杯材质（全纸质、全塑料、半纸半塑料）、纸杯耐温程度（冷水、热水、冷水、冰）、支持盛放液体名称（水、咖啡、牛奶、可乐）

安全性：杯子有没有毒或细菌、装液体多久有化学反应（例如：异味）

可靠性：杯子从不同高度落下的损坏程度

可移植性：杯子在不同的地方、温度等环境下是否都可以正常使用

兼容性：杯子是否能够容纳果汁、白水、酒精、汽油等

易用性：杯子是否烫手、是否有防滑措施、是否方便饮用、装液体多久漏水、装热水多久变形、装多少度热水变形

用户文档：使用手册是否对杯子的用法、限制、使用条件等有详细描述

疲劳测试：将杯子盛上水放 24 小时检查泄漏时间和情况；盛上汽油放 24 小时检查泄漏时间和情况等

压力测试：用根针并在针上面不断加重量，看压强多大时会穿透；手挤压多久变形（单手、双手）

10. 当开发人员说不是 BUG 时，你如何应付？

开发人员说不是 bug，有 2 种情况，一是需求没有确定，所以这个时候可以找来产品经理进行确认，需不需要改动，商量确定好后再看要不要改。二是这种情况不可能发生，所以不需要修改，这个时候可以先尽可能的说出是 BUG 的依据是什么？如果被用户发现或出了问题，会有什么不良结果？程序员可能会给你很多理由，你可以对他的解释进行反驳。如果还是不行，那可以给这个问题提出来,跟开发经理和测试经理进行确认,如果要修改就改,如果不要修改就不改。如果最终 bug 被确定不改，那么就要在测试报告里面记录一下，以便以后查阅。

11. 遇到概率性 bug 怎么办？

概率性 bug，又叫幽灵 bug，首先需要明确的是，该类 bug 也是需要提单的，描述清楚当时操作环境、操作步骤、数据、并提供必要日志，可备注上可能产生原因。然后耐心一点，运用排除法、错误推测找规律，必要时找开发人员、项目经理一起定位分析讨论，如果最终仍未解决，那么需要在测试报告中体现，并分析可能造成的影响，大家一起权衡该 bug 是否可遗留。

12. 一个身份证号码输入框，怎么设计用例

校验身份证号规则的有效性（包括地址码、生日期码、顺序码和校验码

校验 15 位身份证号和 18 位身份正好都是可用的

校验末位是 X 的情况

校验不足 15 位、16-17 位和大于 18 位的情况

如果是必输项，校验不输入的时候会不会有正确的提示

如果不是必输项，则要校验不输入的时候流程能否正常进行

校验输入非数字的情况，是否会有正确提示信息（包括大小写字母、汉字、特殊字符和标点符号）

校验输入全角的数字的时候，系统是否会识别（这个得根据需求确定是否可以使用全角

的数字）

13. 什么是回归测试？如何做回归测试？

回归测试，即就是在软件生命周期中，只要软件发生了改变，就可能给该软件产产生问题;所以，每当软件发生变化时
我们就必须重新测试现有的功能，以便确定修改是否达到了预期的目的，检查修改是否破坏原有的正常功能。
回归测试可以发生在任何一个阶段，包括单元测试、集成测试和系统测试
那我们改如何做回归测试呢？　总结为以下几点

1、在测试策略制定阶段，制定回归测试策略
2、确定需要回归测试的版本
3、回归测试版本发布,按照回归测试策略执行回归测试
4、回归测试通过，关闭缺陷跟踪单（问题单）
5、回归测试不通过，缺陷跟踪单返回开发人员，开发人员重新修改问题，再次提交测试人员回归测试

14. 如何提交一份高质量的缺陷跟踪单

首先要明确，缺陷跟踪单不仅仅是给自己看的，所以高质量的缺陷单，最主要的一条判断标准是，别人一看就懂，标题简洁明了，步骤条理清晰。还需考虑缺陷的完备性，比如缺陷等级、所属功能模块、版本、复现步骤、预期结果、实际结果、产生原因、日志截图等

15. 项目测试流程有哪些？

需求评审、测试计划、技术评审、用例编写、用例评审、测试执行、测试报告、线上验证、项目总结

16. Bug 优先级和严重程度如何划分

Priority(优先级)和 Severity(严重程度)是提交 bug 的两个重要属性。
通常，测试 人员在提交 Bug 时，只定义 Bug 的 Severity, 即该 Bug 的严重程度，而将 Priority 交给 Project Leader 或 Team Leader 来定义，由他们来决定该 Bug 被修复的优先等级。某种意义上来说，Priority 的定义要依赖于 Severity，在大多数情况下，Severity 越严重，那这个 Bug 的 Priority 就越高。
Severity（严重程度）如下：

Blocker（有妨碍的）: 即系统无法执行、崩溃或严重资源不足、应用模块无法启动或异常退出、无法测试、造成系统不稳定

Critical（紧要的）：即影响系统功能或操作，主要功能存在严重缺陷，但不会影响到系统稳定性

Major（严重的）：即界面、性能缺陷、兼容性。

Minor/Trivial（次要的/不严重的）:即易用性及建议性问题。

Priority（优先级）：Immediate（立刻）、Urgent（紧要、优先）、High（高度重视）、Normal（正常）、Low（稍缓）

17. 您认为做好测试用例设计工作的关键是什么

关键点就是熟悉需求，但是需求可以分为以下几个方面

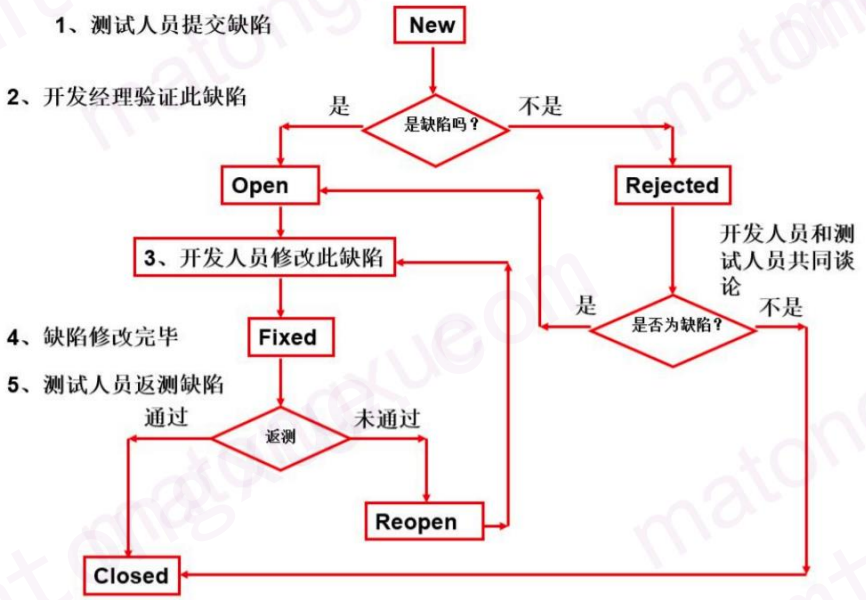
1. 熟悉本次业务需求
2. 熟悉其他系统和本次需求的关联
3. 熟悉开发设计文档，了解开发实现逻辑
4. 熟悉数据库设计文档，了解数据存储
5. 熟悉项目架构，发现隐藏需求

18. 给你一个网站，如何开展测试

1.查找需求说明、网站设计等相关文档，分析测试需求。
2.制定测试计划，确定测试范围和测试策略。
3.设计测试用例，包括功能、兼容、性能、安全等方面
4.开展测试执行
5.回归测试及测试总结

19. bug 的生命周期

New：发现 bug，未经评审决定是否指派给开发人员进行修改。
Open：确认 bug，并且认为需要进行修改，指派给相应的开发人员。
Fixed：开发人员进行修改后标识成修改状态，有待测试人员的回归测试验证。
Rejected：如果认为不是 Bug，则拒绝修改。
Delay：如果认为暂时不需要修改或暂时不能修改，则延后修改。
Closed：修改状态的 Bug 经测试人员的回归测斌验证通过，则关闭 Bug。
Reopen：如果经验证 Bug 仍然存在，则需要重新打开 Bug，开发人员重新修改。



20. 说说微信朋友圈功能的核心测试点

参考答案：

21. 测试用例里都包含哪些要素？

用例编号、所属模块、用例标题、前置条件、操作步骤、优先级、预期结果等

22. 在项目测试过程中，你都测出过哪些类型的 bug？哪些地方最容易出 bug？

bug 的类型：主要是代码逻辑错误、配置错误
哪些地方容易出 bug：参数校验、边界条件、复杂的逻辑、以及一些异常的场景没考虑周全

23. 如何查看一个 APP 的错误日志？

a. 查看 APP 的包名
b. 通过报名查看 APP 的进程号
c. 通过进程号，过滤 adb logcat 中的错误日志

24. 小明在刷抖音时发了一个评论，但是 APP 界面没显示出来，如何排查这个问题？

1.检查客户端网络是否有问题，可以查看其他 APP 能否正常使用
2.检查是否为版本问题，可以换个操作系统（安卓、ios），或者换个其他软件版本试试
3.检查是否为兼容性问题，可以换个手机试试
4.抓包分析，如果 APP 没有向服务器发送请求，或者请求参数对不对，就是 APP 的问题；如果服务端响应数据不对，就是服务端的问题

25. 没有需求文档，如何开展测试

1、没有需求文档不代表没有需求。
2、可以找相关人员进行沟通，获取需求，比如产品经理、开发人员
3、可以参考同行业竞品，总结梳理需求
4、可以根据用户的使用习惯和一些行业的规范，来总结一些功能需求

26. 如何提高用例的覆盖率，减少漏测

1、要根据需求文档来编写用例，确保每条需求都被对应的用例覆盖
2、要充分理解业务，挖掘隐形需求，并编写对应的用例
3、除了正常的业务场景，多考虑一些异常的场景和数据
4、要从多个维度对软件进行测试，功能、性能、安全等各方面来考虑
5、多站在用户的角度去思考问题，模拟用户的使用场景
6、组织用例评审

27. 如何通过 Fiddler 确定是前端还是后端 BUG

（1）测试遇到 bug，首先 Fiddler 开启抓包后看是否能抓到接口，如果不能抓到接口，说明是前端 BUG

（2）如果抓到接口，但是接口请求参数或者参数值错误，说明是前端 BUG

（3）如果是接口应答信息错误，说明是后端 BUG

28. 弱网下等待页面加载超时时间一般多久？

每个公司可能会有不同，有些可能是 5 秒、7 秒，等待加载过程中出现菊花提示，如果加载到 5-10 秒钟还没有加载好，就自动把它结束掉，并进入下一个页面（提示加载失败请检查网络）。

