گزارش کار پروژه تست 3

سید علی امام زاده 810199377، محمدحسین عقیلی 810199576

شناسه اخرین کامیت: **57e4a5daaee611a0596e160d9bbf1d4398ad0123**

سوال 1)

بله میتوان.

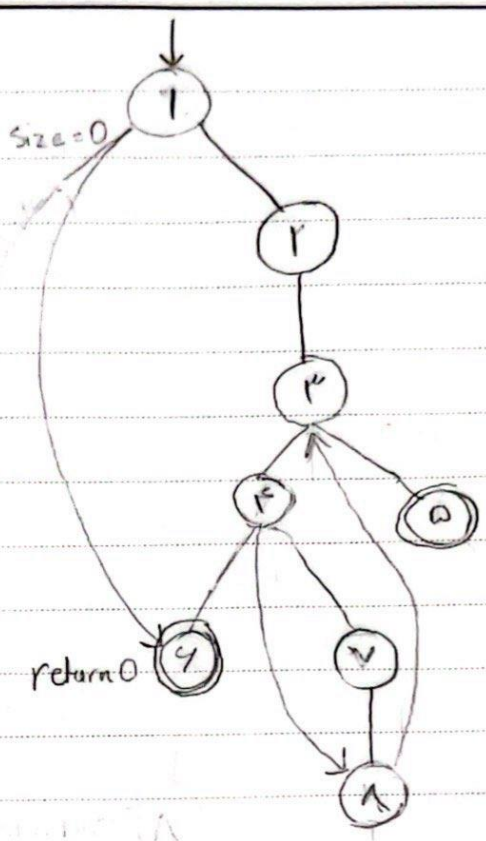
```
public class OrderTest {

    @Test
    public void testEqualsNotInstanceOfOrder() {

        Order order = new Order(1);

        Object otherObject = "not an Order";

        boolean result = order.equals(otherObject);

        assertFalse(result);
    }
}
```

برای دستیابی به پوشش 100٪ جمله اما 100٪ نبودن پوشش شاخه، می توان obj نمونه ای از Order نباشد. به این ترتیب اولین شاخه از دستور if اجرا نمی شود و در نتیجه به پوشش 100٪ شاخه نمی رسد

همچنین اگر کد را بصورتی که در سولل گفته شده تغییر دهیم، باز هم تغییری در تست ما ایجاد نمیشود و به نتیجه مطلوب قسمت قبل خواهیم رسید زیرا متغیر تاثیری بر رفتار متد ما ندارد.

