东南大学国家示范性软件学院
College of Software Engineering
Southeast University

软件测试基础与实践

实验报告

实验名称：黑盒测试实验二

实验地点：计算机楼 268

实验日期：2019 年 11 月 20 日

学生姓名：柳沿河

学生学号：71117230

东南大学 软件学院 制

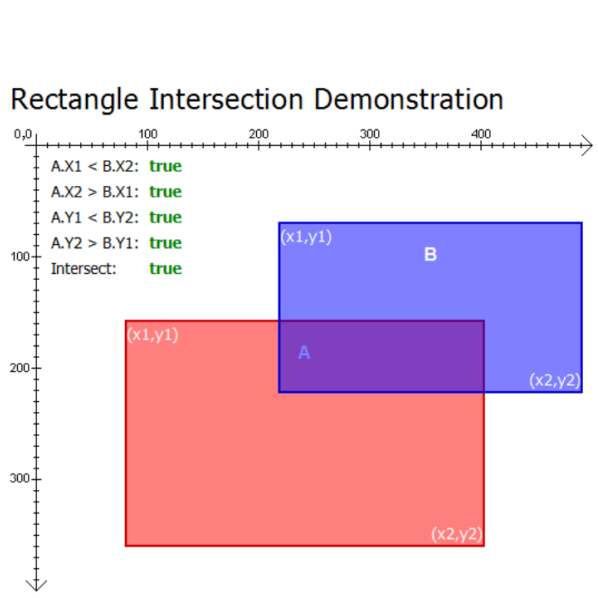
一、实验目的

（1）能根据待测软件的特点，选择合适的方法对软件进行黑盒测试(功能测试)；
（2）了解随机测试，巩固白盒测试和黑盒测试方法；
（3）了解 JUnit 测试开发框架及其应用；
（4）能对一些特定的程序进行蜕变测试。

二、实验内容

（一）题目 1：随机测试 VS 黑盒测试 VS 白盒测试

在游戏引擎开发中，检测物体碰撞是一项重要的基础功能，比如 DOTA 和王者荣耀等游戏中的各种华丽大招的伤害波及范围计算等。
为简单起见，我们这里只考虑二维平面空间的情况，并用 RectManager 程序判断平面上任意两矩形的相交关系（A:不相交，B:相交：B1:相交为一个区域，B12:包含，B13:完全重合，B2:交点为 1 个点，B3:交点为 1 条线段），如果相交，则同时给出相交部分的面积。
这里的二维平面限定为 iphone4 屏幕(640*960 分辨率)，且所有矩形的边都与坐标轴平行。
计算机图形学中，通常用左上角和右下角的坐标来表示一个矩形。
任意两个矩形的关系可借用这个工具来辅助分析：http://silentmatt.com/rectangle-intersection/
坐标系请参照下图：



(1)请编写一简单程序，随机生成两个矩形的数据作为测试用例，请用这些测试用例对 RectManager 进行测试。
要求：
a) 编写程序，实现用随机函数生成大量测试用例（10 万-100 万个），对上述问题进行随机测试。
b) 注意随机测试用例产生的范围应比屏幕范围稍微大一点。屏幕范围：x 取值范围[0-639]，y 取值范围[0-959];
c) 在测试用例生成程序中，同时调用 RectManager 中的方法直接驱动测试自动执行。
d) 对随机测试结果进行统计，分析随机测试用例对两矩形相交的各种关系的覆盖情况（统计上的命中概率）。
e) 给出源代码和测试运行结果。
熟悉 JUnit 的同学，可用 JUnit 实现上述随机测试。
(2)请用黑盒测试方法，设计相应的测试用例来测试程序（可参考并重用实验四中设计的测试用例）；
提示：程序运行命令行：java -jar RectManager.jar
(3)请分析 RectManager 的实现源代码，利用基本路径测试方法对程序进行白盒测试；
只要求针对 **solve()**方法进行测试（只给出基本路径，不用具体设计测试用例）。
(4)在上述实验的基础上分析三种测试方法发现缺陷的能力上有何差别。

答：
（1） 测试代码由两个类构成：
a) TestRectManager 实现对 RectManager 的待测方法的测试
b) TestRunner 负责驱动 TestRectManager