29. 你们公司是怎么做 app 兼容性测试的？公司的测试机少怎么办？

（1）主要还是通过手工去测试已有的机器。

（2）有时可能会有测试机不够的情况，用自己的新手机或者跟公司的同事借一下，以及交叉测试。

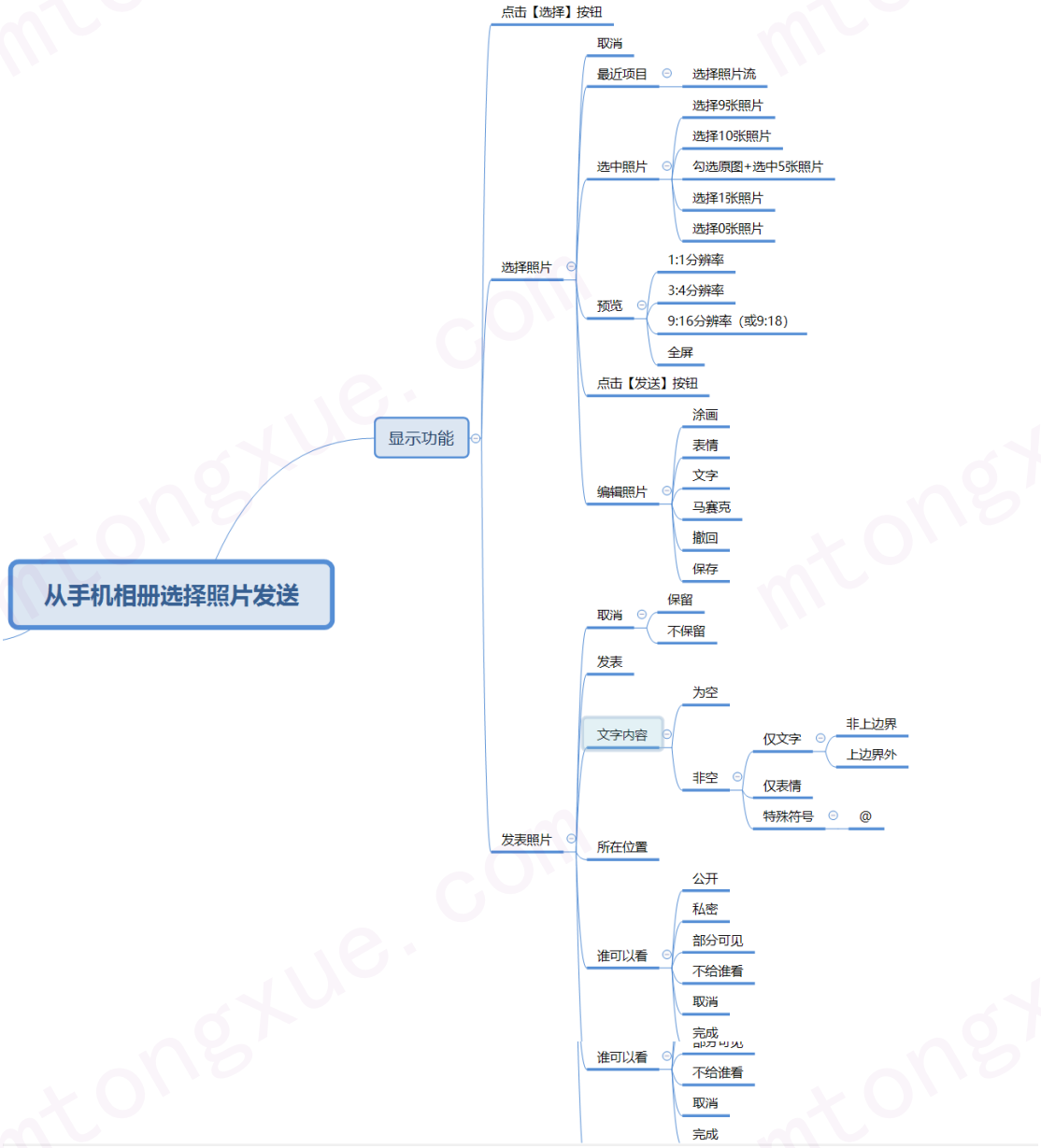
（3）个别用户会反馈在哪些机型上出现一些崩溃，有时哪些机型确实要去测试一下。

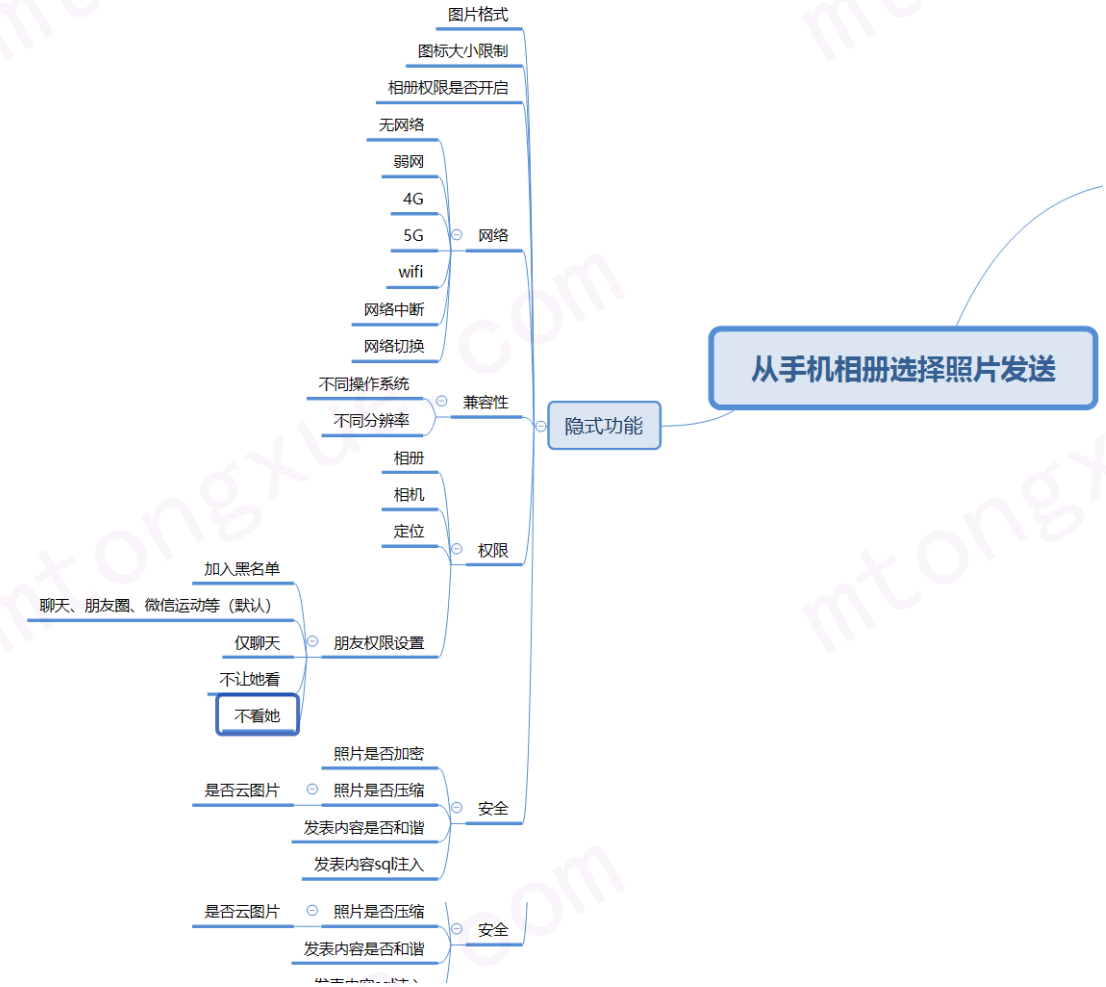
（4）录制自动化脚本，夜间执行自动化测试用例

（5）部分机型通过云测平台（testIn、weTest）的云真机测试，付费居多。

（6）使用模拟器测试可能会出问题，经常出现模拟器没有问题而真机有问题，所以尽可能拿真机覆盖。

30. 请设计某信从手机相册选择照片发朋友圈测试点


从手机相册选择照片发送
显示功能
点击【选择】按钮
选择照片
取消
最近项目
选择照片流
选中照片
选择9张照片
选择10张照片
勾选原图+选中5张照片
选择1张照片
选择0张照片
预览
1:1分辨率
3:4分辨率
9:16分辨率（或9:18）
全屏
点击【发送】按钮
编辑照片
涂画
表情
文字
马赛克
撤回
保存
发表照片
取消
保留
不保留
发表
文字内容
为空
非空
仅文字
非上边界
上边界外
仅表情
特殊符号
@
所在位置
谁可以看
公开
私密
部分可见
不给谁看
取消
完成
谁可以看
不给谁看
取消
完成



31. App 做过哪些专项测试

（1）APP 冷/热启动测试
（2）APP 权限测试（设备权限、app 权限设置）
（3）APP 安装/卸载/（弱、强）升级
（4）APP 消息推送（APP、短信、微信、QQ）
（5）APP 前端性能（耗电量、顺滑度、耗流量）
（6）APP 弱网
（7）APP 稳定性测试

32. 埋点测试如何做？

1.抓取手机日志（不推荐）
（1）比如 iOS 的 XCoder、安卓的 ADB 去进行调试
（2）手机客户端是通过接口向服务器传递数据
2.使用 Fiddler 等抓包工具抓取手机日志
手机客户端是通过接口向服务器传递数据

3.在 Linux 服务器上进行测试（推荐）
数据最后是存储在服务器里面的，
因此可以直接在服务器里面查看数据是否存储成功，
一般在测试环境测试（防止删库）。
4.在数据平台上进行测试
打下埋点后，去数据平台查看数据是否有问题的、App 有多少新的埋点，
很多数据平台有一定延迟性，不会打一个点就立刻在数据平台上显示出来，
一般数据平台会隔一天或者半天拉一次数据。

33. 弱网测试点有哪些

参考答案：

34. 在实际工作中，是不是要 Bug 全部修复完才能达到上线呢？如果上线时间很紧急，还有没修复完的 Bug 怎么办？

参考答案：

1）一般来说，如果有等级是 1 级、2 级的 Bug，是不允许带上线的。

2）如果有 3 级 Bug、4 级 Bug 的话，可以让产品（或项目经理）进行定夺。

3)如果影响范围不大时间又比较急的话,带着不严重的 Bug 上线也是可以接受的，但是一定要在测试报告中注明该遗留 BUG，并说明修复排期。

35. 介绍下你最近做过的一个项目

参考答案：

结合自己测过的业务细节，从以下几个方面来回答：

1） 项目名称

2） 项目平台（终端）：app/web/pad

3） 主要业务：针对****（用户群体，例如青少年群体、上班族）提供**、**、**等功能的软件

4） 主要模块：

5） 我负责**、**、**模块的测试，包括功能、接口、兼容性、界面（app 专项）

6） 项目的主流程：例如冒烟测试用例

7） 系统架构

8） 优势亮点：技术亮点（技术在***项目的应用，例如，使用 Fiddler 断点、mock），管理亮点（例如，协调把控项目进度，分配测试任务，文档质量评审），其他亮点（例如，沟通能力、工作态度、多角度换位思考）

36. 你最近这家公司都用过哪些测试机

参考答案：

1，6、8p、x、xmax，小米 6、华为荣耀、vivo，

2，也可以参考云测平台列举的机型（例如：wetest，testin）

3，大公司测试机多，小公司测试机少

4，根据埋点数据选购热门机型测试。

37. 你上一家公司***APP 项目里面开发和测试人员的比例分配是多少？

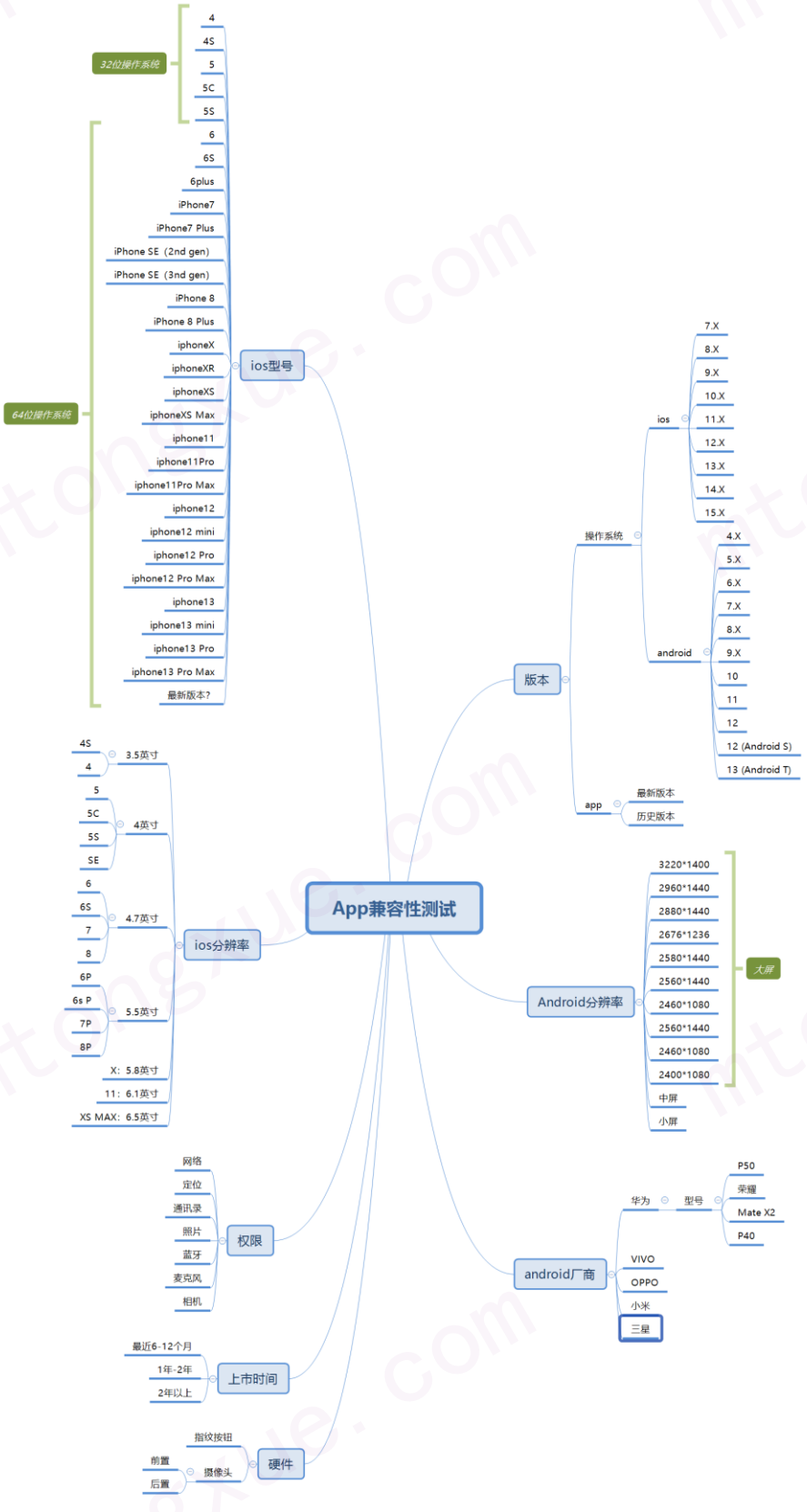
参考答案：

我上一家公司，加上我，一共是四个测试，一个测试的小组长加上三个组员，

前端开发大概是八九个人左右，后端开发大概是四五个人左右，产品是三个人。

38. APP 兼容性测试测试点有哪些？

参考答案：


App兼容性测试
ios型号
32位操作系统
4
4S
5
5C
5S
64位操作系统
6
6S
6plus
iPhone7
iPhone7 Plus
iPhone SE（2nd gen）
iPhone SE（3nd gen）
iPhone 8
iPhone 8 Plus
iphoneX
iphoneXR
iphoneXS
iphoneXS Max
iphone11
iphone11Pro
iphone11Pro Max
iphone12
iphone12 mini
iphone12 Pro
iphone12 Pro Max
iphone13
iphone13 mini
iphone13 Pro
iphone13 Pro Max
最新版本?
版本
操作系统
ios
7.X
8.X
9.X
10.X
11.X
12.X
13.X
14.X
15.X
android
4.X
5.X
6.X
7.X
8.X
9.X
10
11
12
12 (Android S)
13 (Android T)
app
最新版本
历史版本
ios分辨率
3.5英寸
4S
4
4英寸
5
5C
5S
SE
4.7英寸
6
6S
7
8
5.5英寸
6P
6s P
7P
8P
X：5.8英寸
11：6.1英寸
XS MAX：6.5英寸
Android分辨率
大屏
3220*1400
2960*1440
2880*1440
2676*1236
2580*1440
2560*1440
2460*1080
2560*1440
2460*1080
2400*1080
中屏
小屏
权限
网络
定位
通讯录
照片
蓝牙
麦克风
相机
上市时间
最近6-12个月
1年-2年
2年以上
硬件
指纹按钮
摄像头
前置
后置
android厂商
华为
型号
P50
荣耀
Mate X2
P40
VIVO
OPPO
小米
三星

39. 都用过哪些 BUG 缺陷管理工具?

参考答案：

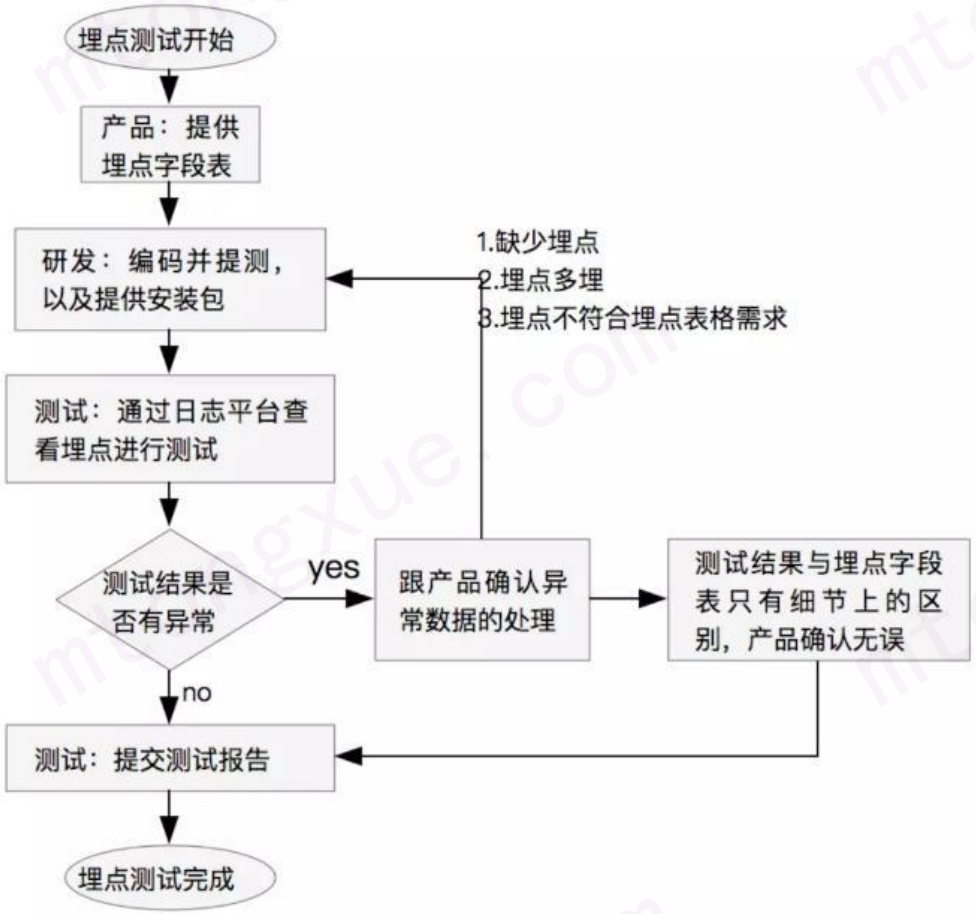
1.市面上常用的是禅道、JIRA

2.有些公司会自己开发一套 Bug 管理工具

3.Bugzilla、Bugfree 用得不多

40. APP 埋点测试流程

参考答案：



41. iOS 系统和 Android 系统的区别

参考答案：

1）iOS 稳定性与安全性高，iOS 可能没有系统性问题，只是可能有屏幕分辨率兼容性问题。

2）Android 因为开源而导致碎片化严重，每个厂商都定制了自己的 ROM。

3）Android 容易信息泄露，各手机厂商都会对 Android 进行修改，有权限等安全性问题。

4）流畅度上，Android 采用虚拟机运行机制，这种机制导致运行一段时间后就容易出现卡顿；ios 几乎不会出现卡顿现象。

5）性价比上，iso 系统拥有自研专利，并且目前为止禁止其他生产厂商使用；

Android 系统是 Google 公司提供的免费、开源系统，Android 比 IOS 开放更多的应用接口 API，Android 的性价比要远高于 IOS。

