Homework 1

Below are four faulty programs. Each includes test inputs that result in failure. Answer the following questions about each program.

```
/**
* Find last index of element
*
* @param x array to search
* @param y value to look for
* @return last index of y in x; -1 if absent
* @throws NullPointerException if x is null
*/
public int findLast (int[] x, int y)
{
   for (int i=x.length-1; i > 0; i--)
   {
        if (x[i] == y)
        {
            return i;
        }
   }
   return -1;
}
// test: x = [2, 3, 5]; y = 2; Expected = 0
// Book website: FindLast.java
// Book website: FindLastTest.java
```

```
/**
* Find last index of zero
*
* @param x array to search
*
* @return last index of 0 in x; -1 if absent
* @throws NullPointerException if x is null
*/
public static int lastZero (int[] x)
{
   for (int i = 0; i < x.length; i++)
   {
        if (x[i] == 0)
        {
            return i;
        }
    }
    return -1;
}
// test: x = [0, 1, 0]; Expected = 2
// Book website: LastZero.java
// Book website: LastZeroTest.java
```

```
/**
* Count positive elements
*
* @param x array to search
* @return count of positive elements in x
* @throws NullPointerException if x is null
*/
public int countPositive (int[] x)
{
    int count = 0;
    for (int i=0; i < x.length; i++)
    {
        if (x[i] >= 0)
        {
            count++;
        }
    }
    return count;
}
// test: x = [-4, 2, 0, 2]; Expected = 2
// Book website: CountPositive.java
// Book website: CountPositiveTest.java
```

```
/**
* Count odd or postive elements
*
* @param x array to search
* @return count of odd/positive values in x
* @throws NullPointerException if x is null
*/
public static int oddOrPos(int[] x)
{
    int count = 0;
    for (int i = 0; i < x.length; i++)
    {
        if (x[i]%2 == 1 || x[i] > 0)
        {
            count++;
        }
     }
     return count;
}
// test: x = [-3, -2, 0, 1, 4]; Expected = 3
// Book website: OddOrPos.java
// Book website: OddOrPosTest.java
```

(a) Explain what is wrong with the given code. Describe the fault precisely by proposing a modification to the code.

答：**fault code用紅色標記**

findLast.java :

for (int i=x.length-1; **i > 0**; i--) 改成 **i >= 0**. **錯誤原因：終止條件錯誤，應該涵蓋index 0**

lastZero.java :

for (**int i = 0; i < x.length; i++**) 改成 for (**int i = x.length-1; i >= 0; i--**). **錯誤原因：起始、終止條件錯誤，原本程式會找到第一個0的index**

countPositive.java :

if (**x[i] >= 0**) 改成 x[i] > 0. **錯誤原因：value 0 not a positive number**

oddOrPos.java :

if (**x[i]%2 == 1** || x[i] > 0) 改成 x[i]%2 != 0. **錯誤原因：Java中負數取餘數後結果是負數(e.g. -3%2=-1)**

(b) If possible, give a test case that does not execute the fault. If not, briefly explain why not.

答：

findLast.java :

x = []; y = 2; Expected = -1

lastZero.java :

x = []; Expected = -1

countPositive.java :

x = [-4, 2, 3, 2]; Expected = 3, 原因:不存在 0 值則不會產生fault

oddOrPos.java :

x = [-4, -2, 0, 1, 4]; Expected = 2, 原因: 負數值中若不存在odd element則不會有fault

(c) If possible, give a test case that executes the fault, but does not result in an error state. If not, briefly explain why not.

答：

findLast.java :

x = [2, 2, 4], target = 2. (Output: 1, Expected: 1)

lastZero.java :

x = [0] (Output: 0, Expected: 0)

countPositive.java :

Impossible, 要出現fault, 測資就一定要出現0, 但有0就會造成count錯誤

oddOrPos.java :

Impossible, 要出現fault, 測資就一定要有負奇數, 但有負奇數就一定會造成count錯誤

(d) If possible, give a test case that results in an error state, but not a failure. Hint: Don't forget about the program counter. If not, briefly explain why not.

答：

findLast.java :

x = [1, 3, 4], target = 2. (Output: -1, Expected: -1)

lastZero.java :

x = [1, 1] (Output: -1, Expected: -1)

countPositive.java :

Impossible，要出現Error state，測資就一定要出現0，但有0則count就會錯誤造成Failure

oddOrPos.java :

Impossible，要出現fault，測資就一定要有負奇數，但有負奇數就一定會造成count錯誤而出現Failure

(e) For the given test case, describe the first error state. Be sure to describe the complete state.