TestRectManager：

```
package ex5;
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

import org.junit.Test;

public class TestRectManager {
   RectManager rectManager = new RectManager();
   Random random = new Random();
   Map<Integer, Long> map = new HashMap<>();
   {
      map.put(-2, (long) 0);
      map.put(-1, (long) 0);
      map.put(0, (long) 0);
      map.put(1, (long) 0);
      map.put(2, (long) 0);
      map.put(3, (long) 0);
      map.put(4, (long) 0);
      map.put(5, (long) 0);
   }

   @Test
   public void testSolve()
   {
      long l = 0;
      while(l < 1_000_000)
        {
            Rect a = new Rect();
            Rect b = new Rect();
            a.left = random.nextInt(660) - 10;//范围：[-10,649]
            a.top = random.nextInt(980) - 10;//范围：[-10,969]
            a.right = a.left + random.nextInt(650 - a.left);//范围：[a.left, 649]
            a.bottom = a.top + random.nextInt(970 - a.top);//范围：[a.top, 969]
            b.left = random.nextInt(660) - 10;//范围：[-10,649]
            b.top = random.nextInt(980) - 10;//范围：[-10,969]
            b.right = b.left + random.nextInt(650 - b.left);//范围：[b.left, 649]
            b.bottom = b.top + random.nextInt(970 - b.top);//范围：[b.top, 969]
            if (!(a.left>=0 && a.right<640)|| !(a.top>=0 && a.bottom<960)
            ||!(a.right>=a.left) || !(a.bottom>=a.top)){
            rectManager.nFlag = -2;
         }
            else if (!(b.left>=0 && b.right<640)|| !(b.top>=0 && b.bottom<960)
            ||!(b.right>=b.left) || !(b.bottom>=b.top)){
            rectManager.nFlag = -1;
         }
            else
               rectManager.solve(a, b);
            map.put(rectManager.nFlag, map.get(rectManager.nFlag) + 1);
            l++;
        }
        System.out.println("------------统计结果----------------------");
        System.out.println("矩形 A 不合法:                                  " +  map.get(-
2) + "     " + (double)map.get(-2)/10_000 + "%");
        System.out.println("矩形 B 不合法:                                  " +  map.get(-
1) + "     " + (double)map.get(-1)/10_000 + "%");
        System.out.println("不相交情
况:                                " +  map.get(0) + "     " + (double)map.get(0)/10_000 + "%");
        System.out.println("相交于一个区
域:                            " +  map.get(1) + "     " + (double)map.get(1)/10_000 + "%");
```

```
        System.out.println("相交于一个区域且为包含关
系:    " +  map.get(2) + "      " + (double)map.get(2)/10_000 + "%");
        System.out.println("相交于一个区域且正好重
合:     " +  map.get(3) + "          " + (double)map.get(3)/10_000 + "%");
        System.out.println("相交于一个
点:                              " +  map.get(4) + "          " + (double)map.get(4)/10_000 + "%");
        System.out.println("相交于一条线
段:                          " +  map.get(5) + "      " + (double)map.get(5)/10_000 + "%");
        System.out.println("----------------------------");

    }
}
```

TestRunner：

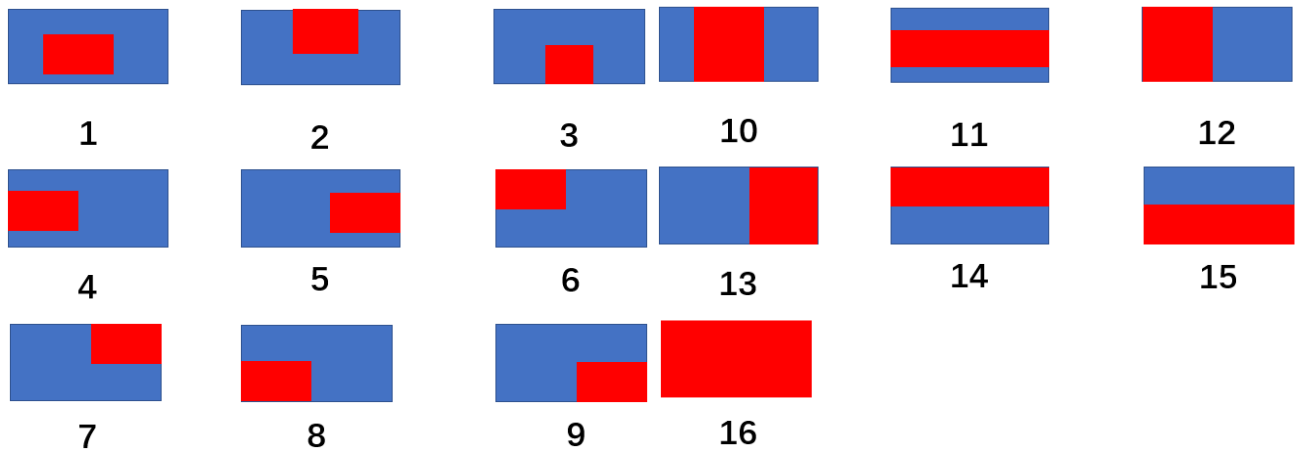
```
package ex5;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner {
    public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestRectManager.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
   }
}
```