42. 你怎么保证你的测试用例能百分之百覆盖所有测试点？

参考答案：

如果拿到一个比较大的功能需求，我会按照以下顺序去写测试用例以及执行

1、冒烟测试，正常的流程是否走的通

2、页面元素的检验，即检查页面字段内容、格式、边界值、数据类型、特殊字符、样式、布局等等跟业务没关系的检查，适用所有系统

3、接口测试，通过工具传参看接口能否正常响应，包括输入一些异常的数据，看接口是否有校验

4、业务逻辑检查，这个需要充分解读需求文档上的每一句话，逻辑判断控制，以及有耦合关系的模块，前置、后置等相关联的业务模块是否都正常，而不只是检查当前的功能模块没问题就可以了

5、数据库表检查，即前台提交的表单是否在对应的每一个表字段都正确的写入。

6、异常类测试，例如系统在弱网或者断网情况下页面是都有提示或者相关的判断，或者是一些交易类的功能可能会回调超时，超时代码是否有重发机制等等（具体要看你测的是什么领域的业务，都有一些特殊的场景或者异常操作，往往这些就是测试的盲点）

7、兼容性测试，即你的系统或者是 App 等是否能在不同的浏览器、系统版本、手机、pad 等各种终端都能够正常运行，一般关注主流的即可

8、性能测试，本次功能根据实际用户体量是否有并发的场景，核心业务、促销活动、复杂逻辑算法等，这些都有可能引起 Cpu、内存、I/O、带宽、数据库等性能问题，这个是需要提前预判的，因为一般出了性能问题都是大问题，如果用户体量很小，可以暂时忽略

8、安全测试，一般是用专业的工具对系统进行扫描，检查系统权限、网络、端口、敏感字、加密信息、配置管理等是否有安全隐患

9、易用性测试，即开发的产品是否通俗易懂，容易操作

10、回归测试，以上测试都完成之后，bug 修复完，需要对系统进行一个全量的测试，至少相关的功能点都要去执行一下。

11、通过 jacoco 等开源白盒测试技术统计代码覆盖率。

以上就是写测试用例或者是界定测试范围的的思路，基本适用于更多场景。当然这样也不能保证 100%覆盖，只能说，做到以上几点，基本覆盖 80-90%，

如果还有 bug 就是你认知以外的东西，做到随时差缺补漏到自己的测试用例中。

43. App 发布上线测试人员都具体做什么？

参考答案:

1，首先对 ios、android 生产环境（有些公司有 UAT 环境）打包验证新功能，包括老功能是否受到影响。

2，验收测试通过的生产包渠道审核：IOS 提交 App Store，Android 打包提交各大应用商店，如小米商店，华为商店，应用宝等；

安卓会审核得比较快，IOS 一般一到三天。

3，通过主要渠道下载线上包，协助产品进行验收测试。

4，收集用户的反馈信息，跟进线上 BUG、埋点数据。

44. 如何测试椅子

一、功能测试（Function test）

1.能否坐人，能坐下几个人

2.能放哪些东西，称重量多少

3.能否挂东西

4.是否有按摩功能

5.是否有弹簧减震

6.能否移动

7.椅子能不能折叠起来

8.能不能旋转，能不能平躺

9.能不能调节高度

10.是否有背靠

11.是否有颈靠

12.是否有扶手

二、界面测试（UI Test）

1.颜色怎么样，是否好看。

2.材质是什么

3.形状怎么样

4.大小是否与客户要求的一致

5.椅子有几条腿、几个转轮

6.是硬垫 还是软垫

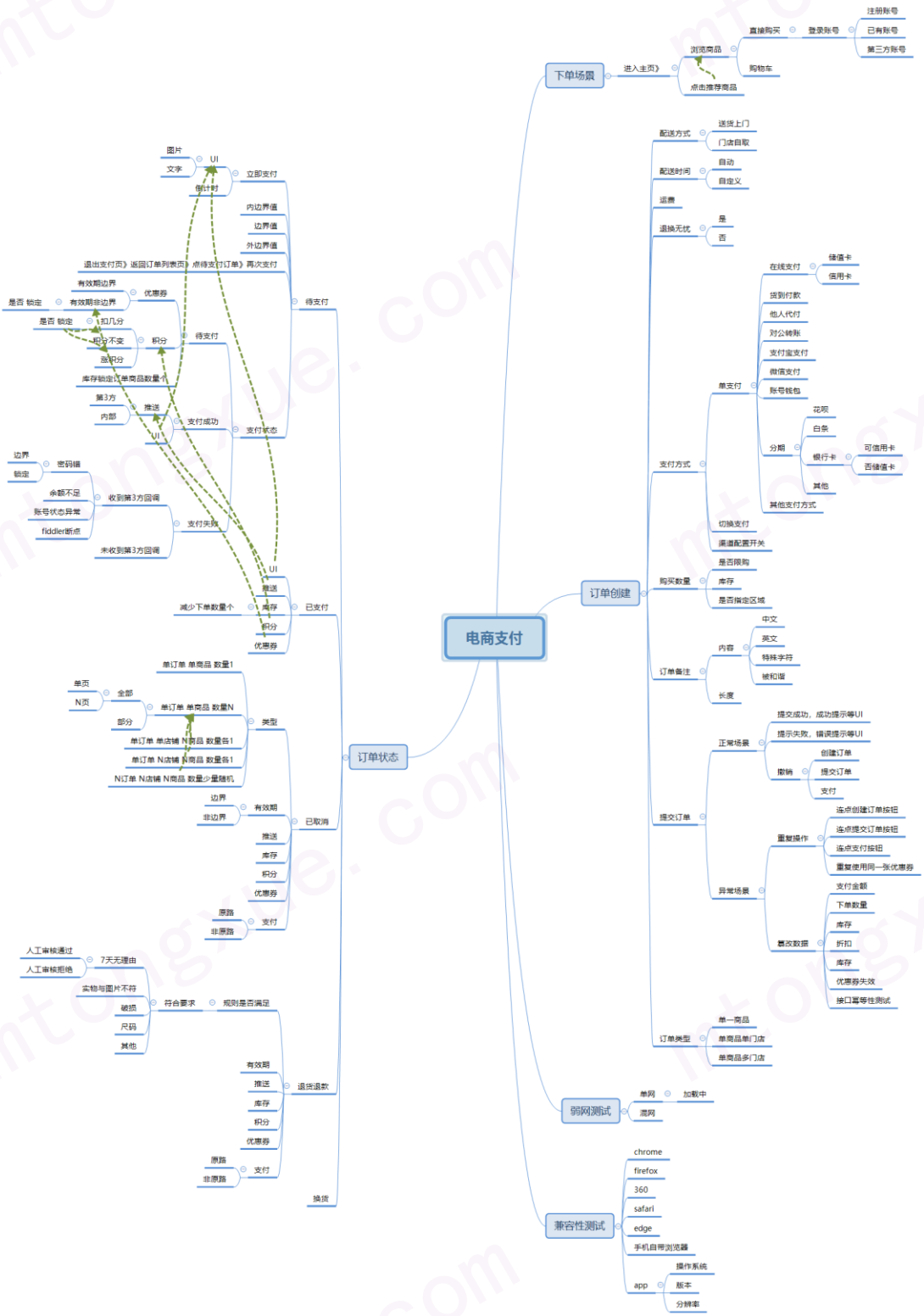
三、性能测试 (performance test)
1.能否持续承受高温或低温
2.防水效果怎么样
3.防火效果怎么样
4.椅子能挂多少东西
5.它的舒适程度怎么样
6.椅子的减震程度怎么样
7.能折叠或者旋转到什么程度
8.椅子的颜色或者图案在摩擦时是否会脱落

四、安全性测试 (Security test)
1.有没有锋利的边缘，会不会割伤人
2.如果倒下会不会砸伤人
3.坐在一侧会不会摔倒
4.地面不平的话会不会移动和摇晃
5.如果有裂痕会不会割伤人
6.材质是否对人体有害，包括散发的气味会不会让人感到不适
7.弯到什么程度会损坏

五、易用性测试 (Usability Test)
1.能应用于什么场合
2.椅子的移动是否方便
3.椅子的折叠或者旋转是否流畅
4.携带是否方便
5.有没有做防滑处理
6.有没有做防磕碰处理
7.是否符合人体生理
8.是否有透气性处理
9.是否有保暖性处理
10.和桌子高低匹配度
11.保养期多久一次（例如皮质、木质）
12.是否组装、拆卸方便

45. 设计电商（商城）支付测试点

参考答案：



46. 做了哪些推动项目的事情？

参考答案：

（1）及时协调沟通，当产品和开发、产品和产品、开发和开发直接出现需要协调

情况，尽快反馈问题，给出解决方案建议

（2）把关需求变更，包括：需求变更合理性、时间点、测试时间调整。

47. 公司的测试机少怎么办？

参考答案：

（1）临时征用借用自己、周围同事、亲朋好友

（2）不同测试业务线交叉使用测试机

（3）如果是高频使用机型走申请购买流程

（4）云测平台（weTest 、testIN）

（5）模拟器（不建议）

48. 你最近这家公司测试机有多少？

参考答案：

（1）小公司一般十几台，例如：苹果 5~6 台，安卓 10+台

（2）大公司测试机很多，兼容性测试更深一点，例如：苹果 30+台，安卓 100+台

49. 遇到的印象最深的 BUG 是？

答题思路：

（1）可以是体现 BUG 定位问题能力的 BUG

（2）可以是体现测试思维模式扩展的 BUG

（3）可以是体现相关技能提升的 BUG

（4）可以是业务逻辑复杂的 BUG

（5）可以是需求分析/评审阶段发现的用户体验类（易用性）的 BUG

（6）可以是需求分析/评审阶段发现的由于需求未明确而存在争议的 BUG

（7）可以是线上严重级别较低但用户体验效果非常差的 BUG

参考答案：

（1）我印象最深的一个 BUG，是一个线上 BUG，这个 BUG 的严重级别是 P4，用户体验级别的 BUG。领红包活动，产品设计了一个领红包的 icon 放在那里。在测试的时候没有发现问题，包括产品、开发的所有人都没有发现问题。

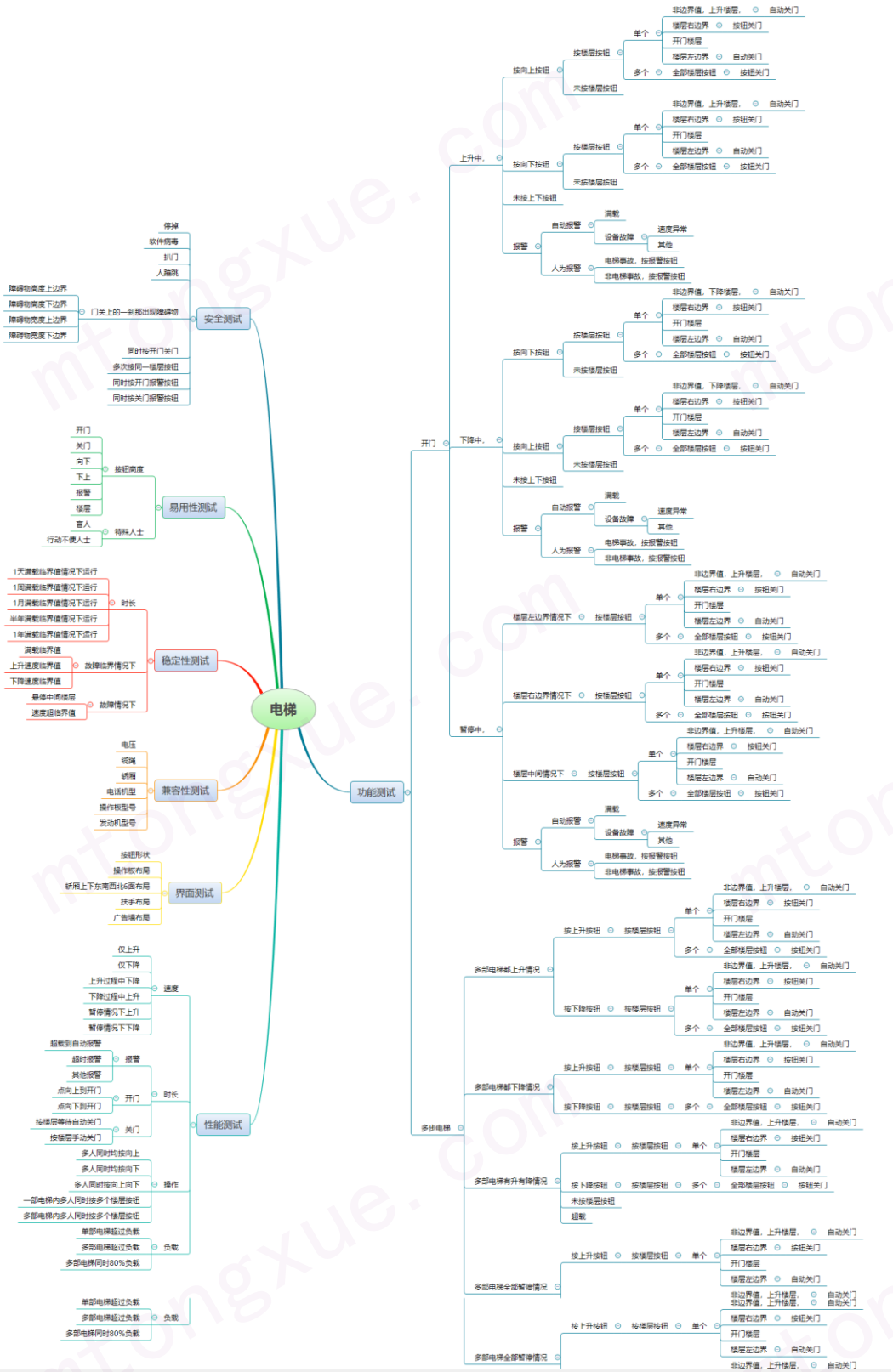
（2）但是上线后客服这边收到用户投诉说首页卡死了，询问用户的操作步骤得知，用户打开 APP，习惯性的认为首页展示推荐商品、搜索框这些信息，突然增加的领红包的 icon，用户不敢点，退出 app 重新进入仍然停在领红包的 icon，这就是用户说的首页“卡死”。

