Respuestas a preguntas del test

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 que no tiene ningún cero.
1.4)
En este caso es imposible: al leer algún cero es imposible que no se acabe con una disfunción.
1.5)
No hay ningún estado erróneo sin su posterior disfunción.
1.6)
Todo funciona correctamente al quitar el igual.

2) lastZero

2.1)
El programa no retorna el ultimo cero, sino el primer cero que se encuentre en caso de que haya alguno. Mi propuesta es añadir variables de control y que la función solo retorne al final.
2.2)
Al igual que en el caso anterior, la única forma de ejecutar código sin que llegue al fallo es pasándole un array vacío (test arrayEmpty,).
2.3)
Usando el test arrayWithZero, que solo contiene un único cero.
2.4)
No es posible, ya que una vez retorna el valor provocando el error, 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 (mi método). También puede suponerse que el bucle empiece por el final, y así sí que se obtendría el último cero como el primero.

3) findLast

3.1)
El error es que no se comprueba el primer elemento del array (i > 0). La solución es sustituir el “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 es imposible crear un test así.
3.4)
Cualquier array que no contenga el número que estamos buscando en la primera posición del array (arrayNumberNotFirst).

3.5)
El único estado erróneo es la no comprobación de la primera posición del array.
3.6)
Realizamos el cambio y funciona correctamente.

4) oddOrPos

4.1)
Este programa no reconoce los número negativos impares, ya que el módulo devuelve -1.
Cambiamos el módulo para que elimine los pares (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 (arrayEmpty).
4.3)
Introduciendo un array que solo tenga número impares positivos o positivos únicamente (arrayPositive).
4.4)
La lectura de un número negativo impar provocará el fallo y su posterior disfunción.
4.5)
No hay ningún estado erróneo sin su posterior disfunción.
4.6)
Realizamos el cambio y funciona correctamente.