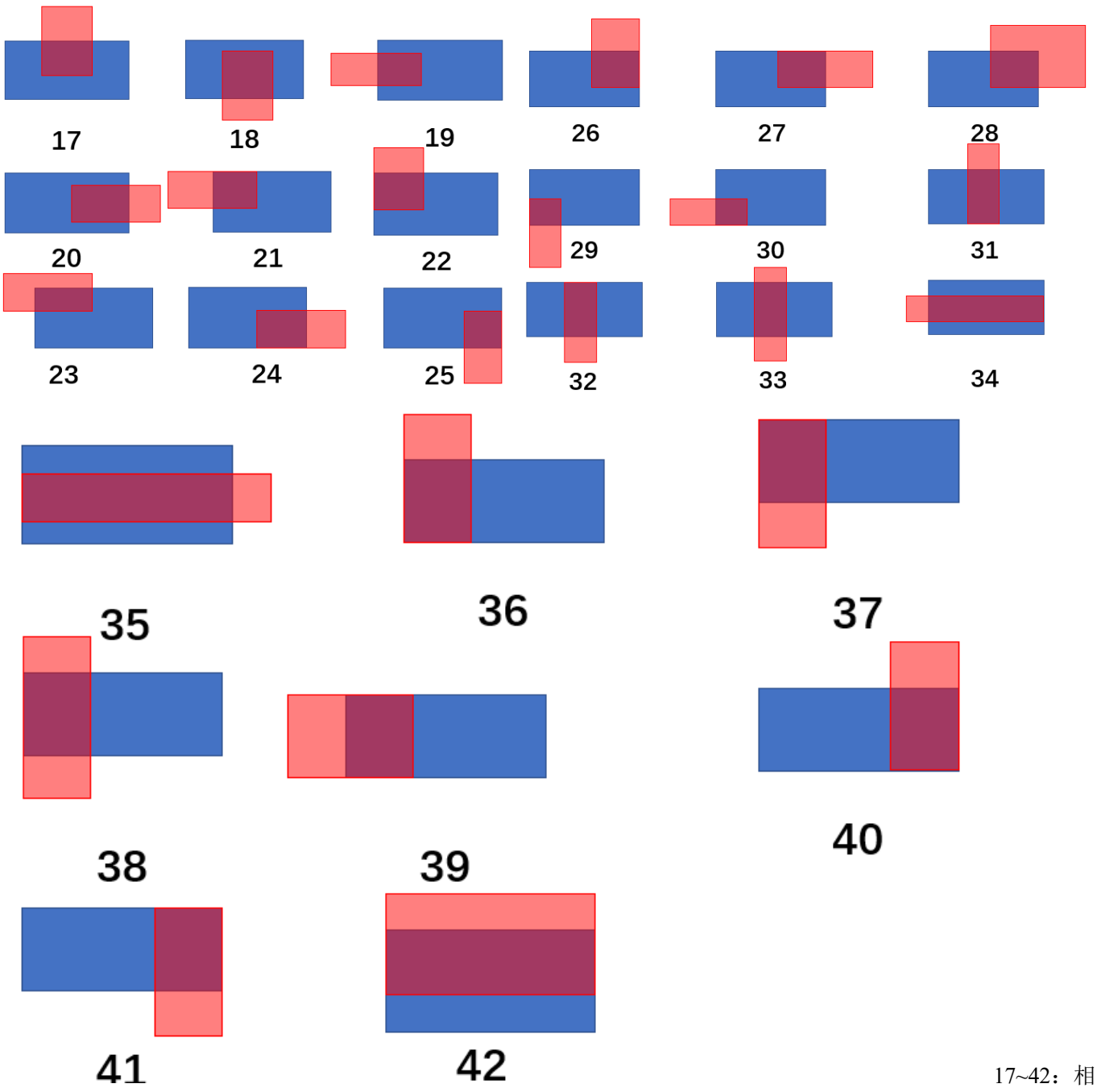
测试结果如下：

```
----------统计结果----------
矩形A不合法:                153545   15.3545%
矩形B不合法:                130183   13.0183%
不相交情况:                 519770   51.977%
相交于一个区域:              168384   16.8384%
相交于一个区域且为包含关系:    25033    2.5033%
相交于一个区域且正好重合：     0        0.0%
相交于一个点：               16       0.0016%
相交于一条线段：             3069     0.3069%
--------------------------------
```