（3）这个 BUG 严重级别很低，但是用户反感度很高。产品、测试、开发的认知以及惯性思维是点 领红包的 icon，没有考虑到用户不点的情况。后面优化处理方案是 2 处，icon 停留时间 5 秒自动消失，icon 右上角增加删除×按钮图标。

（4）深刻反思是，对于用户遇到的问题，不仅仅要从测试的角度去发现 bug，还要从用户的角度去发现 bug。

50. 设计电梯测试点

参考答案：



51. bug 如何提交记录和 bug 的要素有哪些？

参考答案：

1.提交到 bug 管理工具，例如：禅道/jira/公司自研用例管理平台

2.包括 bug 标题以及 bug 描述（前提条件，操作步骤，预期结果，实际结果），所属模块，所属版本，bug 严重程度，提交所对应的开发）。

二．数据库相关

52. 用一条 SQL 语句 查询出每门课都大于 80 分的学生姓名。表 scores 如下：

name	course	score
张三	语文	81
张三	数学	75
李四	语文	76
李四	数学	90
王五	数学	100
王五	英语	90

参考答案:

学生的最低分数大于 80，那么就可以查询出每门课都大于 80 分的学生姓名

```
select name,min(score) from scores group by name having min(scor
e)>80);
```

53. 用一条 SQL 语句 查询两门以上不及格课程的同学的学号以及其平均成绩，并按成绩排序表结构如下：如下：

Student 表：

id	stdentname
1	张三
2	李四
3	王五

student_score 表：

id	coursename	score
1	语文	78
1	数学	50
2	语文	56
2	数学	58
3	语文	78
3	数学	47

参考答案:

```
select student.id, student.stdentname, AVG(student_score.score) from student,student_score
where
    student.id = student_score.id and student_score.score<60
```

```
group by student_score.id
having count(*)>1;
```

54. 什么是事务？

事务：是数据库操作的最小工作单元，是作为单个逻辑工作单元执行的一系列操作；这些操作作为一个整体一起向系统提交，要么都执行、要么都不执行；事务是一组不可再分割的操作集合（工作逻辑单元）；

1、原子性(Atomicity)：事务中的全部操作在数据库中是不可分割的，要么全部完成，要么均不执行。

2、一致性(Consistency)：几个并行执行的事务，其执行结果必须与按某一顺序串行执行的结果相一致。

3、隔离性(Isolation)：事务的执行不受其他事务的干扰，事务执行的中间结果对其他事务必须是透明的。

4、持久性(Durability)：对于任意已提交事务，系统必须保证该事务对数据库的改变不被丢失

55. SQL 中常用的聚合函数都有哪些？

参考答案：

max()：最大值
min()：最小值
avg()：平均值
sum()：求和
count()：统计总数

56. 主键、外键和索引的区别

1）定义
主键：唯一标识一条记录，不能有重复的，不允许为空
外键：表的外键是另一表的主键，外键可以有重复的，可以是空值
索引：该字段没有重复值，但可以有一个空值
2）作用
主键：用来保证数据完整性
外键：用来和其他表建立联系用的
索引：提高查询排序的速度
3）个数
主键：只能有一个
外键：一个表可以有多个外键
索引：一个表可以有多个索引

57. drop、delete、truncate 三者的区别

都表示删除，但是三者有一些差别，

Delete 用来删除表的全部或者一部分数据行，执行 delete 之后，用户需要提交(commmit)或者回滚(rollback)来执行删除或者撤销删除。会触发这个表上所有的 delete 触发器

Truncate 删除表中的所有数据，这个操作不能回滚，也不会触发这个表上的触发器，TRUNCATE 比 delete 更快，占用的空间更小；

Drop 命令从数据库中删除表，所有的数据行，索引和权限也会被删除，所有的 DML 触发器也不会被触发，这个命令也不能回滚。

58. InnoDB 索引和 MyISAM 索引的区别，索引的优缺点

参考答案:

1）存储结构（主索引 / 辅助索引）

InnoDB 的数据文件本身就是主索引文件。而 MyISAM 的主索引和数据是分开的。

InnoDB 的辅助索引 data 域存储相应记录主键的值而不是地址。

而 MyISAM 的辅助索引和主索引没有多大区别。

innoDB 是聚簇索引，数据挂在逐渐索引之下。

2）锁: MyISAM 使用的是表锁；InnoDB 使用行锁

3）事务: MyISAM 没有事务支持和 MVCC；InnoDB 支持事务和 MVCC

4）全文索引: MyISAM 支持 FULLTEXT 类型的全文索引;InnoDB 不支持 FULLTEXT 类型的全文索引，但是 InnoDB 可以使用 sphinx 插件支持全文索引，并且效果更好

5）主键: MyISAM 允许没有任何索引和主键的表存在，索引都是保存行的地址; InnoDB 如果没有设定主键或非空唯一索引，就会自动生成一个 6 字节的主键，数据是主索引的一部分，附加索引保存的是主索引的值

6）外键: MyISAM 不支持；InnoDB 支持

59. 列举几种表连接的方式,有什么区别?

参考答案：

左连接: 左边为主表表数据全部显示, 匹配表的不匹配部分不显示

右连接: 右边为主表表数据全部显示, 匹配表的不匹配部分不显示

内连接: 只有两个元素表相匹配的才能在结果集中显示

全外连接: 连接中的不匹配的数据全部会显示出来

交叉连接: 笛卡尔乘积, 显示的结果是连接表数的乘积

60. 悲观锁和乐观锁的区别？

为什么需要锁？

在多用户环境中，在同一时间可能会有多个用户更新相同的记录，这会产生冲突。这就是著名的并发性问题。

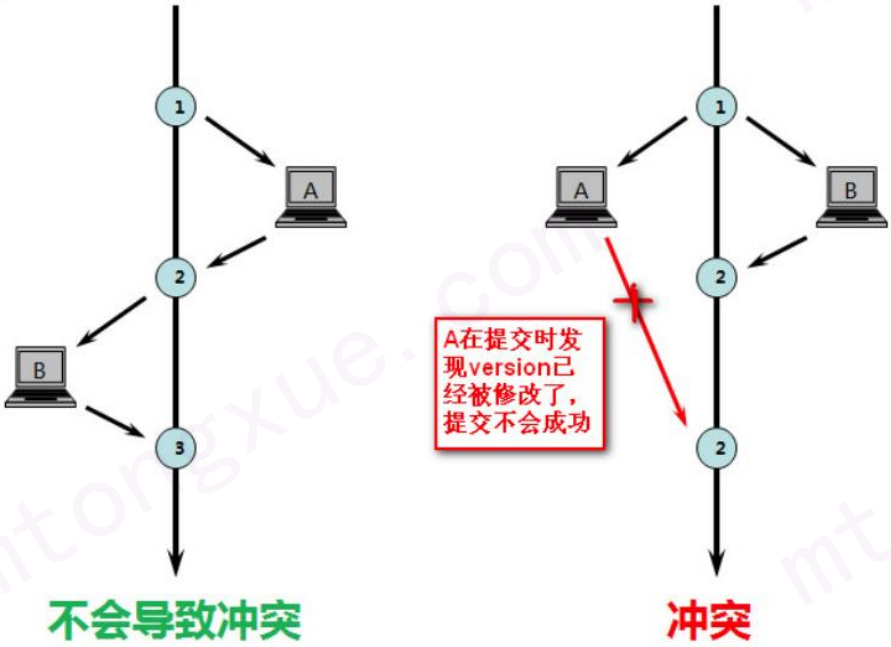
并发控制机制

悲观锁：假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作

乐观锁：假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性

乐观锁原理

使用数据版本（Version）记录机制实现，这是乐观锁最常用的一种实现方式。何谓数据版本？即为数据增加一个版本标识，一般是通过为数据库表增加一个数字类型的“version”字段来实现。当读取数据时，将 version 字段的值一同读出，数据每更新一次，对此 version 值加一。当我们提交更新的时候，判断数据库表对应记录的当前版本信息与第一次取出来的 version 值进行比对，如果数据库表当前版本号与第一次取出来的 version 值相等，则予以更新，否则认为是过期数据。用下面的一张图来说明：



悲观锁原理

需要使用数据库的锁机制，比如 MySQL，

1. 先关闭 auto commit 属性，改为手动提交事务

2. 在事务内通过 select...for update 语句锁定数据，

如：select * from t_goods where id=1 for update

3. 此时在 t_goods 表中，id 为 1 的 那条数据就被我们锁定了，其它的事务必须等本次事务提交之后才能执行。这样我们可以保证当前的数据不会被其它事务修改

61. 工作中的哪些方面会使用到数据库？如何使用？

参考答案：

（1）查询数据（查看数据的准确性）

比如卡里有 1000 元，商品 300 元。下单购买成功，提示扣费成功。

查看数据库库存是否从 300（原库存）变为 299，查看卡里的钱是否变为 700。

（2）造数据（方便测试）

比如参加领券活动，一个用户只能领一次。如果准备多个手机号测试会很麻烦。用券有一个属性字段，如果参加过了就设置字段为 0，没参加过就设置字段为 1。测试 1 次，字段由 1 变为 0。下次再测试，可以将字段改为 1 来继续测试。

（3）初始化、备份还原数据

62. 请按照如下题目要求编写 SQL

部门表 departments（dept_id, dept_name）
员工表 employees（emp_id， emp_name, sex, dept_id, jobs）
薪水表 salary (salary_id, emp_id, money)

一、列出总人数大于 4 的部门编号和该部门人数

```
select dept_id, count(*) from employees e group by dept_id having count(*)>4
```

二、列出开发部和测试部的职工号，姓名

```
select e.emp_id,d.emp_name
from employees e inner join department d on e.dept_id = d.dept_id
where d.dept_name in ('开发部','测试部')
```

三、显示工资最高的前 3 名职工的职工号和姓名

```
select e.emp_id, e.emp_name,s.money
from salary s inner join employees e on s.emp_id = e.emp_id
order by s.money desc
limit 3
```

四、列出工资在 1000-2000 之间的所有职工姓名

```
select e.empname,s.salary
from salary s
inner join employees e on s.empid = e.empid
where s.salary between 1000 and 2000
```

三. linux 及测试环境相关

63. 说几个工作中常用的 Linux 命令？

考察点：

1. 考察是否有 Linux 使用经验
2. 考察对 Linux 的熟悉程度

参考答案：（建议功能 1-3 年）

cd：切换目录

ls：查看文件列表

cp：拷贝文件

mv：移动文件

rm：删除文件

chmod：设置文件权限

cat：浏览文件内容

vi：文件编辑

find：搜索文件

grep：过滤文件内容

64. 用过 docker 吗？常用的 docker 命令有哪些？

参考答案：

docker pull：拉取镜像

docker images：查看本地镜像

docker run：运行镜像为容器

docker ps：查看正在运行的容器

docker logs：查看容器日志

docker cp：拷贝容器文件

docker start/stop/restart：启动、停止、重启容器

65. 在 shell 环境如何杀死一个进程？

考点：

1. linux 下进程 PID 查找命令
2. linux 停止进程命令

参考答案：

比如 tomcat，查看 tomcat 的 PID 命令如下所示：

ps -ef|grep tomcat

比如 tomcat 的 PID 是 12345，停止 tomcat 进程命令如下所示：

kill -9 12345

66. 如何查找当前目录下大于 10K 的文件

参考答案:

查找目录命令 find

当前目录命令.

文件类型参数-type
文件大小参数-size
find . -type f -size +10k

67. linux 下查看/web.log 第 25 行第三列的内容

常用的三种实现方式如下所示

```
sed -n 25p /web.log | cut -d " " -f3
head -n25 /web.log | tail -n1 | cut -d " " -f3
awk -F " " 'NR==25{print $3}' /web.log
```

68. linux 下修改 test.txt 的 23 行 test 为 TEST

参考答案：

1. 修改文件命令 sed
2. 直接修改文件内容参数 i
3. 指定行替换参数 s 加 g

```
sed -i 23s/test/TEST/g test.txt
```

69. 在 Linux 中如何查找日志文件中的 Error 信息

参考答案：

Linux 中通过 grep 命令可以过滤文本文件中的指定信息，如

```
grep "Error" access.log
```

以上命令会将 access.log 中包含 Error 的行打印出来

70. 给/home/demo.txt 文件设置为所有人可读可写权限

```
chmod 666 /home/demo.txt
```

71. 搜索 a.log 文件中包含 Exception 的日志以及其后 10 行

```
grep -A 10 “Exception” a.log
```

72. 查看 mysql 进程是否启动成功

```
ps -ef | grep mysql
```

73. 在/home 目录下搜索 mysql.log 的存放目录

find /home -name mysql.log

74. 说一下常用的 10 个 linux 命令

参考答案：（建议性能、自动化 1-3 年）
head -2 file1 查看一个文件的最前面两行
tail -2 file1 查看一个文件的最后两行
tail -f /var/log/menssage 实时查看被添加到一个文件中的内容
ls 查看目录下文件
ls -l 查看文件和目录的详细信息
ls -a 显示隐藏文件
mkdir dir1 创建一个叫做 dir1 的目录
mv dir1 new_dir2 移动或重命名一个目录
top 查看进程的信息
cat -n file1 标识文件的行数

75. 如何查看 3306 端口号是否被占用

netstat -anp | grep 3306
或
lsof -i 3306

76. 如何查看 a.log 中 zhangsan 用户昨天的操作日志

grep ‘zhangsan’ a.log | grep ‘昨天日期’

77. 关闭防火墙的命令是什么

systemctl stop firewalld

78. 如何统计 a.log 中有多少个 Exception

grep ‘Exception’ a.log | wc -l

四. 自动化测试和性能

79. 如何查看安卓 app 界面的 activity 和起始 activity

参考答案：

1. 使用 pm 管理器查看包名

adb shell pm list package -f

将获取手机内所有 apk 对应的包名和路径

如果是 windows 可以后接 findstr，例如：

adb shell pm list package -f | findstr douban

2. 使用 am 管理器，查看包名

adb shell am monitor

3. 使用 aapt 查看包名、界面名

aapt 是 sdk 自带的一个工具，在 sdk\builds-tools\目录下 以豆瓣 App 为例：

1.把安装包放到 aapt 目录下

2.从 cmd 中切换目录到 sdk\builds-tools\下

3.执行 aapt dump badging

运行后的结果中以下两行分别是应用包名 package 和入口 activity 名称

package:name='com.tencent.qqmusic'

launchable-activity: name='com.tencent.qqmusic.activity.AppStarterActivity'

80. appium 的工作原理是什么

参考答案：

appium 工作原理需要分很多方面进行阐述

安卓：

1.1 appuim 基于 uiautomator2 的原理

appium 服务启动后默认在 4723 端口上创建一个 http 服务，脚本通过服务地址 http://xxxx:4723/wd/hub 和 appium 进行通信

