个小 bug。他在 C 的分支上并创了一个新分支 F2_fix，对代码做了修改来消除这几个小 bug，进而推送至服务器 S；

(7) 之后，C 自己也发现了这几个小 bug。他在本地对代码做了相应的修改和提交。在推送至服务器 S 之前，他先是获得 S 上的最新更新，发现了 D 的修改工作和新分支。他试图将 D 的分支合并到自己的分支 F2，但提示有冲突，后通过手工方式消除了该冲突，成功合并，并将所有本地更新推送至 S；

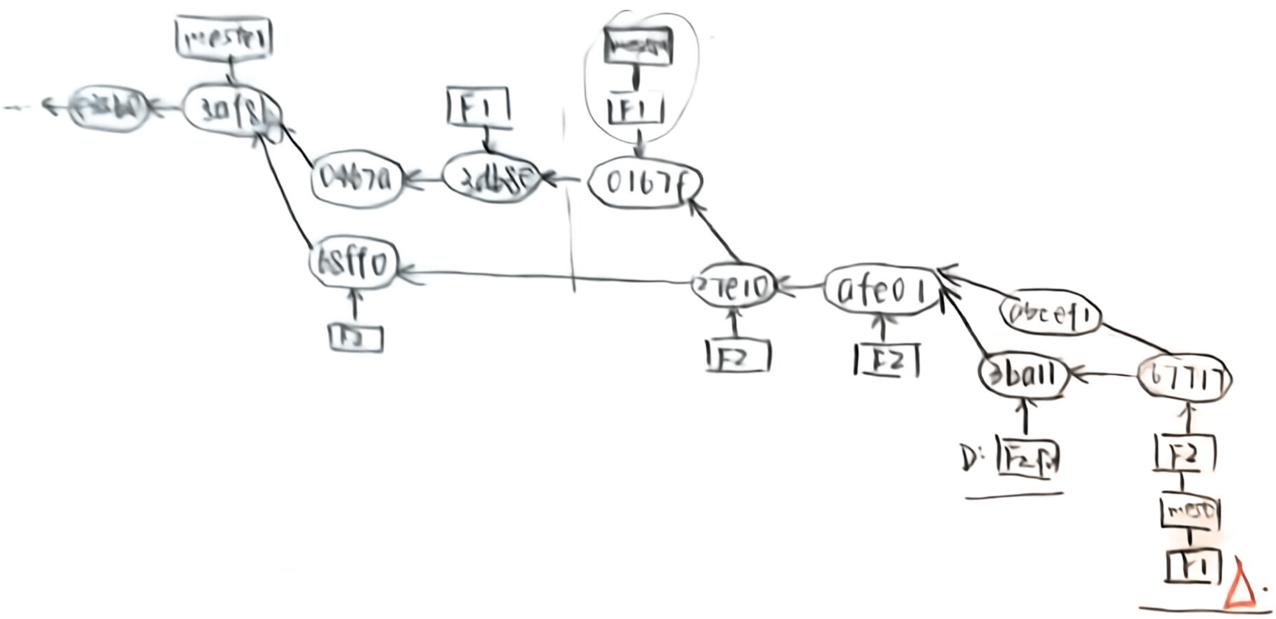
(8) 最后，D 通过 Email 向 A、B、C 分别报告了测试结果；

(9) A 将 B 的分支 F1 合并至主分支并推送至 S 以及 GitHub；

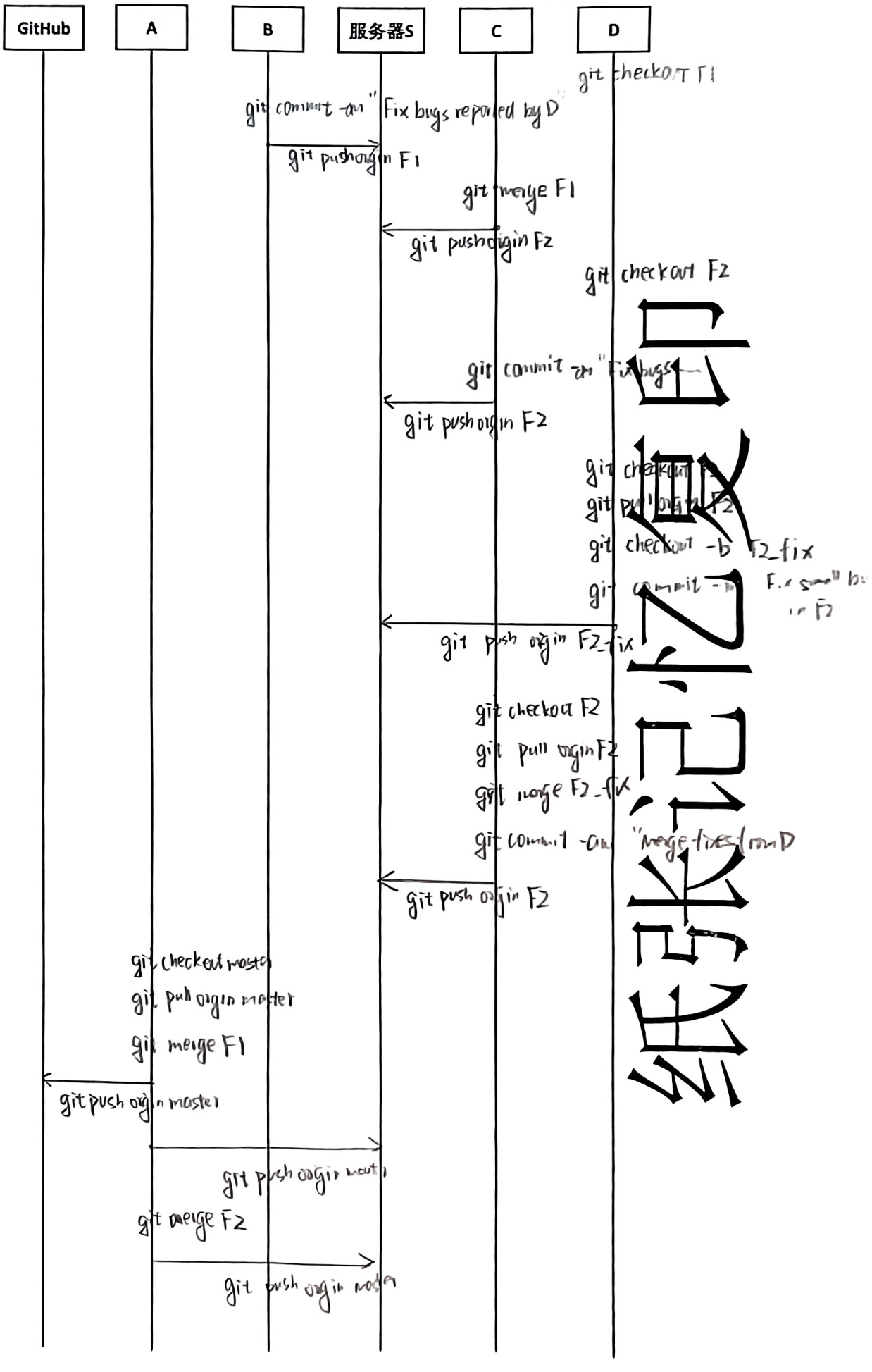
(10) A 将 C 的分支 F2 合并至主分支。由于 F2 特性目前尚不完善，不能对外开源，故仅推送至 S。

题目：

(1) （12 分）在上图基础上，补充出上述所有步骤结束之后服务器 S 上的完整版本变迁历史。请自行为各 commit 的 SHA 指定六位长度的十六进制编号，并确保其唯一性。图中应标明所有分支的位置。



(2)（13 分）在以下时序图上，补充完整上述 1-10 步骤中使用的 Git 指令。



密 封 线

院系 _____ 学号 _____ 姓名 _____ 授课教师 _____

二 软件测试（25 分）

某函数的代码如下所示：

```
    /**
     * 功能描述:              找出指定字符串在目标字符串中的位置
     * @param source          目标字符串
     * @param pattern         指定字符串
     * @return                指定字符串在目标字符串中的起始位置，找不到则输出-1
     */
    public static int match (String source, String pattern) {
1       int index = -1;
2       boolean match = true;
3       for (int i = 0, len = source.length() - pattern.length(); i < len; i++) {
4           match = true;
5           for (int j = 0; j < pattern.length(); j++) {
6               if (source.charAt(i + j) != pattern.charAt(j)) {
7                   match = false;
8                   break;
                }
            }
9           if (match) {
10              index = i;
11              break;
            }
        }
12      return index;
    }
```