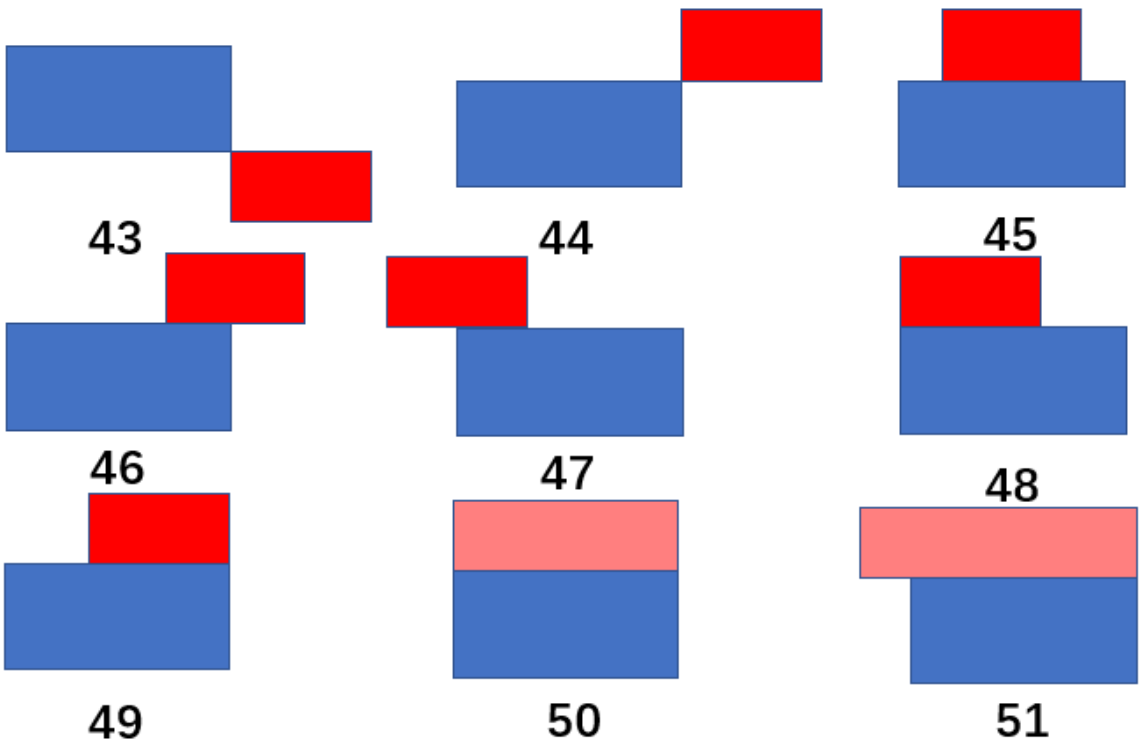
（2）两个矩形有重叠部分时的相对位置的情况如下：



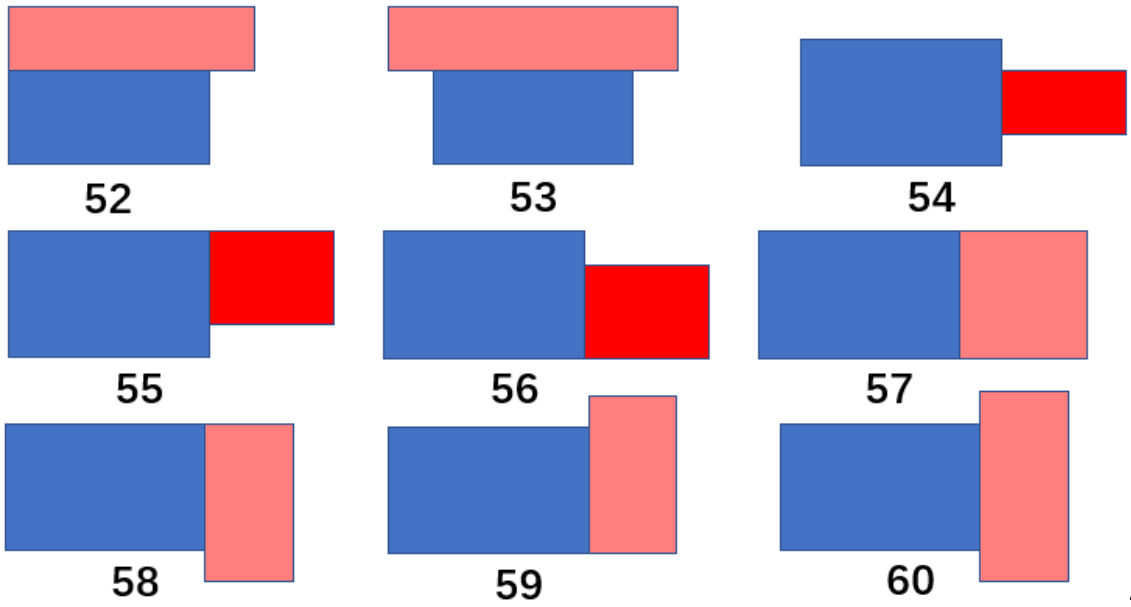
1~16：包含关系和重合（16）



17~42：相交；



43、44：交于一点；

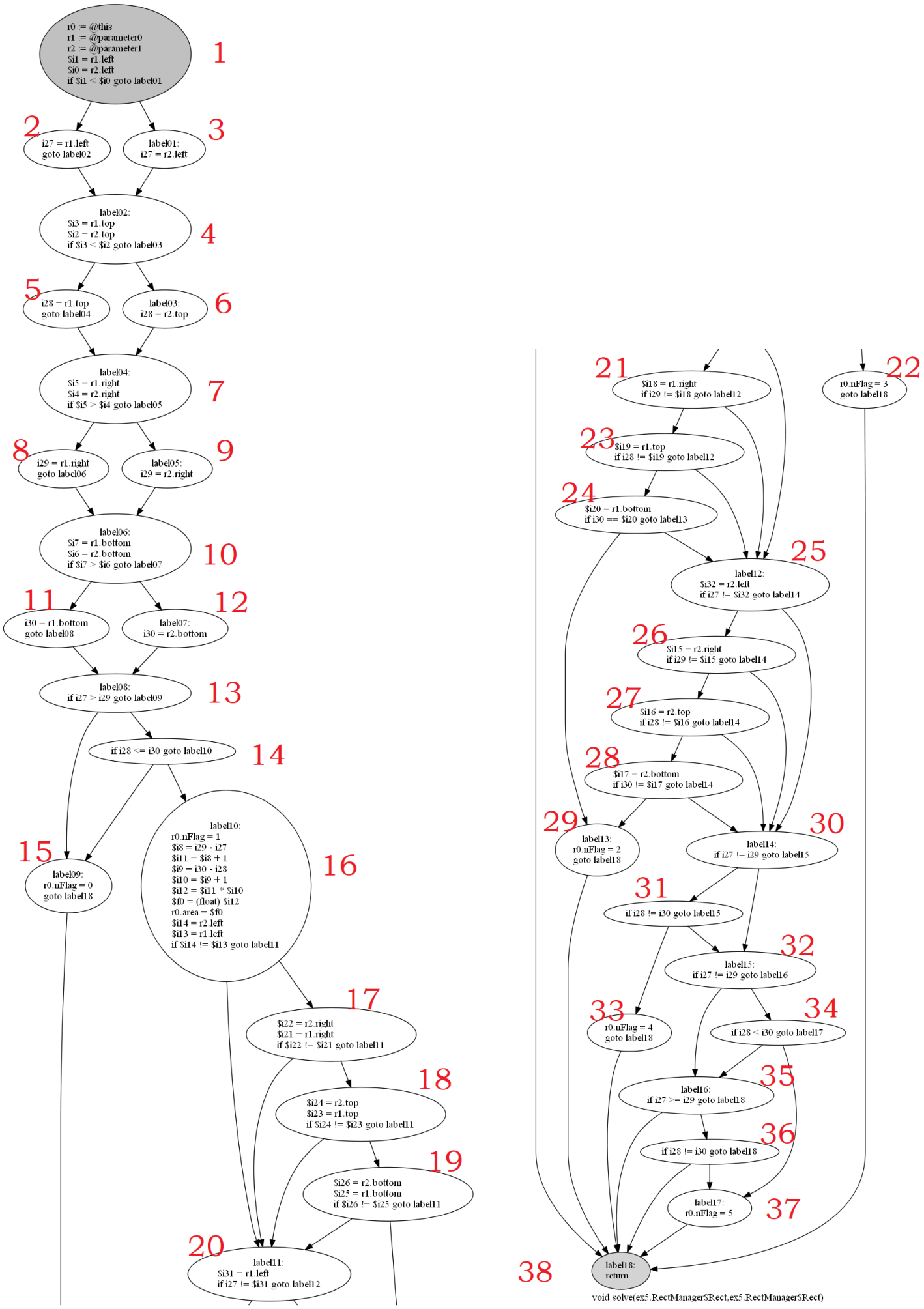


45~60：交于一线段

综合无效等价类后设计测试用例如下：

等价类	输入类型	测试用例								期望输出	实际输出
		矩形A				矩形B					
		x1	y1	x2	y2	x3	y3	x4	y4		
1	有效类	5	4	10	8	6	5	9	7	包含且面积为12	包含且面积为12
2	有效类	5	4	10	8	6	4	9	6	包含且面积为12	包含且面积为12
3	有效类	5	4	10	8	6	6	9	8	包含且面积为12	包含且面积为12
4	有效类	5	4	10	8	5	5	8	7	包含且面积为12	包含且面积为12
5	有效类	5	4	10	8	7	5	10	7	包含且面积为12	包含且面积为12
6	有效类	5	4	10	8	5	4	8	6	包含且面积为12	包含且面积为12
7	有效类	5	4	10	8	7	4	10	6	包含且面积为12	包含且面积为12
8	有效类	5	4	10	8	5	6	8	8	包含且面积为12	包含且面积为12
9	有效类	5	4	10	8	7	6	10	8	包含且面积为12	包含且面积为12
10	有效类	5	4	10	8	6	4	9	8	包含且面积为20	包含且面积为20
11	有效类	5	4	10	8	5	5	10	7	包含且面积为18	包含且面积为18
12	有效类	5	4	10	8	5	4	8	8	包含且面积为20	包含且面积为20
13	有效类	5	4	10	8	7	4	10	8	包含且面积为20	包含且面积为20
14	有效类	5	4	10	8	5	4	10	6	包含且面积为18	包含且面积为18
15	有效类	5	4	10	8	5	6	10	8	包含且面积为18	包含且面积为18
16	有效类	5	4	10	8	5	4	10	8	重合且面积为30	重合且面积为30
17	有效类	5	4	10	8	7	2	9	6	相交且面积为9	相交且面积为9
18	有效类	5	4	10	8	6	5	9	9	相交且面积为16	相交且面积为16
19	有效类	5	4	10	8	2	5	7	7	相交且面积为9	相交且面积为9
