Ejercicios: para cada método de las siguientes 4 transparencias.

1) countPositive

1.1 – El fallo está en el mayor igual a cero, siendo el cero un número neutro (no positivo). Solo tenemos que quitar el igual o cambiar el cero por un uno.

1.2 – Un test de prueba podría ser pasarle al programa un array vacío (test arrayEmpty).

1.3 – Usamos el test arrayWithoutZero, que cuenta los positivos de un array, es decir, que no tienen ningún cero en el array.

1.4 – En este caso es imposible; al leer algún cero terminará siempre con una disfunción.

1.5 – No hay ningún estado erróneo sin una disfunción a posteriori.

1.6 – Todo funciona a la perfección en el momento que corregimos el código como hemos indicado en el punto 1.1.

2) lastZero

2.1 – El programa no retorna el último cero, sino el primer cero que se encuentra en caso de que haya alguno.

2.2 – Al igual que en el caso anterior, la única forma de ejecutar código sin que llegue al fallo es pasarle un array vacío, es decir, el test arrayEmpty.

2.3 – Uso el test arrayWithZero que contiene un único cero en el array.

2.4 – Al igual que en el caso anterior en este punto, no es posible ya que una vez que retorna el valor provocando el error, esto, provoca la disfunción.

2.5 – No hay ningún estado erróneo sin su posterior disfunción.

2.6 – Tenemos que cambiar parte del código, quitando el return del bucle y añadiendo variables de control. Se puede suponer también que el bucle empiece por el final del array, y así se obtendría el último cero como el que hay al comienzo del array.

3) findLast

3.1 – El error es que no se comprueba el primer elemento del arrary (i > 0). La solución es sustituir ese mayor que por un mayor o igual que.

3.2 – Por la misma razón que en los dos ejercicios anteriores, la única forma de que no se llegue a ejecutar el for es usando un array vacío (arrayEmpty).

3.3 – Si se ejecuta el fallo, se provoca inmediatamente un error en el estado, por lo que no es factible crear un test así.

3.4 – Cualquier array que no contenga el número que no estamos buscando en la primera selección del array (arrayNumberNotFirst).

3.5 – El único estado erróneo es la no comprobación de la primera posición del array.

3.6 – Cuando realizamos el cambio que nombramos, funciona correctamente.

4) oddOrPos

4.1 – Este programa no reconoce los números negativos impares, ya que el módulo devuelve -1. Tenemos que cambiar el módulo para que elimine los pares, es decir, (x[i]%2 != 0)

4.2 – El programa pasa sí o sí por la condición, por lo que hay que provocar una excepción por lista vacía, es decir, arrayEmpty.

4.3 – Introduciendo un array que sólo tenga número impares positivos o positivos tan sólo (arrayPositive).

4.4 – La lectura de un número negativo impar provocará el fallo y la posterior disfunción.

4.5 – No hay ningún estado erróneo sin una posterior disfunción.

4.6 – Realizamos el cambio y funciona correctamente, como en todos los casos anteriores.