在初始化脚本和 appium 连接的过程中 appium 会向手机就安装辅助 app uiautomator2.server.apk 和 uiautomator2.server.test.apk，并且做端口转发 adb forward tcp 8200 tcp 6790,安装以后会在手机上启动 uiautomator2 的 server，这个 server 启动后会在手机上创建一个 netty server，端口是 6790，appium 和手机上的 uiautomator2 server 的 6790 端口进行通信，把从 4723 端口收到的脚本指令通过 8200 端口转发到手机的 6790 端口上

1.2 appium 基于 uiautomator1 的原理

Appium 服务启动后默认在 4723 端口上创建一个 http 服务，脚本通过服务地址 http://xxxx:4723/wd/hub 和 appium 进行通信

在初始化脚本和 appium 连接的过程中 appium 会向手机发送 AppiumBootstrap.jar，并

且做端口转发 adb forward tcp 4724 tcp 4724,安装以后会在手机上启动 AppiumBootstrap.jar，启动后会在手机上创建一个 socket 服务，端口是 4724，appium 和手机上的 socket 服务的 4724 端口进行通信，把从 4723 端口收到的脚本指令通过 4724 端口转发到手机的 4724 端口上

1.3 appium 基于 chromedriver 的原理，测试 H5 时使用

appium 服务启动后默认在 4723 端口上创建一个 http 服务，脚本通过服务地址 http://xxxx:4723/wd/hub 和 appium 进行通信

在初始化脚本和 appium 连接的过程中会启动 chromedriver 创建一个 http 服务，端口是 8000，appium 和 chromedriver 的服务通过 8000 端口进行通信，chromedriver 服务接收到 appium 指令后去操作手机，操作完成再返回给 appium，appium 再返回给脚本

IOS 手机：

Appium 服务启动后默认在 4723 端口上创建一个 http 服务，脚本通过服务地址 http://xxxx:4723/wd/hub 和 appium 进行通信

在初始化脚本和 appium 连接的过程中会向手机编译安装 webdriveragent app，并且启动 wda 在手机上创建一个基于 8100 的 http 服务，appuim 通过 4723 的端口接收到脚本传递的指令，appium 再通过本地的 8100 端口将收到的指令转发给手机上 8100 wda 服务，wda 服务接收到指令再去操作待测 app，操作完成后返回给 appium 操作结果，appium 再将结果返回给脚本

81. pytest 单元框架里，前置条件怎么处理？

1.使用 setup

函数级的（setup_function、teardown_function）只对函数用例生效，而且不在类中使用

类级的（setup_class、teardown_class）在类中使用，类执行之前运行一次，类执行之后运行一次

类中方法级的（setup_method、teardown_method）在每一个方法之前执行一次，在每一个方法之后执行一次

模块级的（setup_module、teardown_module）

2.使用 conftest

@pytest.fixture(scope="module")

scope 参数的可选范围：

-function：每一个函数或方法都会调用

-class：每一个类调用一次，一个类中可以有多个方法

-module：每一个.py 文件调用一次，该文件内又有多个 function 和 class

-session：是多个文件调用一次，可以跨.py 文件调用，每个.py 文件就是 module

82. pytest 参数化怎么实现

参考答案：

使用@pytest.mark.parametrize 装饰器

范例：

@pytest.mark.parametrize('字符串形式接收参数名', [(参数 1-1, '参数 2-1'), (参数 1-2, '参数 2-2')],ids=['第 1 条参数对应的用例名', '第 2 条参数对应的用例名'])

@pytest.mark.parametrize('goods_id,stock,exp', [(12, 1, '缺失规格'), ('商品编号', '1', '商品不存在或已删除')],ids=['不填写规格参数加购', '商品编号为异常值'])

83. pytest 里如何进行 case 的组装

参考答案：

1.默认使用检查以 test_ .py 或**test.py 命名的文件名，在文件内部查找以 test 打头的方法或函数，并执行

2.可以使用自定义 marker（标签），比如 pytest 运行的时就只运行带有该 marker 的测试用例，比如下面的@pytest.mark.P0

3.在命令行使用 指定文件

4.参数：-k args 模糊匹配 case（关键字 args：可以是 py 文件名，也可以是函数名）

84. 说说 pytest 里的钩子函数

参考答案：

几个常用的钩子：

pytest_configure(config):添加自定义的标签等

pytest_collection_modifyitems(items)：在 case 收集后调用，可以对项目顺序或其他功能进行自定义

pytest_addoption(parser):为命令行添加自定义参数

85. 什么是 P0 模式，Po 模式的优点是什么？

参考答案：

页面对象模型(PageObject)是一种设计模式,用来编写和维护自动化测试

Po 模式的优点：

1、PO 提供了一种业务流程与页面元素操作分离的模式，这使得测试代码变得更加清晰。

2、页面对象与用例分离，使得我们更好的复用对象。

3、可复用的页面方法代码会变得更加优化

4、更加有效的命名方式使得我们更加清晰的知道方法所操作的 UI 元素

如何编写 PO 模式

1.抽象每一个页面

2.页面中元素不暴露,仅报错操作元素的方法

3.页面不应该有繁琐的继承关系

4.页面中不是所有元素都需要涉及到,核型业务元素做建模使用

5.把页面划分功能模块,在 Page 中实现这些功能方法

86. 说说接口 case 的设计思路

首先是分析接口：

1.分析接口对应的业务
2.分析接口的参数、有几种响应、字段数据的来源
3.分析接口之间的依赖关系
4.分析接口对应的业务依赖关系
然后根据信息，可以从以下方向设计用例：
接口业务
接口参数单项测试
业务依赖
接口依赖
数据依赖
接口安全
接口性能

87. mock 技术一般用在什么场景，简述 mock-server 的设计思路

使用场景：
①需求提测后，后端未提测 但是 前端完成。使用 mock 验证前端
②业务的自动化项目中，有个别接口不稳定。使用 mock 代替
设计思路：
使用 flask 编写 mock-web 服务
①按需实现 mock 逻辑
②针对不需要实现 mock 的业务，使用通用透传方法处理
③mock 响应数据，应该抽离出来。方便维护

88. 你是怎么测试接口的

先了解接口的业务功能、入参出参以及接口对应的数据存储，再依据接口测试用例设计方法完成接口测试的设计，用例设计先业务场景再参数判断，比如参数的边界值、格式、组合等等，最后依据测试用例使用接口测试工具完成接口测试，并在测试过程中查看日志及数据以确保接口测试结果的正确性

89. ui 自动化中定位不到元素的原因有哪些

参考答案：
1) 定位器选择错误
2) 定位字符串错误
3) 元素嵌套在 frame 当中
4) 页面元素没有及时加载
5) 元素在新窗口中
6) 脚本流程与实际不符
7) 元素不在当前页

90. 如何保证自动化测试的稳定性

参考答案：

a. 避免使用固定的数据，测试用例中使用老的测试数据，可能会被别人修改或删除。所以每次跑脚本前，在脚本中构造新的数据，跑完脚本后，把数据清理掉。
b. 降低用例之间的耦合性，每个用例尽量都走完整的流程，不要依赖于其他用例，避免其他用例执行失败，影响了后续的用例。
c. 提升依赖环境的稳定行，通常某些用例会依赖第三方系统的环境，如果第三方环境不稳定，会造成用例执行的不稳定。可以采用 mock 的形式，屏蔽第三方环境，提升环境稳定性。
d. 脚本的异常处理，在脚本中要多考虑可能出现的异常，尽量对每种异常都有对应的处理方法，避免失败后程序退出

91. 接口测试中的加密参数如何处理

参考答案：

首先了解参数的加解密方式，常见的有 md5、aes、rsa 等等，如果是 aes 的需要找开发要私钥，如果是 rsa 需要找开发要公钥和私钥，然后在接口测试工具中引用加解密的代码实现参数的加解密过程，实现参数加解密的处理；如果公司有自定义的加密算法则需要找开发要加解密的代码实现，然后在测试工具中使用。

92. 使用 jmeter 如何做接口之间的关联

接口关联指的就是一个接口要使用另一个接口的返回值作为参数，在 jmeter 中针对不同的响应数据格式都有不同的处理组件，json 格式的采用 json 提取器，xml 或者 html 格式的采用 xpath 提取器，其他格式的可以采用正则表达式提取器，BeanShell 后置处理器也可以从响应结果中提取响应内容，通过这些组件提取所需内容后，在需要关联的接口中引用变量即可完成关联

93. web 自动化中如何处理 alert 弹窗

selenium 里提供了 switch_to.alert 方法来处理弹窗，处理代码如下（Python）

```
#切换到 alert 窗口  alert = driver.switch_to.alert
#点击确定  alert.accept()
```

94. 说一下你知道的 HTTP 状态码，以及它们代表什么意思

200（请求成功）

302（重定向）

307（重定向，服务器要求客户端重新请求一个新的 URL）

400（Bad Request/错误请求）

401（Unauthorized/未授权，需要身份认证）

403（Forbidden/服务端禁止访问）

404（Not Found/未找到，请求的资源未找到，比如 url 写错了，页面被删除等）

405（Method Not Allowed/方法未允许）

415 不支持的媒体类型

500（Internal Server Error/内部服务器错误）

502（Bad Gateway/错误的网关）

503（Service Unavailable/服务无法获得）

504（Gateway Timeout/网关超时）

95. 说一下 DNS 解析流程

参考答案：
1. 浏览器先检查自身缓存中有没有被解析过的这个域名对应的 ip 地址，如果有，解析结束。同时域名被缓存的时间也可通过 TTL 属性来设置。
2. 如果浏览器缓存中没有（专业点叫还没命中），浏览器会检查操作系统缓存中有没有对应的已解析过的结果。而操作系统也有一个域名解析的过程。在 windows 中可通过 c 盘里一个叫 hosts 的文件来设置，如果你在这里指定了一个域名对应的 ip 地址，那浏览器会首先使用这个 ip 地址。
3. 如果至此还没有命中域名，才会真正的请求本地域名服务器（LDNS）来解析这个域名，这台服务器一般在你的城市的某个角落，一般都会缓存域名解析结果，大约 80% 的域名解析到这里就完成了。
4. 如果 LDNS 仍然没有命中，就直接跳到 Root Server 域名服务器请求解析。
5. 根域名服务器返回给 LDNS 一个所查询域的主域名服务器（gTLD Server，国际顶尖域名服务器，如.com .cn .org 等）地址
6. 此时 LDNS 再发送请求给上一步返回的 gTLD
7. 接受请求的 gTLD 查找并返回这个域名对应的 Name Server 的地址，这个 Name Server 就是网站注册的域名服务器
8. Name Server 根据映射关系表找到目标 ip，返回给 LDNS
9. LDNS 缓存这个域名和对应的 ip
10. LDNS 把解析的结果返回给用户，用户根据 TTL 值缓存到本地系统缓存中，域名解析过程至此结束

96.同步和异步区别

参考答案：

同步和异步是一种通讯方式

同步：执行一个操作时，需要等待其处理完成，然后再进行下一个操作

异步：执行一个操作时，不需要等待返回，就进行下一个操作，一般需要使用消息中间件

举例：

下单接口中，需要调用库存接口做库存判断，所以必须等待库存接口返回数据才能进行下一步操作，这是同步；

下单接口中，下单成功后需要调用邮件通知接口，不用等待接口返回成功，就可以直接进行下一步操作，这是异步

97. Tcp 三次握手流程

参考答案：

第一次握手：建立连接时，客户端发送 syn 包（syn=j）到服务器，并进入 SYN_SENT 状态，等待服务器确认

第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态

第三次握手：客户端收到服务器的 SYN+ACK 包，向服务器发送确认包 ACK(ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED（TCP 连接成功）状态，完成三次握手。

98. 如何模拟弱网测试

参考答案：

很多抓包工具都可以做到模拟网络情况，比如 fiddler、charles。如果是网站还可以采用 chrome 开发者工具模拟弱网，如果是手机 app 则可以在手机自身的网络设置里设置为 2G/3G/4G/飞行模式。

99. 自动化测试在什么阶段执行会带来什么收益

参考答案：

对于接口测试来说均需要采用自动化的手段，因此在开发联调完成和集成测试之间可以介入，在进入集成测试之后接口测试自动化脚本可以用来做接口层面的回归测试，以保障接口方面不出问题；UI 自动化会在产品稳定之后介入，主要用来保障 UI 层面的回归，那么自动化测试均是为了提升测试效率，保障原有功能的正确性

100. 自动化测试框架都包括哪些模块

自动化测试框架模块包括基础方法、数据驱动、PO 模式分层、自定义异常、工具包、配置文件、测试报告、日志收集、关键字驱动、接口分层、接口数据管理等模块，以上模块均需要集成相关的工具进行二次封装。

101. 性能测试中，TPS 比较低，可能是哪些方面的问题？

参考答案：

1. 压力机本身性能瓶颈
2. 网络 IO 瓶颈
3. 中间件（tomcat/nginx/mysql）连接数限制
4. Java 线程的阻塞、等待
5. 本系统资源的瓶颈（cpu、内存、磁盘、网络等）
6. 其他外部系统响应时间过长，造成本系统的 time-wait

102. 性能测试脚本中为什么要做参数化

参数化是将请求中的参数动态化，每次请求都使用不一样的参数，从而模拟了不同用户使用不同数据来做业务请求

常用的参数化方式有：随机数、随机字符串、时间戳、文件参数化、UUID 等

103. 如何准备性能测试数据

参考答案：

a)调用业务接口构造数据，一般适用于数据逻辑比较复杂的情况下。

b)直接写 jdbc 代码造数据，一般适用于数据量较大且数据逻辑较简单的情况。

c)存储过程造数据，一般适用于数据量巨大且数据逻辑较简单的情况。

d)导入 sql，一般适用于数据安全级别较低且数据量巨大的情况。

104. 性能测试过程中如何对瓶颈进行分析？

参考答案：

性能瓶颈分析参考准则：排除法，从上至下、从局部到整体！

针对不同的瓶颈采用不同的分析方法，一般分为： 内存分析方法、处理器分析法、磁盘 I/O 分析方法、进程分析方法、网络分析方法等等。

内存分析方法：内存分析用于判断系统有无内存瓶颈，是否需要通过增加内存等手段提高系统性能表现。

处理器分析法：通过处理器性能计数器的值体现服务器整体处理器利用率，判断是否存在处理器瓶颈。

磁盘 I/O 分析方法：通过磁盘 I/O 性能计数器的值体现服务器整体磁盘 I/O 使用情况，判断是否存在处理器瓶颈。

进程分析方法：通过进程性能指标数据，判断是否存在进程瓶颈。

网络分析方法：通过网络性能指标数据，判断是否存在网络瓶颈。

105. get 和 post 的区别

参考答案:

最常见的 http/https 协议的请求方式是 GET 和 POST，他们之间的区别如下：

1.请求参数的位置

get 在 url 里面传输，post 在请求体里传输

2.安全性

正因为传输参数位置的区别，所以 get 的安全性不如 post

3.传输数据的大小

正因为传输参数位置的区别，get 的参数长度受限于浏览器，post 的长度不受限制

4.表现形式

请求行的表现形式不一样，因为请求行里面包括请求方法，所以当然不一样了

106. http 和 https 的区别

参考答案:

HTTP:超文本传输协议，是一个客户端和服务器端的请求和应答的标准。

HTTPS:是以安全为目标的 HTTP 通道，HTTP 的安全版本，HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL,因此加密的详细内容就需要 SSL.