20	有效类	5	4	10	8	8	5	13	7	相交且面积为9	相交且面积为9
21	有效类	5	4	10	8	1	4	6	6	相交且面积为6	相交且面积为6
22	有效类	5	4	10	8	5	2	7	6	相交且面积为9	相交且面积为9
23	有效类	5	4	10	8	2	3	6	5	相交且面积为4	相交且面积为4
24	有效类	5	4	10	8	8	6	12	8	相交且面积为9	相交且面积为9
25	有效类	5	4	10	8	8	6	10	10	相交且面积为9	相交且面积为9
26	有效类	5	4	10	8	8	2	10	6	相交且面积为9	相交且面积为9
27	有效类	5	4	10	8	8	4	12	6	相交且面积为9	相交且面积为9
28	有效类	5	4	10	8	8	2	12	6	相交且面积为9	相交且面积为9
29	有效类	5	4	10	8	5	6	7	10	相交且面积为9	相交且面积为9
30	有效类	5	4	10	8	2	7	6	8	相交且面积为4	相交且面积为4
31	有效类	5	4	10	8	7	2	9	8	相交且面积为15	相交且面积为15
32	有效类	5	4	10	8	7	4	9	10	相交且面积为15	相交且面积为15
33	有效类	5	4	10	8	7	2	9	10	相交且面积为15	相交且面积为15
34	有效类	5	4	10	8	2	5	10	7	包含且面积为18	包含且面积为18
35	有效类	5	4	10	8	5	5	12	7	包含且面积为18	包含且面积为18
36	有效类	5	4	10	8	5	2	7	8	相交且面积为15	相交且面积为15
37	有效类	5	4	10	8	5	4	7	10	相交且面积为15	相交且面积为15
38	有效类	5	4	10	8	5	2	7	10	相交且面积为15	相交且面积为15
39	有效类	5	4	10	8	2	4	6	8	相交且面积为10	相交且面积为10
40	有效类	5	4	10	8	8	2	10	8	相交且面积为15	相交且面积为15
41	有效类	5	4	10	8	8	4	10	10	相交且面积为15	相交且面积为15
42	有效类	5	4	10	8	5	2	10	6	包含且面积为18	包含且面积为18
43	有效类	5	4	10	8	10	8	12	10	相交于一点	相交于一点
44	有效类	5	4	10	8	10	2	12	4	相交于一点	相交于一点
45	有效类	5	4	10	8	6	2	8	4	相交于一条线段且面积为3	相交于一条线段且面积为3