已有的测试用例如下：

source	pattern	期望的输出
abc	d	-1

问题：

(1) (5 分) 以上测试用例对上述代码可否满足判定覆盖？如果不能，请补充最少数量的测试用例，使之满足判定覆盖的要求，并给出它们可满足判定覆盖的依据。

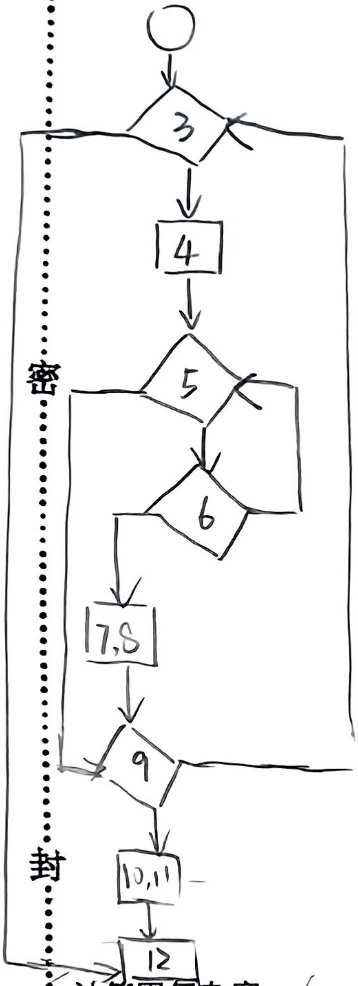
不满足 因为判定覆盖是程序中每个分支都至少通过一次

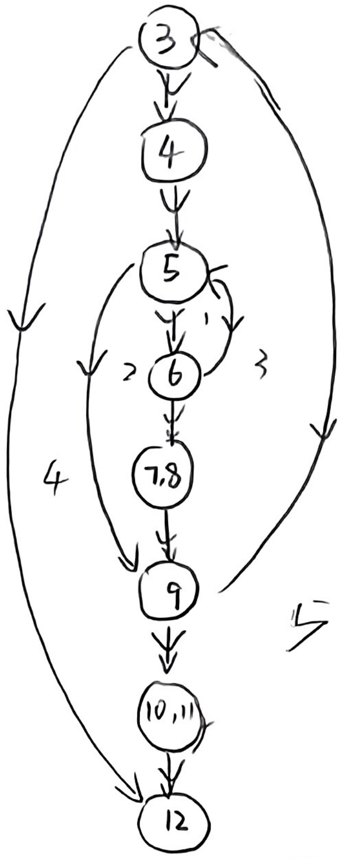
而上述用例第6,9分支没有通入

用例： abc b 1

(2) (15 分) 使用基本路径法，设计一组新的测试用例，对以上表格里的测试用例加以补充，使之可覆盖全部基本路径。增加的测试用例填入右侧表格。另，在完成本题时请使用上页代码中给出的行号，且无需考虑题(1)中增加的测试用例。

绘制控制流图（图中节点编号请使用代码行号）：





至少2条

计算圈复杂度：6

列出基本路径：

编号	包含的代码行的序列（如 1-2-3-4-5）
1	3-12
2	3-4-5-9-10-11-12
3	3-4-5-6-7-8-9-3-12
4	3-4-5-6-5-9-10-11-12

密
封
线

你要开发一个日历应用，帮助用户管理自己的日程安排。这是一个 Web 应用，支持多个用户使用。

每个用户可以建立 1 或多个日历，每个日历可用于管理该用户不同类型的日程安排。例如，张三希望自己的第一个日历记录自己的学习相关活动、第二个日历记录自己的生活相关活动。

用户向自己拥有的某个日历中添加活动，每个活动具有特定的名称（例如"软件工程期末考试"）、简介（0-200 字）、起始日期和起始时间（例如 2016-12-24 13:00）、结束日期和结束时间（例如 2016-12-24 15:00）、地点（例如致知楼 33）。活动也可以不指定具体的起始/结束时间而只有起始/结束日期（例如从 2016-12-24 到 2016-12-26）。

除了添加活动，用户还可以删除已有活动，修改已有活动、平移已有活动（例如将某活动延迟 3 天进行、将某活动提前 1 小时进行）。

用户希望能在特定日历上执行查询操作、查重操作。"查询"是指：给定某关键词，查到该日历中哪些活动包含该关键词并列出这些活动；"查重"操作是指：给定某日期，看是否活动之间存在重叠（例如：活动 1 的时间范围是 2016-12-24 13:00-15:00，活动 2 的时间范围是 2016-12-24 14:30-18:30，那么它们就有重叠，重叠范围是当日 14:30-15:00），该操作的结果是"活动对"及重叠日期/时间范围（例如： <活动 1，活动 2，2016-12-24, 14:30, 15:00>）。

更复杂的操作是跨日历的查询和查重操作，意即：在用户拥有的多个日历上进行全局查询（例如，满足条件的某些活动来自于日历 1，某些活动来自于日历 2）和全局查重（例如，日历 1 中的某个活动和日历 2 中的某个活动存在重叠）。

下一个操作是日历的合并，即：给定同一用户拥有的两个日历 A 和 B，将 B 中包含的全部活动合并进入 A，然后删除 B。

最后一个操作是计算用户各天的"忙碌度"。给定特定的日期（例如 2016-12-24），在用户的所有日历上计算该日期的忙碌度。"忙碌度"定义为：该天的全部活动的累积小时数 ÷ 24。可以看出，如果某天内的多个活动之间存在重叠，忙碌度可能会大于 1。