**软件测试实验-第二周**

实验日期：2025 年 9 月 16 日

本次实验主题：大模型辅助下的软件测试

相关资料下载链接：https://github.com/SYSUSELab/Software-Test-Course/tree/main/Lab1

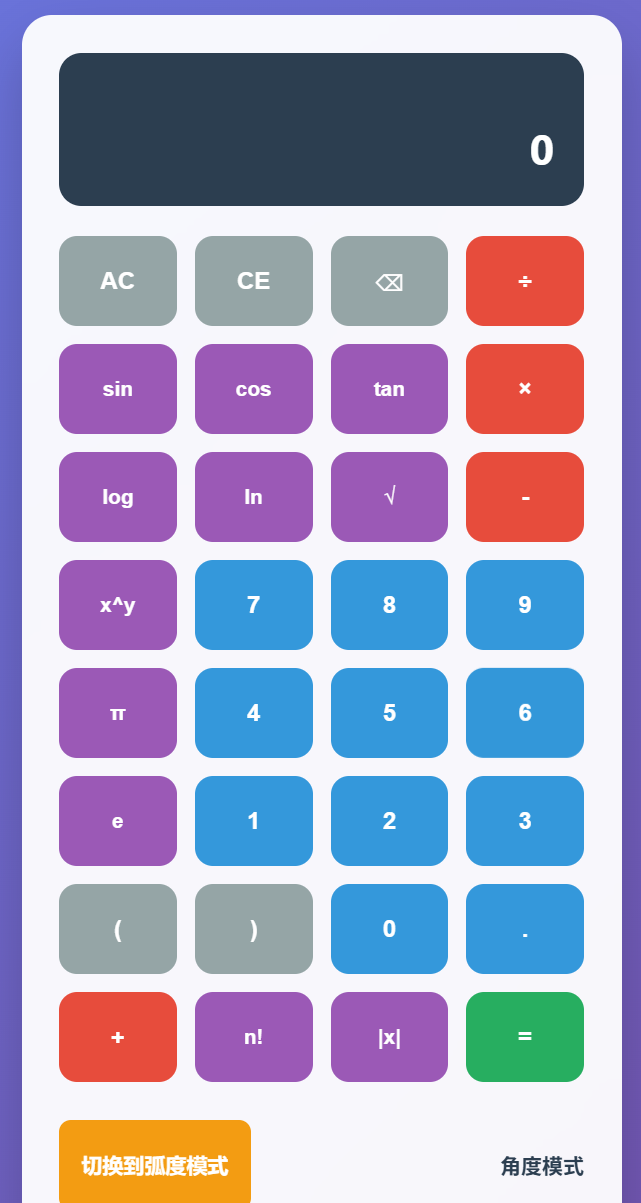
**实验内容说明**

本次实验材料与第一次实验相同，但这次需要大家通过设计大模型的提示词，与大模型合作寻找bug。

实验的资料中包含一个计算器网页程序 calculator-project，该程序包含以下文件：

* index.html # 计算器主页面
* style.css # 元素样式
* script.js # JavaScript 实现的计算器逻辑

使用浏览器打开 index.html, 可以看到以下内容：



这个程序支持**点击按钮**或**键盘输入**进行科学数学公式运算，实验过程中应对计算器功能自由探索，检查程序中可能存在的问题，例如：

* 功能错误：运算结果与数学预期不一致；
* 输入问题：非法输入未被正确处理；
* 交互问题：按钮点击无效或显示异常；
* 界面问题：屏幕显示混乱，或提示信息不清楚。

**大模型提示词示例：**

**角色与目标 (Role and Goal):** 你是一位资深的软件质量保证（QA）工程师，专长于前端 Web 应用的白盒与黑盒测试。你的任务是为一款基于 HTML, CSS, 和 JavaScript 实现的网页版科学计算器，设计一份全面且深入的测试计划。目标是尽可能多地找出潜在的 Bug，包括功能错误、输入问题、交互问题和界面问题。