46	有效类	5	4	10	8	8	2	12	4	相交于一条线段且面积为3	相交于一条线段且面积为3
47	有效类	5	4	10	8	2	2	7	4	相交于一条线段且面积为3	相交于一条线段且面积为3
48	有效类	5	4	10	8	5	2	7	4	相交于一条线段且面积为3	相交于一条线段且面积为3
49	有效类	5	4	10	8	6	2	10	4	相交于一条线段且面积为5	相交于一条线段且面积为5
50	有效类	5	4	10	8	5	2	10	4	相交于一条线段且面积为6	相交于一条线段且面积为6
51	有效类	5	4	10	8	2	2	10	4	相交于一条线段且面积为6	相交于一条线段且面积为6
52	有效类	5	4	10	8	5	2	12	4	相交于一条线段且面积为6	相交于一条线段且面积为6
53	有效类	5	4	10	8	2	2	12	4	相交于一条线段且面积为6	相交于一条线段且面积为6
54	有效类	5	4	10	8	10	5	12	6	相交于一条线段且面积为2	相交于一条线段且面积为2
55	有效类	5	4	10	8	10	4	12	6	相交于一条线段且面积为3	相交于一条线段且面积为3
56	有效类	5	4	10	8	10	5	12	8	相交于一条线段且面积为4	相交于一条线段且面积为4
57	有效类	5	4	10	8	10	4	12	8	相交于一条线段且面积为5	相交于一条线段且面积为5
58	有效类	5	4	10	8	10	4	12	10	相交于一条线段且面积为5	相交于一条线段且面积为5
59	有效类	5	4	10	8	10	2	12	8	相交于一条线段且面积为5	相交于一条线段且面积为5
60	有效类	5	4	10	8	10	2	12	10	相交于一条线段且面积为5	相交于一条线段且面积为5
61	有效类	5	4	10	8	11	6	12	7	不相交	不相交
62	无效类	5	4	10	8	-1	0	1	2	矩形B输入不合法	矩形B输入不合法
63	无效类	5	4	10	8	1	-2	3	0	矩形B输入不合法	矩形B输入不合法
64	无效类	5	4	10	8	-1	-2	2	4	矩形B输入不合法	矩形B输入不合法
65	无效类	5	4	10	8	3	2	1	4	矩形B输入不合法	矩形B输入不合法
66	无效类	5	4	10	8	3	4	5	1	矩形B输入不合法	矩形B输入不合法
67	无效类	5	4	10	8	3	4	1	2	矩形B输入不合法	矩形B输入不合法