他们的区别如下：

1. HTTP 信息是明文传输的，而 HTTPS 是安全的 具有安全性的 ssl 加密传输

2. HTTP 标准端口是 80 ，而 HTTPS 的标准端口是 443

3. HTTP 无需证书，而 HTTPS 需要认证证书.需要到 CA 申请证书，一般免费证书较少，因而需要一定费用。

107. cookie 和 session 的区别？

参考答案:

会话（Session）跟踪是 Web 程序中常用的技术，用来跟踪用户的整个会话

Cookie 通过在客户端记录信息确定用户身份，Session 通过在服务器端记录信息确定用户身份

区别：

1、数据存放位置不同：

cookie 数据存放在客户的浏览器上，session 数据放在服务器上。

2、安全程度不同：

cookie 不是很安全，别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗,考虑到安全应当使用 session。

3、性能使用程度不同：

session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能,考虑到减轻服务器性能方面，应当使用 cookie。

4、数据存储大小不同：

单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie，而 session 则存储与服务端，浏览器对其没有限制。

108. web ui 自动化测试中显式等待，隐式等待的区别

参考答案:

1. 相同点都是智能等待，在一定时间范围内不断查找元素，一旦找到立刻结束查找继续执行代码，没找到才会一直找到超时为止

2. 不同点是隐式等待是全局性设置，并且可以随时更改，在更改后对之后的 findxxx 方法生效，对点击、输入之类的操作不起作用；

显式等待仅仅针对单一元素或一组生效，并且不仅仅是针对查找，也可以针对 Alert、iframe，或者元素的某些属性进行自定义判断

109. 验证码的几种处理方式

参考答案:

针对验证码有如下方法：

1.在产品没有上线前，需要找开发先给 web 验证码留后门，也就是将验证码验证先注释掉

2.让开发给 web 验证码留一个万用验证码，只要输入给定的验证码，就可以强制登录

3.当有的页面可以勾选保存用户名，密码可以通过 Cookie 跳过登录验证码，使用抓包工具就行了

4.使用验证码识别技术

110. 进程和线程的区别

参考答案:

进程与线程的区别

进程是资源分配最小单位，线程是程序执行的最小单位；

进程有自己独立的地址空间，每启动一个进程，系统都会为其分配地址空间，建立数据表来维护代码段、堆栈段和数据段，线程没有独立的地址空间，它使用相同的地址空间共享数据；

CPU 切换一个线程比切换进程花费小；

创建一个线程比进程开销小；

线程占用的资源要比进程少很多。

线程之间通信更方便，同一个进程下，线程共享全局变量，静态变量等数据，进程之间的通信需要以通信的方式（IPC）进行；（但多线程程序处理好同步与互斥是个难点）

多进程程序更安全，生命力更强，一个进程死掉不会对另一个进程造成影响（源于有独立的地址空间），多线程程序更不易维护，一个线程死掉，整个进程就死掉了（因为共享地址空间）；

进程对资源保护要求高，开销大，效率相对较低，线程资源保护要求不高，但开销小，效率高，可频繁切换；

111. unittest 和 pytest 的区别

参考答案:

一、用例编写规则

1.unittest 提供了 test cases、test suites、test fixtures、test runner 相关的类,让测试更加明确、方便、可控。使用 unittest 编写用例,必须遵守以下规则:

（1）测试文件必须先 import unittest

（2）测试类必须继承 unittest.TestCase

（3）测试方法必须以“test_”开头

（4）测试类必须要有 unittest.main()方法

2.pytest 是 python 的第三方测试框架,是基于 unittest 的扩展框架,比 unittest 更简洁,更高效。使用 pytest 编写用例,必须遵守以下规则:

（1）测试文件名必须以“test_”开头或者"_test"结尾(如:test_ab.py)

（2）测试方法必须以“test_”开头。

（3）测试类命名以"Test"开头。

二、用例前置和后置

1.unittest 提供了 setUp/tearDown，每个用例运行前、结束后运行一次。setUpClass 和 tearDownClass，用例执行前、结束后，只运行一次

2.pytest 提供了模块级、函数级、类级、方法级的 setup/teardown，比 unittest 的 setUp/tearDown 更灵活。

三、断言

1.unittest 提供了 assertEqual、assertIn、assertTrue、assertFalse。

2.pytest 直接使用 assert 表达式。

四、报告

1.unittest 使用 HTMLTestRunnerNew 库。

2.pytest 有 pytest-HTML、allure 插件。

五、失败重跑

1.unittest 无此功能。

2.pytest 支持用例执行失败重跑，pytest-rerunfailures 插件。

六、参数化

1.unittest 需依赖 ddt 库，

2.pytest 直接使用@pytest.mark.parametrize 装饰器。

七、用例分类执行

1、unittest 默认执行全部用例，也可以通过加载 testsuit，执行部分用例。

2、pytest 可以通过@pytest.mark 来标记类和方法，pytest.main 加入参数("-m")可以只运行标记的类和方法。

112. 在你做自动化过程中，遇到了什么问题吗？举例下

参考答案:

这个问题，不管是自动化还是任何工作，都会被问到。主要想知道你是如何解决问题的，从而推断你问题分析和解决的能力。

当然有遇到问题和挑战，主要有以下几点：

频繁地变更 UI，经常要修改页面对象里面代码

运行用例报错和处理，例如元素不可见，元素找不到这样异常

测试脚本复用，尽可能多代码复用

一些新框架产生的页面元素定位问题，例如 ck 编辑器，动态表格等

113. 在 selenium 中如何处理多窗口？

这个多窗口之间跳转处理，在实际 selenium 自动化测试经常遇到。点击一个链接，这个链接会在一个新的 tab 打开，然后接下来要查找元素在新 tab 打开的页面，需要先将 driver 切换至 window，然后再定位，步骤如下

1. 先获取当前的 windowhandle
2. 操作打开新界面后，获取所有的 windowhandles
3. 遍历 windowhandles，判断和当前的 windowhandle 不一样则切换至该 windowhandle
4. window 太多则可以按照 title、url 等其他信息进行判断切换

114. 你查找元素遇到过在 iframe 里面吗?你是如何处理 iframe 里面元素定位的？

有时候我们知道元素定位表达式没有问题，但是还是提示 no such element，那么我们就需要考虑这个元素是否在 iframe 中，通过 f12 查看元素，如果有 iframe，需要先将 driver 切换到 iframe 中，可以通过 frame 的 name 和 id 和索引三种方法来定位 frame,或者先找到 iframe 的元素，然后把这个元素传递进去也可以。

115. webdriver 中关闭浏览器的 quit 和 close 有什么区别

简单来说，两个都可以实现退出浏览器 session 功能，

close 是关闭你当前聚焦的 tab 页面，而 quit 是关闭全部浏览器 tab 页面，并退出浏览器 session。

知道这两个区别，我们就知道 quit 一般用在结束测试之前的操作，close 用在执行用例过程中关闭某一个页面的操作

116. 性能测试的流程是什么？

参考答案：

需求调研-环境搭建-脚本编写-准备数据-执行测试-回归调优-测试报告

117. 性能场景怎么设计？一般都有哪些性能场景？

参考答案：

一般基本的场景包括：基准测试、单交易测试、混合测试、稳定性测试 其他场景的可选场景：高可用性测试、异常测试等，以及其他的结合各自项目业务的场景

118. 性能测试中，一般都关注哪些指标？

参考答案

TPS：每秒事务数，代表了性能的好坏，TPS 越高，性能越好

平均响应时间：请求的平均耗时，响应时间越短，性能越好

并发数：同时向服务端发起请求的虚拟用户数，在不同的工具里可以用多个进程/线程来实现

错误率：失败的请求比例

吞吐量：网络上行和下行流量的总和，吞吐量是网络瓶颈定位的重要指标

119. 什么是长连接，什么是短连接？

长连接和短连接是客户端和服务端之间的通信机制。

长连接：客户端和服务端建立连接后，后续无论进行多少次通信，所有的请求和响应数据都是在这个链接上进行，这就是长连接。

短连接：客户端每一次和服务端进行通信时，都重新创建一个链接，通信完成后关闭连接。

120. 网络七层模型都是哪七层，HTTP 协议是在哪一层，Tcp 协议在哪一层？

参考答案：

网络 OSI 七层模型：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层

其中，HTTP 协议处于应用层，TCP 协议处于传输层

121. 产品就只给一个需求，需求调研的内容都不知道，也没人告诉你，怎么开展性能测试

参考答案：

1. 没有任何途径进行需求调研的情况下，可以跳过需求调研，直接开始压测。

2. 压测完成后，可以把本次压测数据开会讨论，共同决定是否满足性能需求；

3. 或者根据行业内的通用指标规范，比如高频接口响应时间<100ms，低频<200ms 的标准来判断

122. 说说你对集合点的理解以及在项目中的应用

1. 集合点是测试脚本中的一个标记，当每个虚拟用户执行到标记处时，会停留在标记处等待其他的虚拟用户，当达到预期设置的并发数时，标记处的所有用户同时启动执行后续的请求

2. 集合点会产生瞬间高并发，但是也会降低平均压力。所以在压测过程中，如果有要求瞬间高并发的业务，就需要使用集合点，比如抢购，秒杀之类的业务。

3. 没有类似业务则不需要加集合点

4. lr_rendezvous（“集合点名称”），同步定时器

123. 性能测试中的思考时间应该怎么用

参考答案:

对于交互系统来说，用户在向服务器发送一个请求后，会等待一段时间后再发送下一个请求。

在性能测试过程中，使用思考时间来描述这段时间。

一般思考时间是在一定时间范围内随机变化的，例如 1 至 3 秒之间随机变化。

在 LoadRunner 中主要通过 lr_think_time 函数+RTS 模拟随机思考时间，

在 jmeter 中通过高斯随机定时器来模拟随机思考时间。

思考时间的应用场景一般是构造测试数据接口、混合压测场景、稳定性压测场景

124. 工作中常用的 jmeter 自带函数有哪些

1. digest 特定哈希算法的加密函数
2. urlencode、urldecode 分别是 URL 编码、解码函数
3. time 获取当前各种格式时间的函数
4. Random 获取指定范围数值的随机数函数
5. StringToFile 指定字符串写入文件的函数
6. UUID 函数返回一个伪随机类型 4 通用唯一标识符

125. LoadRunner 中 unique 参数化是怎么实现的

参考答案：

1. unique 参数化主要原理：根据并发用户个数，将参数池某个连续的范围分段，每个用户使用其中的一段数据,

2. unique 规定了第一段的起始值和每一段的 size，从而使每个用户在各自的数据段的取值，而不会产生重复的值

126. 出现内存泄露的根本原因是什么？你是怎么定位内存泄露原因的？

1. 内存泄露的根本原因是 Jvm 中老年代中存在着大量存活的对象，这些对象不能被 GC

回收掉，从而占满了整个老年代，造成 Jvm 一直处于 FGC 的状态，程序没有响应，服务器报 OOM 错误

2. 内存泄露主要通过分析老年代中占用空间最大的类都有哪些，然后去代码中找对应的类的创建。通常可以使用 jdk 提供的 jvisualvm 和 jmap 进行堆内存的分析

127. Nginx 常用负载均衡配置有哪些

参考答案:

1. nginx.conf
2. 轮询模式，默认；
3. weight 权重模式，权重越高被访问概率越大，weight=70；
4. ip_hash，在 upstream 中采用 ip_hash 指令,如果客户已经访问了某个服务器，当用户再次访问时，会将该请求通过哈希算法，自动定位到该服务器。

每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器，可以解决 session 的问题。此种策略，可以实现同一个用户访问同一台服务器，会话不会丢失，但是可能会分配不均

128. 怎么根据线下环境评估线上环境的性能

参考答案:

1. 首先线下必须要有专门的性能测试环境
2. 线下环境单台机器配置和线上不能相差很大，可以通过单台的机器性能推算出多台机器性能
3. 如果线下机器配置很差，只能测试出程序有无性能问题，这样线下测试出来的数据对线上没有太大参考意义
4. 如果想获取比较准确的线上性能情况，建议最好做线上的性能测试

129. 什么是幂等性

简单的说，幂等测试就是验证数据一致性和事务完整性。

可能出现幂等问题的场景：

用户重复提交——非常容易发生，前端、后端均需要控制；

网络重发

消息重发

系统间重试

所以说保证接口的幂等性是非常重要的。

测试幂等的手段：前端幂等测试，注意按钮的多次快速点击；

后端接口的幂等测试，使用 postman 或 jmeter 多次发送同一参数的请求，查看服务端响应。

130. 工作中用过哪些 git 命令

参考答案：
git clone 克隆指定仓库
git add 将文件添加到暂存区
git commit 将文件提交到服务器
git push 将文件提交到远程仓库
git status 查看当前仓库状态
git diff 比较版本的区别
git log 查看 git 操作日志
git reset 回溯历史版本
git checkout 切换到指定分支

131. 什么是正则的贪婪匹配？

参考答案：
贪婪匹配:正则表达式一般趋向于最大长度匹配，也就是所谓的贪婪匹配；
非贪婪匹配：就是匹配到结果就好，最少的匹配字符；
默认是贪婪模式；
在量词后面直接加上一个问号？就是非贪婪模式

132. fiddler 的工作原理是什么？在项目测试过程中主要在哪些场景下使用？

工作原理：fiddler 主要是在客户端和服务端之间起到了一个代理的作用，启动 fiddler 后，所有的请求和响应都会经过 fiddler。

应用场景：

（1）判断前后端 bug

（2）抓包进行数据分析

（3）请求断点：主要是测试服务端对客户端提交的非法数据是否进行校验，比如购票数量为 0，无效的活动时间等

（4）响应断点：主要测试服务端返回异常数据后，客户端的处理方式，比如服务端返回 500，或者某些其他异常业务数据

（5）弱网测试，可以修改 fiddler 中内置的网络延迟时间，默认是上传 300ms 和下载 150ms，这两个值越大，网络速率越低

133. 接口测试用例的编写要点有哪些？

参考答案:
1）必填字段：请求参数必填项、可选项

2）合法性：输入输出合法、非法参数
3）边界：请求参数边界值等
4）容错能力：大容量数据、频繁请求、重复请求（如：订单）、异常网络等的处理
5）响应数据校验：断言、数据提取传递到下一级接口...
6）逻辑校验：如两个请求的接口有严格的先后顺序，需要测试调转顺序的情况
7）性能：对接口模拟并发测试，逐步加压，分析瓶颈点
8）安全性：构造恶意的字符请求，如：SQL 注入、XSS、敏感信息、业务逻辑（如：跳过某些关键步骤；未经验证操纵敏感数据）

134. 在浏览器中输入了一个 url 后，请求流程是什么样的

参考答案：
1、DNS 域名解析
2、与服务器建立 TCP 连接
3、发起 HTTP 请求，发送数据
4、服务器响应 HTTP 请求，返回数据
5、浏览器解析数据、渲染
6、关闭连接