**核心测试要求 (Core Testing Requirements):** 请根据我提供的图片，生成一份详细的测试用例列表。测试用例需要覆盖以下四个主要方面，并请尽可能考虑各种边缘情况和组合操作。对于每一个测试用例，请遵循以下格式：

* **测试类别:** (例如: 功能-三角函数)
* **测试用例描述:** (简要说明测试的目标)
* **复现步骤:** (清晰、可操作的步骤)
* **预期结果:** (在没有 Bug 的情况下，程序应有的正确表现)

**测试计划生成指令 (Test Plan Generation Instruction):**

**一、功能性与正确性测试 (Functional & Correctness Testing)**

1. **基础算术:**

* 测试加、减、乘、除的整数和浮点数运算。
* 测试运算优先级（PEMDAS/BODMAS），例如 3 + 5 \* 2 预期应为 13 而非 16。
* 测试连续运算，例如 1+2-3\*4/5。
* 测试包含负数的运算，例如 -5 \* (-3)。

1. **科学函数:**

* **三角函数 (角度/弧度模式):**
* 在 **角度模式** 下，测试 sin(90), cos(180), tan(45)。
* 切换到 **弧度模式**，测试 sin(π/2), cos(π), tan(π/4)。
* 测试特殊角度/弧度值，如 0, 30, 60, 1, 2π 等。
* **对数函数:** 测试 log(10), log(100), ln(e), ln(1)。
* **幂与根函数:** 测试 2^3, 4^0.5, 9^2, √16, √0。
* **阶乘:** 测试 5!, 0!, 1!。
* **绝对值:** 测试 |-10|, |5-8|。

1. **括号与表达式:**

* 测试简单的括号优先级，如 (3 + 5) \* 2。
* 测试复杂的嵌套括号，如 10 - (2 + (5-3)\*2)。

1. **控制功能:**

* **AC:** 输入一长串表达式后，按 AC 是否能完全清空显示区和内部状态。
* **CE:** 在输入 123+456 时，按 CE 是否只清除 456，保留 123+。
* **退格 (⌫):** 测试能否正确删除最后一个字符，无论是数字、小数点还是运算符。

**二、边界值与异常情况测试 (Edge Cases & Exception Handling)**

1. **数学逻辑错误:**

* **除以零:** 执行 10 ÷ 0。
* **无效的函数输入:**
* 计算 √(-4) (负数平方根)。
* 计算 log(0), log(-10), ln(0)。
* 计算 (-2)!, (1.5)! (负数或非整数的阶乘)。

1. **输入异常:**

* **连续运算符:** 输入 5 \* - + 3。
* **开头的运算符:** 直接输入 \* 5 或 + 3。
* **结尾的运算符:** 输入 5 + 然后按 =。
* **重复的小数点:** 在一个数字中输入两次小数点，如 3.14.15。
* **括号不匹配:** 输入 (5+3 或 5+3) 然后按 =。
* **空输入:** 未输入任何内容时直接按 =。

1. **数据范围:**

* **大数溢出:** 计算一个非常大的数，如 999999999 \* 999999999 或 99!。
* **精度问题:** 计算 0.1 + 0.2，检查结果是否为 0.3 或存在浮点数精度问题（如 0.30000000000000004）。

**三、交互与用户界面测试 (Interaction & UI/UX Testing)**

1. **输入一致性:**

* 混合使用 **键盘输入** 和 **鼠标点击** 输入同一个表达式，检查是否会产生冲突或状态错误。
* 测试所有按钮是否都能通过键盘上的对应按键触发（例如数字键、+, -, \*, /, Enter 对应 =）。

1. **显示逻辑:**

* 当输入的表达式或计算结果非常长时，显示区域是**如何处理的**？（例如：换行、滚动、截断、还是字体缩小？）。
* 错误信息是否清晰友好？例如，除以零时是显示 "Error", "Infinity", "NaN" 还是直接崩溃？

1. **模式切换:**

* 在输入了包含三角函数的表达式 **中途** 切换角度/弧度模式，计算结果是否会相应改变？
* 切换模式后，界面上的模式提示文本（"角度模式"）是否立刻更新为"弧度模式"？