（3）利用 soot 辅助画出程序流图如下：



基本路径如下：

序号	路径
1	1→2→4→5→7→8→10→11→13→15→38
2	1→2→4→5→7→8→10→11→13→14→15→38
3	1→2→4→5→7→8→10→11→13→14→16→20→25→30→31→33→38
4	1→2→4→5→7→8→10→11→13→14→16→20→25→30→31→32→35→38
5	1→2→4→5→7→8→10→11→13→14→16→20→25→30→31→32→35→36→38
6	1→2→4→5→7→8→10→11→13→14→16→20→25→30→31→32→35→36→37→38
7	1→2→4→5→7→8→10→11→13→14→16→20→25→30→31→32→34→35→38
8	1→2→4→5→7→8→10→11→13→14→16→20→25→30→31→32→34→37→38
9	1→2→4→5→7→8→10→11→13→14→16→20→25→30→32→34→37→38
10	1→2→4→5→7→8→10→11→13→14→16→20→25→26→30→31→33→38
11	1→2→4→5→7→8→10→11→13→14→16→20→25→26→27→30→31→33→38
12	1→2→4→5→7→8→10→11→13→14→16→20→25→26→27→28→30→31→33→38
13	1→2→4→5→7→8→10→11→13→14→16→20→25→26→27→28→29→38
14	1→2→4→5→7→8→10→11→13→14→16→20→21→25→26→30→31→33→38
15	1→2→4→5→7→8→10→11→13→14→16→20→21→23→25→26→30→31→33→38
16	1→2→4→5→7→8→10→11→13→14→16→20→21→23→24→25→26→30→31→33→38
17	1→2→4→5→7→8→10→11→13→14→16→20→21→23→24→29→38
18	1→2→4→5→7→8→10→11→13→14→16→17→20→21→23→24→29→38
19	1→2→4→5→7→8→10→11→13→14→16→17→18→20→21→23→24→29→38
20	1→2→4→5→7→8→10→11→13→14→16→17→18→19→20→21→23→24→29→38
21	1→2→4→5→7→8→10→11→13→14→16→17→18→19→20→22→38
22	1→2→4→5→7→8→10→12→13→15→38
23	1→2→4→5→7→9→10→12→13→15→38
24	1→2→4→6→7→9→10→12→13→15→38
25	1→3→4→6→7→9→10→12→13→15→38

（4）由上述三种测试方法比较可得：

发现缺陷的能力：基本路径测试 ＞ 黑盒测试 ＞ 随机测试

分析：基本路径测试作为白盒测试技术的一种，可以深入到代码逻辑，测试比较彻底，发现缺陷能力最强；黑盒测试需要考虑各种输入的情况，容易发生情况的遗漏；随机测试很难发现缺陷。

（二）题目 2：蜕变测试问题

在很多软件测试活动中，人们发现给出一个测试用例的期望输出是一件很困难的事情，在一些情况下，人们甚至无法给出测试用例的预期输出。这种测试预测输出无法获得的问题，就称为测试预言问题(Test Orcale Problem)。为解决 Test Orcale 问题，1998 年 T.Y. Chen 提出的蜕变测试的概念。

游戏引擎中需要高效的计算，所以其中的数学函数都需要重新进行快速实现，不能调用系统的自带数学函数。

如下的程序片段是一个 sin(x)函数的实现代码示例：

```
//always wrap input angle to -PI..PI
if (x < -3.14159265)
    x += 6.28318531;
else
if (x >  3.14159265)
    x -= 6.28318531;

//compute sine
if (x < 0)
    sin = 1.27323954 * x + .405284735 * x * x;
else
    sin = 1.27323954 * x - 0.405284735 * x * x;
```

