1.你怎么保证你写的测试用例是完美的？举例？

自我检测+测试用例评审

1.写好的测试用例要多读几遍，确定测试用例流程；

2.不能有测试点遗漏

3.不能有重复的测试用例

4.不能有无效的测试用例

测试用例例审——评审报告——修改——继续评审

除了评审还有什么方法呢？

需求串讲；

以需求文档为准

需求改动，测试用例随之改变

需求改动，代码改动，测试用例随之改动

怎样写出完美的测试用例，测试用例的设计是一个循序渐进，逐步完善的过程

1.自我检查——不要有遗漏的测试点，不能有重复的，无效的

2.评审——内部评审（我跟我的同事相互评审）

——公开评审（项目组所有的成员：需求人员、项目经理、开发、测试）——评审后，会输出评审报告

————报告记录需要修改的地方，便于下次评审的，直接评审有问题的地方

3.需求串讲

4.以需求文档为主

5.需要改动，开发代码改动，我们的测试用例随之改变

找bug

1.缺陷编号：提交了bug后自动生成

2.缺陷标题：问题产生了，我们描述出来

3.缺陷发现者：我们测试

4.发现缺陷的日期：以当前系统日期为准

5、发现缺陷所属模块：

6.发现缺陷版本:美萍酒店管理系统V1.0\_第一轮

7.指派给谁处理：指给开发人员，但是如果不知道是哪个开发，指派给开发的负责人：项目经理，项目经理会分配给对应的开发人员。

8.缺陷的等级

致命：一般不建议提致命的bug。如果是致命的修改为严重，立即修复

严重：影响主，本版本修复

一般：上线前修复

轻微（提示）：运行上线产品中存在

9.缺陷的优先级

哪些是优先修复的

哪些是次要修复的

10.缺陷的状态

待分流

待处理

待验证

11.缺陷描述

标题

步骤

实际结果

期望结果

截图

缺陷的处理流程

1.由测试人员发现问题，提交到禅道

2.项目经理，分类——对应的人员（如果知道对于的开发人员是谁，就直接选择对应的开发人员）

3.开发人员进行修复问题

4.我们进行回归测试，没有问题——关闭缺陷

有问题——驳回，开发人员继续修改

偶发性问题

在公司里面检查会遇到你的电脑发现了bug，去开发哪里操作就没有问题

然后避免这种问题？

——发现bug一定要重现2次，再提bug。

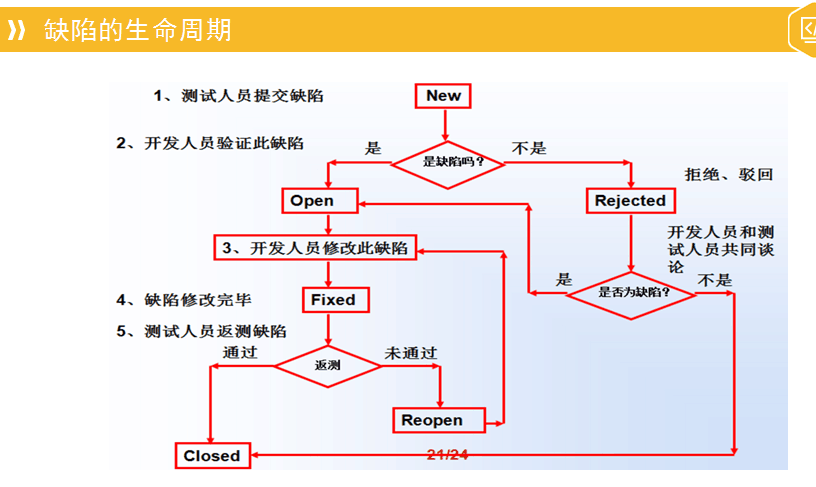
——发现问题。一定要截图——有图有真相

——发一个问题，第二次操作没有了，然后又没有截图

记录下来：你刚刚输入的数据：123，哪个模块，哪个功能，待下次发现的时候，告诉开发

如果发现的问题时严重的问题，就不需要等一下，直接告诉开发，让他留意一下

缺陷的生命周期



1. 测试人员提交缺陷->开发人员验证此缺陷（开发和测试人员共同讨论是否为缺陷）->开发人员修改此缺陷-缺陷修改完毕->测试人员返测缺陷->通过->关闭

日清日结

缺陷定义

1. 软件没有实现产品说明书要求功能
2. 软件出现了产品说明书指明不应该出现的错误
3. 软件实现了说明书未提及的还能
4. 软件未实现产品说明书未明确，但应该实现的功能
5. 软件难以理解。不易使用、运行缓慢等问题
6. 缺陷是系统在开发或者维护过程中就存在的问题
7. 缺陷是系统某种功能失效了