1. **按钮状态:**

* 当鼠标悬停（hover）或点击（active）按钮时，是否有视觉反馈（如颜色变化）？
* 是否存在某些情况下按钮无法点击或点击无效的问题？

**四、状态管理测试 (State Management Testing)**

1. **链式计算:**

* 执行 2 + 3 =，显示结果 5。此时直接按 \* 4，程序是计算 5 \* 4 还是报错？（预期应为 20）。

1. **计算后输入:**

* 执行 2 + 3 =，显示结果 5。此时直接按数字键 9，显示区是应该被 9 覆盖，还是拼接成 59？（预期应为覆盖，开始新的计算）。

1. **CE/退格后的状态:**

* 输入 123+456，按 CE 清除 456，此时输入 789，表达式是否变为 123+789？
* 输入 sin(90)，按退格键删除 )，再删除 0，再删除 9，每一步的状态是否都正确？

请开始生成这份详细的测试计划吧！

**实验报告要求**

除实验日期、姓名、学号等基本内容外，本次报告需要呈现你在测试过程中发现的漏洞，每个漏洞的报告形式为：

1. 大模型的提示词和模型回复
2. 漏洞简述：简要说明漏洞的表现形式
3. 复现步骤：设计测试过程的思路，以及触发漏洞的操作步骤
4. 预期结果：正常情况下应该出现的反馈
5. 实际结果：程序的实际表现

其中，大模型的提示词及回复可以通过多种形式展现（截图、分享链接或直接复制文字均可），复现步骤、预期结果和实际结果可以添加图片辅助说明。

**注意：**

1. 漏洞报告部分不限页数，其余内容不超过半页。
2. 如果本次实验过程中没有找到漏洞，可以在报告中呈现你的测试思路和执行过的操作。

**大模型参考资料**

prompt技巧：https://zhuanlan.zhihu.com/p/24256272871

prompt设计：https://mp.weixin.qq.com/s?\_\_biz=MzIzOTU0NTQ0MA==&mid=2247540408&idx=1&sn=3181257fb4f8fc393b50c1a1943c6ded&poc\_token=HOa1xmijYd9NeusGOubHslHFpQOmM5lQykfOFGyO

**可用大模型（也可自行选择）**

Qwen：https://chat.qwen.ai/

Kimi（K2）：https://www.kimi.com/

Doubao：https://www.doubao.com/chat/

Deepseek：https://chat.deepseek.com/

**关于大模型辅助软件测试的小Tips**

相对于人工来说，大模型一个显而易见的优势其实是“知识储备”，能够让一个普通人在短时间内快速获得某个领域的大量知识和较为专业的建议，比如在软件测试当中，一个新手就可以迅速的借用大模型完成多种主流的测试方法，甚至可以达到普通程序员的水平，甚至由于其庞大的知识储备，可以在很早期就发现更多可能存在的bug。

另一个又是就是快速，在一些软件测试方法当中，比如说单元测试，需要大量编写重复性的代码进行重复性的代码覆盖测试，而这种测试相对来说比较简单，因此大模型能够更好的胜任。

但是在本次实验当中，不涉及具体代码，所以更重要的是如何更好的利用大模型的“知识储备”，以下有几点可能的建议：

1. 在prompt开头明确赋予LLM一个专家角色，比如软件测试专家，并设计具体的目标，比如找出尽可能多的BUG。
2. 提供详细的上下文，除了像我示例中一样通过发送截图让大模型理解任务外，可以通过发送网页源代码的方式让模型更详细的了解任务（这一点不做强制要求）。
3. 给出具体的指令，比如不要说请找出所有bug，而是让LLM专注于某个具体的领域，比如UI bug、边界值bug等。
4. 不断追问获取更深更多的回复，比如“还有没有除法方面的可能的bug？”。

除此之外，大家还可以在实验过程中思考以下几个问题：

1. 如何能更加高效地指引大模型做测试，发现Bug
2. 大模型有什么不足
3. 你如何验证大模型这位“专家”的建议？
4. 大模型是如何影响你的测试思维的？有没有一个例子让你感到“茅塞顿开”，觉得“原来还可以这样测试”？

**作业提交**

文件格式：PDF 文档

作业提交链接： https://send2me.cn/5zeHG\_UN/QDqiVuvpm0s91A



本次提交截止时间：下课前 (2025年9月16日 9:40/11:50)

逾期提交请联系助教