答：

findLast.java :

test: x = [2, 3, 5]; y = 2; Expected = 0

When i = 2, no error

When i = 1, no error

When i = 0, first **Error** occur.

lastZero.java :

test: x = [0, 1, 0]; Expected = 2

When i = 0, first **Error** occur because i shall be 2. And x[0] is 0, so it return index 0 as result.

countPositive.java :

test: x = [-4, 2, 0, 2]; Expected = 2

When i = 0, no Error

When i = 1, no Error, count = 1

When i = 2, first **Error** occur. count = 2, but it shall be 1

When i = 3, Error state. count = 3, but it shall be 2

oddOrPos.java :

test: x = [-3, -2, 0, 1, 4]; Expected = 3

When i = 0, first **Error** occur. Because -3%2==-1, count=0. Count shall be 1

When i = 1, Error state, count = 0, it shall be 1

When i = 2, Error state. count = 0, it shall be 1

When i = 3, Error state. count = 1, but it shall be 2

When i = 4, Error state. count = 2, but it shall be 3

(f) Implement your repair and verify that the given test now produces the expected output. Submit a screen printout or other evidence that your new program works.

答：

(A.)

```java
import static java.lang.System.out;

public class HW1 {
    public static int findLast (int[] x, int y)
    {
        for (int i=x.length-1; i >= 0; i--)
        {
            if (x[i] == y)
            {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] test_case = {2, 3, 5};
        int y = 2;
        int result = findLast(test_case, y);
        System.out.println(result);
    }
};
```

```
Run: HW1
0
Process finished with exit code 0
```

(B.)

```java
import static java.lang.System.out;

public class HW1 {
    public static int lastZero (int[] x)
    {
        for (int i = x.length-1; i >= 0; i--)
        {
            if (x[i] == 0)
            {
                return i;
            }
        }
        return -1;

    }

    public static void main(String[] args) {
        int[] test_case = {0, 1, 0};
        int result = lastZero(test_case);
        System.out.println(result);
    }
};
```

```
Run: HW1
2
Process finished with exit code 0
```

(C.)

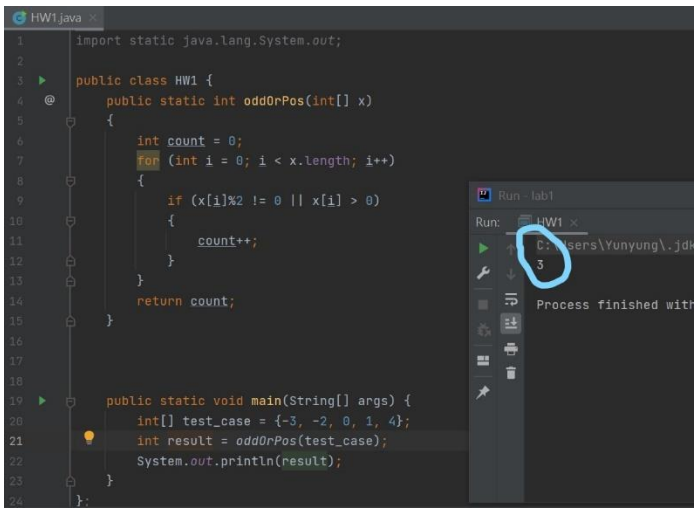
```java
import static java.lang.System.out;

public class HW1 {
    public static int countPositive (int[] x)
    {
        int count = 0;
        for (int i=0; i < x.length; i++)
        {
            if (x[i] > 0)
            {
                count++;
            }
        }
        return count;
    }


    public static void main(String[] args) {
        int[] test_case = {-4, 2, 0, 2};
        int result = countPositive(test_case);
        System.out.println(result);
    }
};
```

```
Run: HW1
2
Process fi
```

(D.)


HW1.java
import static java.lang.System.out;
public class HW1 {
public static int oddOrPos(int[] x)
{
int count = 0;
for (int i = 0; i < x.length; i++)
{
if (x[i]%2 != 0 || x[i] > 0)
{
count++;
}
}
return count;
}
public static void main(String[] args) {
int[] test_case = {-3, -2, 0, 1, 4};
int result = oddOrPos(test_case);
System.out.println(result);
}
};
Run - lab1
Run:
HW1
3
Process finished with