Software Test Homework2

- Name：Chen Shuyu
- ID：3017218119
- Class：3

> Key content of homework:
>
> **RIPR** Model
>
> `Reachability` The location or locations in the program that contain the fault must be reached
>
> `Infection` The state of the program must be incorrect
>
> `Propagation` The infected state must cause some output or final state of the program to be incorrect
>
> `Revealability` The tester must observe part of the incorrect portion of the final program state

Program1

1.Identify the fault.

The fault of the first program：

> Line 7: 'i>0' in for loop statement
>
> 'i' can not equal to 0, so the first element in the x array cannot be accessed.

```
for(int i = x.length-1;i>0;i--)
// i>0(×) --> i>=0(√)
```

2.If possible identify a test case that does not execute the fault. (Reachability)

```
test: x = []; y = 3
Expected: -1
Actual: -1
Error: none
Failure: none
```

3. If possible, identify a test case that executes the fault, but does not result in an error state. （Infection）

```
test: x = [1,2,3]; y = 3
Expected: 2
Actual: 2
Error: none
Failure: none
```

4.If possible identify a test case that results in an error, but not a failure. （Propagation）

```
test: x = [1,2,3]; y = 0
Expected: -1
Actual: -1
Error: i has not reached 0
Failure: none
```

Program2

1.Identify the fault.

The fault of the second program：

> Line6: 'int i = 0; i < x.length; i++' in for loop statement
>
> It is wrong to increase 'i' from 0. In this way, the position of the first 0 is found. It should be reduced from 'x.length-1' so that position of the last 0 can be accurately found.

```
for(int i = 0; i < x.length-1; i++) //(×)
// for(int i = x.length-1;i>=0;i--) (√)
```

2.If possible identify a test case that does not execute the fault. (Reachability)

```
test: x = []
Expected: -1
Actual: -1
Error: none
Failure: none
```

3. If possible, identify a test case that executes the fault, but does not result in an error state. （Infection）

```
test: x = [0]
Expected: 0
Actual: 0
Error: none
Failure: none
```

4.If possible identify a test case that results in an error, but not a failure. （Propagation）

```
test: x = [1,2,0]
Expected:2
Actual:2
Error: i is not reduced from x.length-1
Failure: none
```