135. 说几个常见的 HTTP 请求头字段吧

参考答案：

字段名	解释
Host	目标域名或ip
Content-Length	请求数据长度
Accept	客户端希望接受的数据类型，*/*代表所有类型
User-Agent	客户端使用什么工具去访问（浏览器）
Content-Type	请求body中数据的类型，常用的有： x-www-form-urlencoded代表表单页面 application/json代表是json字符串
Cookie	请求中携带的cookie信息

136. 为什么要做接口测试

参考答案：
A、在公司里，客户端和服务端通常是由不同的团队开发的，在项目开发过程中，客户端和服务端开发的进度不一致，比如服务端先开发完了，这个时候可以先对服务端进行

接口测试，确保服务端逻辑和返回数据是正确的，然后再测试客户端。另外某些测试部门，专门测试服务端开发团队，因此，他们的测试对象就是接口。

B、在测试某些业务时，不能仅仅通过前端来测试，比如用户注册，前端限制了用户名不能为空，但是有些人可能通过工具绕过前端直接调用服务端接口，如果服务端没有做相关的逻辑判断，就会造成数据错误。包括接口数据传输过程中是否对关键信息加密等。所以必须针对服务端接口单独做测试。

C、 在开发提测后，可以先通过工具把服务端的接口测试跑一遍，确保接口测试用例都是通过的，快速判断服务端接口是否符合预期。然后再通过 UI 界面进行测试。否则接口有 bug，前端页面必定有 bug。

137. 接口测试和 web 页面测试有什么区别？

参考答案：

1) Web 页面测试是通过界面操作来进行测试的，输入不同的数据来测试不同的场景。
2) 接口测试是使用工具直接像服务器发送 HTTP 请求去测试，输入不同的参数来测试不同的场景。
3) 通常 web 页面会限制某些输入数据，比如必填项、数据的格式等。而接口测试是可以输入任何数据的，可以测试更多的异常数据场景。
4) Web 测试需要考虑浏览器的兼容，接口测试不需要
5) Web 测试需要将前端，服务端全部开发好后 才可以进行测试，接口测试只要服务端开发完成，就可以开始测试

138. pytest 里如何进行 case 的组装

参考答案：

1) 默认使用检查以 test_ .py 或**test.py 命名的文件名，在文件内部查找以 test 打头的方法或函数，并执 行
2) 可以使用自定义 marker（装饰器），比如 pytest 运行的时就只运行带有该 marker 的测试用例
3) 在 pytest.ini 配置文件中可以针对 pytest 执行时的 case 做处理，比如要执行某个目录下的用例，执行某个脚本文件等等

139. 应用服务器 cpu 高和数据库服务器 cpu 高的分析思路是什么？

参考答案：

1) 应用服务器的 cpu 高，先要看 tps 和响应时间，如果 tps 比较高，我们认为是正常的 cpu 消耗；如果 tps 比较低，那么往往某些代码过于消耗 cpu，可以考虑使用代码剖析工具分析下
2) 数据库服务器 cpu 高，往往是因为 sql 语句执行效率比较低，可以通过对数据库慢查询是监控，结合执行计划进行分析，是否是相关表没有索引或索引未生效

140.Fiddler 手机 app 抓包设置

参考答案：

（1）手机与电脑连接同一网段（比如同一个 WIFI）

（2）设置代理服务器与网络连接起来

（3）在手机上设置服务器与端口（ipconfig 查看电脑 IP 或通过 Fiddler 的 Online 查看 IP）

141.什么 csrf 攻击原理？如何解决？

参考答案:

（1）csrf 攻击原理

简单来说就是: 你访问了信任网站 A，然后 A 会用保存你的个人信息并返回给你的浏览器一个 cookie，然后呢，在 cookie 的过期时间之内，你去访问了恶意网站 B，它给你返回一些恶意请求代码，要求你去访问网站 A，而你的浏览器在收到这个恶意请求之后，在你不知情的情况下，会带上保存在本地浏览器的 cookie 信息去访问网站 A，然后网站 A 误以为是用户本身的操作，导致来自恶意网站 C 的攻击代码会被执行，比如发邮件，发消息，修改你的密码，购物，转账，偷窥你的个人信息，导致私人信息泄漏和账户财产安全受到威胁。

（2）常见解决方案

Cookies Hashing：每一个表单请求中都加入随机的 Cookie，由于网站中存在 XSS 漏洞而被偷窃的危险。

HTTP refer：可以对服务器获得的请求来路进行欺骗以使得他们看起来合法，这种方法不能够有效防止攻击

142.工作中常用的 JMeter 参数化哪些

参考答案:

1、函数，例如 digest、intSum、longSum、counter、Random、RandomString、time

2、csv 配置元件

3、测试计划、用户定义的变量

4、beanshell 内置变量 vars、props

5、关联：后置处理器、例如 正则提取、json 提取

143.JMeter 参数化在性能测试中有哪些适用场景

参考答案:

1，造数据

2，用户差异性随机行为模拟

3，禁止重复数据情况

4、sign 签名校验字段
5、身份校验（后面讲 cookie、session、token）

144.如何使用 JMeter 测试 HTTP/HTTPS 接口？

参考答案
（1）在 Jmeter 里添加线程组、Http 请求、头文件管理器、查看结果树、聚合报告。
（2）在线程组里配置线程数和运行时间。
（3）在 Http 请求里配置协议、IP、端口号、请求方式、URL、参数(化)、关联、断言。
（4）在查看结果树里查看接口的请求数据、响应数据。
（5）在聚合报告中查看接口响应时间。

145.如何使用 Jmeter 做 HTTP 性能测试

参考答案：
（1）压测脚本准备：基本信息、参数化、关联、断言、聚合报告
（2）压测执行：依据需求确定压测场景，单交易并发测试、混合交易负载测试、稳定性测试，线程组下设置并发数、压测时长或者迭代次数
（3）压测结果查看对比：聚合报告中查看响应时间、错误率、吞吐量等指标来分析性能

146.有遇到过接口有错误的情况吗？

参考答案：
1）业务错误
返回信息错误。
例如：备注字段值为边界值字符串情况下，接口返回异常 Exception。
例如 2：没有走折扣情况下，订单金额和支付金额不一致情况下，仍能支付，接口返回支付成功。
2）异常错误
接口传入异常参数。
例如：充钱填了负数，反而能充钱成功。
例如 2：手机号接口参数值传英文字母字符串没有处理，接口报 null 错。

四.代码

147.使用 python 语法实现 I love China 输出 China love I

参考答案：

```
def resverse(str_list, start, end):
    while start < end:
        str_list[start], str_list[end] = str_list[end], str_list[start]
        start = start + 1
        end = end - 1


setence = 'I love China'
str_list = list(setence)
i = 0
while i < len(str_list):
    if str_list[i] != ' ':
        start = i
        end = start + 1
        while (end < len(str_list)) and (str_list[end] != ' '):
            end += 1
        resverse(str_list, start, end - 1)
        i = end
    else:
        i += 1
str_list.reverse()
print(' '.join(str_list))
```

148. 现在有一个列表 a = [1, 3, 12, 7, 3, 1, 5, 8, 12, 5, 21, 44]去除其中的重复项。

参考答案：

```
a = [1, 3, 12, 7, 3, 1, 5, 8, 12, 5, 21, 44]
l = list(set(a))
print(l)
```

149. python 中为什么使用* args，** kwargs？

如果我们不确定要往函数中传入多少个参数，或者我们想往函数中以列表和元组的形式传参数时，那就使要用*args

如果我们不知道要往函数中传入多少个关键词参数，或者想传入字典的值作为关键词参数时，那就要使用**kwargs

150. 请说一下单例模式的概念及应用场景

参考答案:

单例模式（Singleton），是一种常用的软件设计模式，单例对象的类必须保证只有一个实例存在。

网站的计数器，一般采用单例模式，否则难以实现同步；

多线程的线程池设计一般也是单例模式，方便对池中的线程进行控制；

操作系统的文件系统，因为一个操作系统只能有一个文件系统；

web 应用的配置对象的读取，一般也是单例模式，这是由于配置文件是共享的资源；

Windows 的 Task Manager（任务管理器）就是很典型的单例模式；

数据库连接池的设计一般也是采用单例模式，因为数据库连接是一种数据库资源。数据库软件系统中使用数据库连接池，主要是节省打开或者关闭数据库连接所引起的效率损耗，这种效率上的损耗还是非常昂贵的，因为何用单例模式来维护，就可以大大降低这种损耗

151. 请用代码实现冒泡排序，语言不限

参考答案：

Java:

```java
public static void bubbleSort(int[] array) {
    // 进行多少次排序动作
    for (int m = 0; m < array.length - 1; m++) {
        // 每次排序比较的次数
        for (int n = 0; n < array.length - 1 - m; n++) {
            int tmp;
            if (array[n] > array[n + 1]) {
                tmp = array[n];
                array[n] = array[n + 1];
                array[n + 1] = tmp;
            }
        }
    }
}
```

Python:

```python
def bubble_sort(array):
    for j in range(len(array) - 1, 0, -1):
        for i in range(j):
            if array[i] > array[i + 1]:
                array[i], array[i + 1] = array[i + 1], array[i]
```

152. Java 中实现多线程都有哪些方式？

参考答案：

三种方式：

1. 继承 Thread 类，重写 run 方法
2. 实现 Runnable 接口，实现 run 方法
3. 实现 callable 接口，实现 call 方法

实现 Runnable 接口比继承 Thread 类所具有的优势：

1. 适合多个相同的程序代码的线程去处理同一个资源
2. 可以避免 java 中的单继承的限制
3. 线程池只能放入实现 Runable 或 callable 类线程，不能直接放入继承 Thread 的类

callable 和 runnable 的区别：

1. Callable 规定的方法是 call()，而 Runnable 规定的方法是 run().
2. Callable 的任务执行后可返回值，而 Runnable 的任务是不能返回值的。
3. **call()**方法可抛出异常，而 **run()**方法是不能抛出异常的。
4. 运行 Callable 任务可拿到一个 Future 对象， Future 表示异步计算的结果。 它提供了检查计算是否完成的方法，以等待计算的完成，并检索计算的结果。 通过 Future 对象可了解任务执行情况，可取消任务的执行，还可获取任务执行的结果。
5. Callable 是类似于 Runnable 的接口，实现 Callable 接口的类和实现 Runnable 的类都是可被其它线程执行的任务。
6. Callable 执行的任务可以取消、判断是否取消、获取是否执行结束、获取线程返回结果等方法，Runnbable 均不具备这些功能

153. 在 Java 中，重写和重载的区别

Java 的方法重载，就是在类中可以创建多个方法，它们具有相同的名字，但具有不同的参数和不同的定义。调用方法时通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法。

在 Java 继承关系中，若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则子类方法将覆盖父类方法，这叫做重写

154. Java 中的 List 和 Set 有什么区别

参考答案：

两个接口都是继承自 Collection，是常用来存放数据的集合，主要区别如下：

1. 在 List 中允许插入重复的元素，而在 Set 中不允许重复元素存在。
2. List 是有序集合，会保留元素插入时的顺序，Set 是无序集合。
3. List 可以通过下标来访问，而 Set 不能。

155. Java 中的 String、StringBuilder、StringBuffer 有什么区别

参考答案：

String 是字符串常量

StringBuffer 是字符串变量
StringBuilder 是字符串变量
执行速度：StringBuilder > StringBuffer > String
StringBuilder 是非线程安全的，适合单线程处理字符串
StringBuffer 是线程安全的，适合多线程处理字符串

156. 用代码实现裴波那契数列，语言不限

Java:

```
public static long fibonacciTest(long n) {
    if (n == 0 || n == 1){
        return n;
    }
    return fibonacciTest(n-1) + fibonacciTest(n-2);
}
```

Python:

```
def fib_test(n):
    if n == 0 or n == 1:
        return n
    return fib_test(n - 1) + fib_test(n - 2)
```

157. Python 中的列表和元组的区别是什么

参考答案：

1. 列表可以看成是动态数组,它们是可变的并且可以重新增加、修改元素
2. 元组可以看成是静态的数组,它们是不可变的,并且长度也是一旦创建就无法改变

158. 用代码实现单例模式

Java

```

public class SingletonTest {

    // 1、构造方法私有化
    private SingletonTest() {}

    // 2、创建私有静态内部类
    private static class SingletonHolder {
        // 3、创建静态私有 final 类型的实例对象
```

```
        private static final SingletonTest singleton2 = new
SingletonTest();
    }

    // 4、创建公有静态获取实例的方法
    public static SingletonTest getInstance() {
        return SingletonHolder.singleton2;
    }
}
```

Python

```
def singleton(cls):
    _instance = {}

    def inner():
        if cls not in _instance:
            _instance[cls] = cls()
        return _instance[cls]
    return inner

@singleton
class Cls(object):
    def __init__(self):
        pass
```

159.用代码实现选择排序算法

参考答案：

Java

```
public static String selectionSort(int[] arr) {
    for(int i=0; i<arr.length; i++) {
        //最小数的索引
        int minIndex = i;
        for(int j=i; j<arr.length; j++) {
            //找到最小数，并记录最小数的索引
            if(arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        //交换符合条件的数
        int tmp = arr[i];
```

```
                arr[i] = arr[minIndex];
                arr[minIndex] = tmp;
            }
        }
        return Arrays.toString(arr);
    }
```

Python

```
def sort(arr):
    for j in range(len(arr)-1):
        minIndex = j
        for i in range(j+1,len(arr),1):
            if(arr[i] < arr[minIndex]):
                minIndex = i
        arr[j],arr[minIndex] = arr[minIndex],arr[j]
```

160. 使用代码判断一个数字是否是回文数

参考答案

设 n 是一任意自然数，如果 n 的各位数字反向排列所得自然数与 n 相等，则 n 被称为回文数

参考答案：

Java

```
public boolean isPalindrome(int x) {
    String str = String.valueOf(x);
    final StringBuilder builder = new StringBuilder(str);
    if(builder.reverse().toString().equals(str)){
        return true;
    }
    return false;
}
```

Python

```
def is_palindrome(num):
    n = list(str(num))
    tmp = int("".join(n[::-1]))
    return num == tmp
```

161. 接口和抽象类的区别

参考答案

抽象类

其中包含抽象方法的类，但是也可以包含成员变量和成员方法

1. 抽象方法必须为 public 或者 protected（因为如果为 private，则不能被子类继

承，子类便无法实现该方法），缺省情况下默认为 public

2．抽象类不能用来创建对象

3．如果一个类继承于一个抽象类，则子类必须实现父类的抽象方法。如果子类没有实现父类的抽象方法，则必须将子类也定义为 abstract 类

接口

接口中可以包含变量和方法，但是变量会隐式的指定为 public static final，方法会被隐式的指定为 public abstract，并且接口中的方法不能任何具体实现。也就是说接口中都是抽象方法。接口是一种极端抽象的类型。