Sin.exe（该执行程序见课程主页）是一个用某种数值计算方法求解数学函数 f(x)=sin(x)的程序。请设计测试用例，实现对该程序的黑盒测试。

(1) 给出每个测试用例设计的理由；

(2) 这个实验中展示的测试用例输出值无法预测的情形还可能出现在哪些实际应用中。

答：

（1） 测试用例如下：

高亮结果为错误结果。

（2） 还可能出现在的应用：其他各种数学计算函数、无向图中的最短路径问题、电脑图形图像软件产生图像且在屏幕上打印图像、桥梁应力测试等。

序号	输入	设计理由	期望输出	实际输出
1	0	特殊值且sin(x)=-sin(-x)	0	0
2	2Pi	sin(x)=sin(x+2π) sin(x)=-sin(2π-x)	0	3.11E-07
3	Pi	sin(x)=-sin(x+π) sin(x)=sin(π-x)	0	-7.73E-07
4	-Pi	sin(x)=-sin(-x)	0	-3.14159
5	Pi/6	特殊值	0.5	0.5
6	13Pi/6	sin(x)=sin(x+2π)	0.5	0.5
7	7Pi/6	sin(x)=-sin(x+π)	-0.5	-0.5
8	-Pi/6	sin(x)=-sin(-x)	-0.5	-0.523599
9	5Pi/6	sin(x)=sin(π-x)	0.5	0.500001
10	11Pi/6	sin(x)=-sin(2π-x)	-0.5	-0.500001
11	Pi/4	特殊值	0.707	0.707106
12	9Pi/4	sin(x)=sin(x+2π)	0.707	0.707106
13	5Pi/4	sin(x)=-sin(x+π)	-0.707	-0.707105
14	-Pi/4	sin(x)=-sin(-x)	-0.707	-0.785398
15	3Pi/4	sin(x)=sin(π-x)	0.707	0.707107
16	7Pi/4	sin(x)=-sin(2π-x)	-0.707	-0.707107
17	Pi/3	特殊值	0.866	0.866025
18	7Pi/3	sin(x)=sin(x+2π)	0.866	0.866025
19	4Pi/3	sin(x)=-sin(x+π)	-0.866	-0.866026
20	-Pi/3	sin(x)=-sin(-x)	-0.866	-1.0472
21	2Pi/3	sin(x)=sin(π-x)	0.866	0.866025
22	5Pi/3	sin(x)=-sin(2π-x)	-0.866	-0.866024
23	Pi/2	特殊值且sin(x)=sin(π-x)	1	1
24	5Pi/2	sin(x)=sin(x+2π)	1	1
25	3Pi/2	sin(x)=-sin(x+π) sin(x)=-sin(2π-x)	-1	-1
26	-Pi/2	sin(x)=-sin(-x)	-1	-1.5708
27	37°	随机	0.602	0.601815
28	397°	sin(x)=sin(x+2π)	0.602	0.601815
29	217°	sin(x)=-sin(x+π)	-0.602	-0.601814
30	-37°	sin(x)=-sin(-x)	-0.602	-0.645772
31	143°	sin(x)=sin(π-x)	0.602	0.601816
32	323°	sin(x)=-sin(2π-x)	-0.602	-0.601815
33	53°	随机	0.8	0.798636
34	413°	sin(x)=sin(x+2π)	0.8	0.798636
35	233°	sin(x)=-sin(x+π)	-0.8	-0.798636
36	-53°	sin(x)=-sin(-x)	-0.8	-0.925025
37	127°	sin(x)=sin(π-x)	0.8	0.798636
38	307°	sin(x)=-sin(2π-x)	-0.8	-0.798633
39	165°	随机	0.259	0.258819
40	525°	sin(x)=sin(x+2π)	0.259	0.258819
41	345°	sin(x)=-sin(x+π)	-0.259	-0.258821
42	-165°	sin(x)=-sin(-x)	-0.259	-2.87979
43	15°	sin(x)=sin(π-x)	0.259	0.258819
44	195°	sin(x)=-sin(2π-x)	-0.259	-0.258819
45	236°	随机	-0.829	-0.829038
46	596°	sin(x)=sin(x+2π)	-0.829	-0.829038
47	416°	sin(x)=-sin(x+π)	0.829	0.829038
48	-236°	sin(x)=-sin(-x)	0.829	-4.11898
49	-56°	sin(x)=sin(π-x)	-0.829	-0.977384
50	124°	sin(x)=-sin(2π-x)	0.829	0.829038
51	299°	随机	-0.875	-0.874619
52	659°	sin(x)=sin(x+2π)	-0.875	-0.874619
53	479°	sin(x)=-sin(x+π)	0.875	0.87462
54	-299°	sin(x)=-sin(-x)	0.875	-5.21853
55	-119°	sin(x)=sin(π-x)	-0.875	-2.07694
56	61°	sin(x)=-sin(2π-x)	0.875	0.87462
57	544°	随机	-0.07	-0.0697576
58	904°	sin(x)=sin(x+2π)	-0.07	-0.0697576
59	724°	sin(x)=-sin(x+π)	0.07	0.0697565
60	-544°	sin(x)=-sin(-x)	0.07	-9.49459
61	-364°	sin(x)=sin(π-x)	-0.07	-6.353
62	-184°	sin(x)=-sin(2π-x)	0.07	-3.21141

三、实验体会

（1）通过测试，是否发现程序中存在的缺陷？
（2）经过一系列的实验，请谈谈你所感受的软件测试中存在哪些挑战性的难题。
（3）随机测试中，对很多同学的随机数生成范围来说，矩形不相交的统计概率不是 **0.5**，你的实验是否存在这种现象？尝试用概率知识对结果进行分析一下。
（4）蜕变关系是否容易发现？蜕变测试的可适用场景有哪些？如果是 tan(x), log2(x)等函数的实现，其蜕变测试怎么做呢？需要寻找什么蜕变关系？

答：
（1） 经过测试发现 Sin.exe 存在缺陷，具体表现为在计算负角度的正弦值的误差碎角度的减小而增大
（2） 挑战性的难题：
  a) 测试用例的设计：要设计覆盖全面且不重复的测试用例非常困难
  b) 对难测试的问题的测试：例如有些程序中存在不可达路径、有些程序无法获得预期结果等。
  c) 测试的自动化：目前测试工作主要由人力完成，很难由机器主导。
（3） 我的实验中结果较为接近 0.5。出现不是 0.5 的情况可能是因为横向或纵向两端增大的量不相等，例如将原来的[0,639]变大到[-10, 659]，最小值减少了了 10 个单位而最大值扩大了 20 个单位，所以影响了分布概率。
（4） 蜕变关系不太容易发现，需要去深入思考和挖掘。蜕变测试适用于无法得知预期输出的测试场景。若测试其他数学函数如 tan，log 等，可以通过其函数性质如周期性和对称性等来设计测试用例完成测试。