bject: ۸۱۰۱۹۹۵۷۴
ate

محمد حسین عقیلی

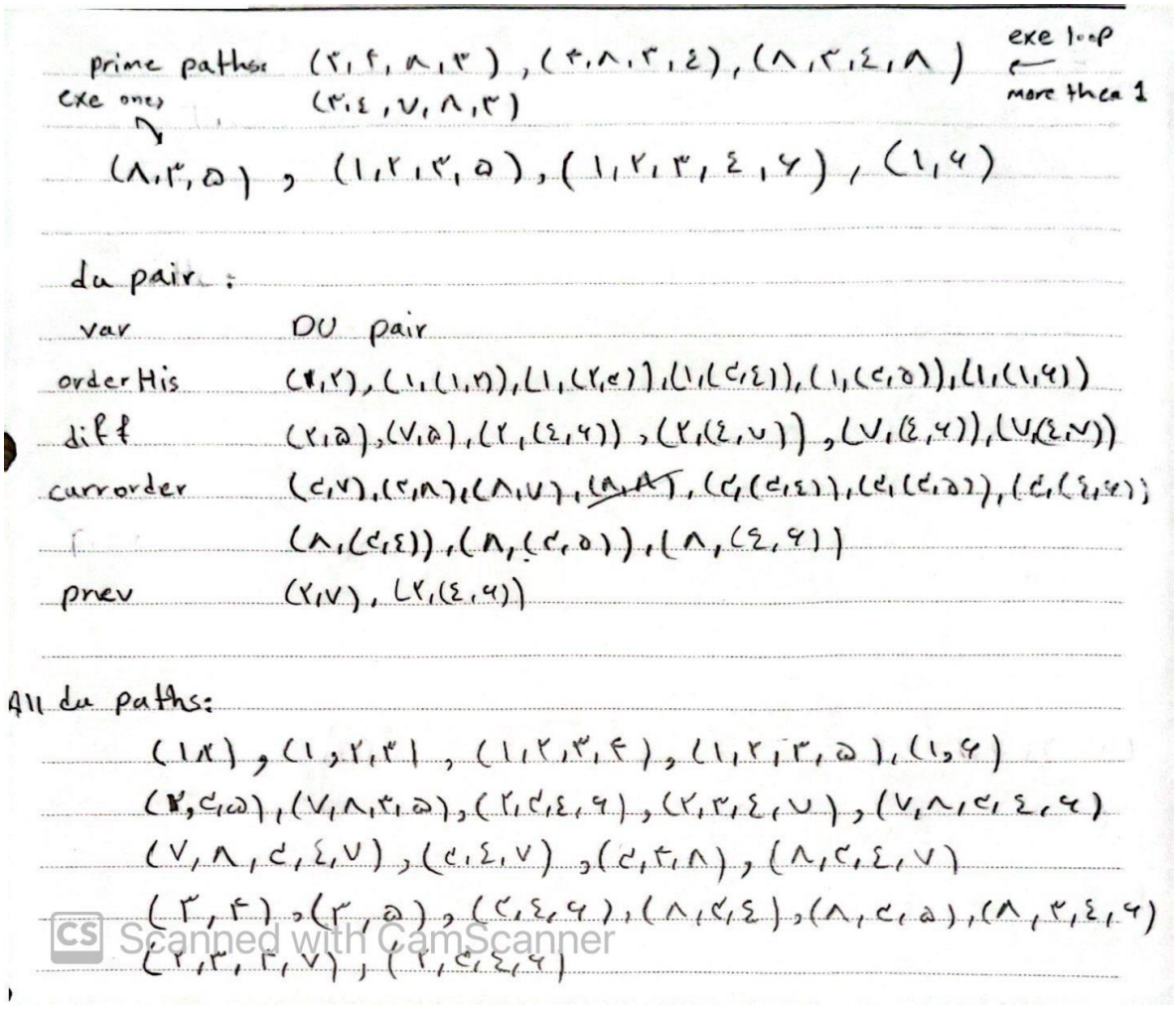


diff

جدول use, def

Node	Def	Use
۱	orderHis	
۲	diff, Prev	orderHis
۳	currorder	
۴		
۵		diff
۶		
۷	diff	currorder, prev
۸	currorder	currorder

Edge	Use
(۱,۲)	orderHis
(۲,۳)	orderHis
(۳,۴)	currorder, orderHis
(۳,۵)	currorde, orderHis
(۴,۶)	diff, currorder, prev
(۴,۷)	diff
(۱,۶)	orderHis
(۷,۸)	
(۸,۳)	
(۴,۸)	



سوال 3)

بله ممکن است. میتوانیم مراحل زیر را دنبال کنیم.

1. شناسایی DU Paths: همه مسیر های du ممکن شناسایی میکنیم.

2. ایجاد تست برای مسیرهای DU: تست هایی مینویسیم که هر مسیر du شناسایی شده را اجرا کند و هر تست شامل سناریوهایی است که متغیرها به روش‌های مختلف تعریف و استفاده می‌شوند.

3. اجتناب از Specific Prime Path: مسیر اصلی خاصی را در برنامه مشخص میکنیم که می خواهیم آن را بدون پوشش رها کنیم. این مسیر اصلی از مسیرهایی که شما شناسایی کرده اید متمایز هستند.

4. طراحی آزمایش‌هایی برای رد شدن از مسیر اصلی: هنگام طراحی تست ها، عمداً از پوشش prime path خاصی که در مرحله 3 شناسایی کرده‌ایم اجتناب کنید و میتواند شامل اجتناب از نتایج تصمیم‌گیری خاص، نادیده گرفتن عبارات خاص یا گرفتن شاخه‌های مختلف در کد باشد.

سوال 4)

به دلایل زیر:

1. معیار مسیر اصلی با در نظر گرفتن مسیرهای مستقل خطی و du paths به دستیابی به coverage تست بالاتر کمک می کند و احتمال تشخیص خطا یا نقص در برنامه را افزایش می دهد.

2. به شناسایی تعاملات پیچیده بین بخش های مختلف برنامه کمک می کند که ممکن است هنگام در نظر گرفتن تنها مسیرهای مستقل خطی مشهود نباشد و برای شناسایی مسائل در تعامل متغیرها و توابع بسیار مهم است.

3. تشخیص روابط Def-Use: شناسایی مسیرهای du برای درک چگونگی تعریف و استفاده متغیرها در برنامه ضروری است و برای شناسایی مشکلات احتمالی مربوط به مقادیر متغیر و اطمینان از اینکه تمام استفاده‌های ممکن از یک متغیر در طول آزمایش در نظر گرفته می‌شود، بسیار مهم است.

4. تشخیص خطا: با در نظر گرفتن مسیرهای مستقل خطی و مسیرهای du، معیار مسیر اصلی شانس تشخیص خطاها، از جمله موارد مربوط به سوء استفاده از متغیر، تعاملات ناخواسته، یا پوشش ناقص کد را افزایش می دهد.