区别

语法层面的区别

1．抽象类中可以包含具体的实现方法，而接口中只能有 public abstract 方法

2．抽象类中的成员变量可以是各种类型的，而接口中只能有 public static final 类型

3．抽象类中可以包含静态代码块和静态方法，而接口中不可以

4．一个类只能继承一个抽象类，但是可以实现多个接口

设计层面的区别

1．抽象层次不同，抽象类对整个类整体进行抽象，包括属性、行为；而接口只是对类的局部行为进行抽象。比如，门和警报的例子：门都有 open 和 close 的动作，可以定义一个抽象类 Door，包含 open 和 close 方法。这样所有的门都可以继承这个抽象类，从而包含了这两个方法。另外报警功能 alarm 是一种延伸的功能，但是不可以加到抽象类 Door 中，因为并不是每个门都有报警的功能。所以需要定义一个接口，接口中包含 alarm 方法，这样需要有报警功能的门来实现这个接口，没有则不需要。

2．跨域不同，抽象类和子类之间体现的是一种继承关系，父类和子类在概念本质上相同的。接口则不然，接口和其实现者不需要有任何关联，接口的实现者仅仅是实现了接口的规定的规范。

3．设计层次不同，抽象类是自下而上的设计，先有多个子类，然后抽取共同特性，形成了抽象类。而接口则不同，接口的设计的时候根本不知道会有什么实现者以什么形式来实现。接口只是一种实现定义好的规范。

162. 什么是多态？多态有什么好处？

参考答案

多态是同一个行为具有多个不同表现形式或形态的能力。在代码层面，多态指的是父类引用指向子类对象。

比如，父类 Animal，子类 Dog，子类 Cat。

```
Animal animal1 = new Dog()
Animal animal2 = new Cat()
```

多态的存在有三个前提

1．要有继承关系

2．子类要重写父类方法

3．父类引用指向子类对象

多态的优点

多态的出现大大提高了程序的扩展性

多态的弊端

父类引用指向子类对象，不能使用子类的特有的成员和方法，如果想访问子类特有成员和方法，必须进行向下转换，把父类转换为子类

163. Java 反射是什么，它有什么应用场景？

参考答案：

反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；

对于任意一个对象，都能够调用它的任意一个方法和属性；

这种动态获取的信息以及动态调用对象的方法的功能称为 java 语言的反射机制。

获取 class 对象的三种方式

1．对象的 getClass 方法：object.getClass()

2．类的 class 属性：Object.class

3．Class 类的 forName 方法 （最常用）

可获取的数据：属性、方法、构造方法、方法注解等

主要应用场景：基础框架开发

164. ArrayList 和 LinkedList 有什么区别？

参考答案

ArrayList：底层数据结构为数组，可以实现自动扩容，特点是查询操作快，增删慢

LinkedList：底层数据结构为双向链表，特点是增删快，查询慢

165. HttpClient 使用方法是什么样的？

参考答案

使用 HttpClient 发送请求、接收响应一般需要如下几步：

1．创建 HttpClient 对象。

2．创建请求方法的实例，并指定请求 URL。如果需要发送 GET 请求，创建 HttpGet 对象；如果需要发送 POST 请求，创建 HttpPost 对象

3．如果需要发送请求参数，对于 HttpPost 对象而言，也可调用 setEntity(HttpEntity entity)方法来设置请求参数。

4．调用 HttpClient 对象的 execute(HttpUriRequest request)发送请求，该方法返回一个 HttpResponse。

5．调用 HttpResponse 的 getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头；调用 HttpResponse 的 getEntity()方法可获取 HttpEntity 对象，该对象包装了服务器的响应内容。程序可通过该对象获取服务器的响应内容。

6．释放连接。无论执行方法是否成功，都必须释放连接。

166.JVM 内存结构了解吗，每个区域都存放什么数据？

参考答案

Java 程序是运行在 JVM 之中的，所有对象的创建和分配都在 JVM 中。

内存结构：

方法区：各线程共享，主要存放类信息、常量、静态变量

虚拟机栈：线程私有，主要存放基本数据类型（int、char、float...）和对象的引用

本地方法栈：线程私有，为虚拟机使用到的 Native 方法服务，如 Java 使用 c 或者 c++ 编写的接口服务时，代码在此区运行

堆：线程共享，主要存放对象的实例和数组

程序计数器：它的作用可以看做是当前线程所执行的字节码的行号指示器，记录线程上次执行到程序的哪个位置

167.数据结构-双向链表 代码实现

示意图

双向链表的数据结构：

参考答案

java 代码

```
public class MyDoubleLink {
    public int linkSize = 0;
    public Node head = null;// 头结点
    public Node tail = null;// 尾结点

    // 使用内部类定义节点
    public class Node {
        public Object data;
        public Node prev = null;
        public Node next = null;
        public Node(Object data) {
            this.data = data;
        }
    }

    /**
     * 添加节点至尾部
     * @param data
     */
    public void addLast(Object data) {
        linkSize++;
        Node newNode = new Node(data);
        if (head == null) {
            // 第一个元素
            head = newNode;
            return;
        }
        if (tail == null) {
            // 第二个元素
            tail = newNode;
            tail.prev = head;
            head.next = tail;
            return;
        }
        // 将新节点的指针和 tail 节点建立关系
        tail.next = newNode;
        newNode.prev = tail;
        tail = newNode;
    }

    /**
     * 删除指定位置的节点
     * 遍历到指定节点，将指定位置的上一个节点与下一个节点指针建立关系
```

```
     * @param index
     * @return
     */
    public boolean delete(int index) {
        if (index < 0 || index >= linkSize) {
            return false;
        }
        linkSize--;

        // 删除的是头结点
        if (index == 0) {
            // 只有一个头结点
            if (head.next == null) {
                head = null;
                return true;
            }
            head = head.next;// 删除头结点
            head.prev = null;
            return true;
        }

        // 删除的是尾节点
        if (index == (linkSize - 1)) {
            tail = tail.prev;
            tail.next = null;
            return true;
        }
        Node tmpNode = head;
        // 遍历链表，获取被删除的节点
        for (int i = 0; i < index; i++) {
            tmpNode = tmpNode.next;
        }
        // tmpNode 是被删除节点，将其上下两个节点之间建立关系
        tmpNode.prev.next = tmpNode.next;
        tmpNode.next.prev = tmpNode.prev;
        return true;
    }

    /**
     * 获取指定位置的节点
     * @param index
     * @return
     */
    public Node get(int index) {
```

```
        if (index < 0 || index >= linkSize) {
            throw new RuntimeException("指针越界");
        }
        Node tmpNode = head;
        // 遍历节点，获取指定位置的节点
        for (int i = 0; i < index; i++) {
            tmpNode = tmpNode.next;
        }
        return tmpNode;
    }

    /**
     * 打印链表上所有节点的数据
     */
    public void print() {
        Node tmpNode = head;
        for (int i = 0; i < linkSize; i++) {
            System.out.println(tmpNode.data);
            tmpNode = tmpNode.next;
        }
    }

}
```

168. 请解释，下面这段代码的输出结果将是什么？

代码如下所示：

```
class Parent(object):
    x=1
class Child1(Parent):
    pass
class Child2(Parent):
    pass
print(Parent.x,Child1.x,Child2.x)
Child1.x=2
print(Parent.x,Child1.x,Child2.x)
Parent.x=3
print(Parent.x,Child1.x,Child2.x)
```

参考答案：

```
1 1 1 #继承自父类的类属性 x，所以都一样，指向同一块内存地址。
1 2 1 #更改 Child1，Child1 的 x 指向了新的内存地址。
3 2 3 #更改 Parent，Parent 的 x 指向了新的内存地址。
```

169. 说出如下代码运行结果以及原因

说出如下代码运行结果以及原因

```
def extendList(val,list=[]):
    list.append(val)
    return
list list1=extendList(10) # list1=[10] 坑点
list2=extendList(123,[]) #list2=[123]
list3=extendList('a') # list1 和 list3 的内存地址是同一个 [10,'a']
print("list1=%s"%list1)
print("list2=%s"%list2)
print("list3=%s"%list3)
```

参考答案：

列表是可变数据类型，只要 list=[]不变，那么内存地址就不会变

```
list1=[10, 'a']
list2=[123]
list3=[10, 'a']
```

170. 现有字典 d={"a":24，"g":52，"i":12，"k":33}请按字典中的 value 值进行排序？

参考答案：

```
sorted(d.items(), key=lambda x: x[1])
中 d.items() 为待排序的对象；
key=lambda x: x[1] 为对前面的对象中的第二维数据（即 value）的值进行排序
```

171. 请用 python 代码写一个单例模式，并简述单例模式的应用场景

参考答案：

```
class Singleton(object):
    def__new__(cls):# 为对象分配内存空间
        if not hasattr(cls,'instance'):  # instance 做一个标记，如果
instance 存在，那么就证明已经生成过对象
cls.instance=super(Singleton,cls).__new__(cls) # 分配内存地址
            return cls.instance

        应用场景：
        1. 任务管理器
        2. 回收站
```

```
3. 网站的计数器
4. 日志应用
5. Web 应用的配置对象
6. 数据库连接池
```

172. 简述什么是装饰器及应用场景，并手写装饰器

参考答案：

定义：装饰器的本质是闭包，主要作用就是在不改变被装饰函数的原有功能的基础上，增加额外的功能。

应用场景：插入日志、性能测试、事务处理等方面。

计算函数执行时间的装饰器如下所示：

```
import time
def timer(func):
   def deco(*args,**kwargs):
       start_time=time.time()
       func(*args,**kwargs)
       stop_time=time.time()
       print("running time is %s "%(stop_time-start_time))
       return deco
   @timer def test1():
       time.sleep(3)
       print("in the test 1")
       test1()
```

173. python 中 match 和 search 的区别？

参考答案：

match()函数只检测是不是在 string 的开始位置匹配，

search()会扫描整个 string 查找匹配；

也就是说 match()只有在 0 位置匹配成功的话才有返回，

如果不是开始位置匹配成功的话，match()就返回 None。

174. 简要说明一下你对生成器和迭代器的理解？

参考答案：

迭代器：

迭代是 Python 最强大的功能之一，是访问集合元素的一种方式。 迭代器是一个可以记住遍历的位置的对象。 迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。 迭代器有两个基本的方法： **iter()** 和 **next()** 。 字符

串，列表或元组对象都可用于创建迭代器：任何实现了 __iter__ 和 __next__() 方法的对象都是迭代器。 __iter__ 返回迭代器自身，__next__返回容器中的下一个值。

生成器：

生成器是一种特殊的迭代器，它的返回值不是通过 return 而是用 yield，生成器算得上是 Python 语言中最吸引人的特性之一，生成器其实是一种特殊的迭代器，不过这种迭代器更加优雅。它不需要再像上面的类一样写 __iter__() 和 __next__() 方法了，只需要一个 yield 关键字。 生成器一定是迭代器，

但迭代器不一定是生成器。生成器是边计算边遍历的。

175. 简述 ORM 及其优缺点？

参考答案：

ORM 框架：

对象关系映射，通过创建一个类，这个类与数据库的表相对应！类的对象代指数据库中的一行数据。

让用户不再写 SQL 语句，而是通过类以及对象的方式，和其内部提供的方法，进行数据库操作！

把用户输入的类或对象转换成 SQL 语句，转换之后通过 pymysql 执行完成数据库的操作。

ORM 的优缺点：

优点：

1.提高开发效率，降低开发成本

2.使开发更加对象化

3.可移植

4.可以很方便地引入数据缓存之类的附加功能

缺点：

1.在处理多表联查、 where 条件复杂之类的查询时， ORM 的语法会变得复杂。就需要写入原生 SQL 。

176. 现在有一个列表 L = [1, 2, 3, 4, 5, 6, 6, 5, 4, 3, 2, 1]，去重列表中的重复项，用多种方式处理。

参考答案

第一种方法：借助字典键值的唯一性

```
L = [1,2,3,4,5,6,6,5,4,3,2,1]
d = {}
new_d = d.fromkeys(L)
obj = new_d.keys()
```

```
new_l = list(obj)
new_l 是去重之后的列表

第二种方法：set()去重
new_l = list(set(L))
new_l 是去重之后的列表

第三种方法：遍历
new_l = []
for x in L:
   if x not in new_1:
       new_1.append(x)
new_l 是去重之后的列表
```

五. 求职相关

177. 你对于加班是怎么看的？

参考答案：

（1） 首先，加班不是目的而是手段，我会尽量在正常的上班时间完成我今天所有的工作。

（2） 重复性工作找到提效方法

（3） 如果确定需要测试加班情况（例如：临时发版），我这边保证积极配合，保质保量完成测试工作任务

（4） 如果其他情况需要加班情况，比如业务、技术培训、盘点会议等等，我这边也非常乐意和大家通过这种形式，进一步提示自己实力、以及同步信息。

178. 你有面试过其他公司吗？

参考答案：

（1）建议大家答第一家，尽量不要回答超过 3 家。

（2）如果说面试过其它的，它就会继续问有没有拿到 offer ，说没拿到 offer 好像能力显得挺 low；

（3）如果说拿到 offer，人家会考虑你那边有 offer 了，说不定我的 offer 没人家的好，那你就不来了，干脆就不给你发了

（4）或者继续询问你拿到 offer 那家公司的相关信息。

179. 为什么离职？

参考答案：

1.不要 diss（吐槽） 上家公司领导和同事。

2.不要说在上家公司学不到东西。

3.尽量不要归咎于个人原因，比如不想再继续异地恋了。

4.尽量陈述正面的客观情况，比如谋求更好的发展空间、公司倒闭大裁员、停发工资。

180. 你期待什么样的公司？

参考答案：

1. 经济--待遇，福利
2. 发展前途--手工/自动化
3. 平台大小--公司规模
4. 项目业务--如互联网金融、电商、智能家居、人工智能、支付、财会
5. 离家远近--建议陈述交通是否便利，建议陈述能够就近租房
6. 工作量大小--任务量，加班量
7. 公司环境--公司文化，同事相处

181. 你离招聘公司有多远

参考答案：

（1）离公司不远

（2）租房马上到期可以搬

182. 期望薪资？

参考答案：

1.期待薪资范围（按照 JD 说）但可谈

2.按照不同城市工资基数说

3.反问 HR 贵公司薪资结构

4.拿到多个 offer 情况下，薪资每次提 500 到 1000 元不等。

183. 你有什么想问的？

参考答案：

1.问公司的流程（测试、非测试方向都行）

2.问自己入职后的工作内容和范围

3.什么时间可以得到面试答复

4.问应聘公司的核心业务

5.问应聘部门团队人员构成情况

6.问应聘团队规划计划
7.平时一般公司加班到几点

184.和其他应聘者相比，你觉得你有什么优势？

答题思路：
1.业务经验和贵公司高度匹配
2.专业技能、综合技能
3.快速融入团队
4. 特长额外加分项：比如钢琴、体育这些

参考答案：

第一，我对咱们公司的业务有了解，我上家公司的业务和你们公司的业务相同（重合度很高），我在这块有业务经验，我对这块业务非常感兴趣。

第二，我的测试技术技能优势，对测试的技能非常熟练，对（比如银行、CRM、电商、物流、ERP）业务熟悉，除此之外还擅长接口、性能测试。

第三，我这个人能特别快地融入工作环境，我真的来公司上班，我可以很快地上手工作。

185.你未来的发展规划？

参考答案：
1.业务扩展，针对面试公司和自己以往工作差异性业务的补充。
2.技能提升，自动化、性能、安全、测开方向。