SEILA OLIVA MUÑOZ

PRÁCTICA TEST

FindLast

1. FALLO: en la función findLast, el bucle no contempla que i=0, ya que tiene como límite i>0, por tanto, al 0 no llega. Para solucionarlo habría que poner que el bucle llegue hasta i>=0.

2. El único caso en el que no se ejecute el código que tiene el fallo es no introducir ningún valor cuando ejecutamos el programa, ya que en este caso nos salta el mensaje avisándonos que hay que introducir valores y el programa se acaba. En todos los casos en los que introducimos algún valor el código con el error se ejecuta.

3. java FindLast 1 2 3 4
Enter an integer you want to find:
3

→ The index of the last element equal to 3 is: 2

Aquí vemos que el resultado es correcto, a pesar de ejecutarse el código que contiene el error. Esto es posible porque entra en el bucle, pero como encuentra el número antes de llegar a 0 no continúa y por tanto no llega a provocar un error en el estado.

4. No es posible, ya que la disfunción se produce cuando se produce el error en el estado: cuando el número que estoy buscando se encuentra en la primera posición de los números que he introducido al ejecutar el programa, el bucle no llega a contemplar esa posición (i nunca es 0 en el bucle → error) y nos devuelve -1 porque no ha comprobado dicho número, cuando nos debería devolver 0 (disfunción).

5. No había caso de prueba posible en el apartado anterior.

6. Cambio en el bucle i>0 por i>=0.
Cuando ejecuto los test todo está correcto → OK (2 test).

LastZero

1. El fallo se encuentra en el bucle de la función lastZero. Lo que hace este bucle es encontrar el primer cero, en vez del último. Esto es porque empieza por el principio del array, y cuando encuentra un 0 para. Debería empezar por el final del bucle e ir hacia atrás, de esta manera el primer cero que encuentre es el último del array.

2. El único caso en el que no se ejecute el código que tiene el fallo es no introducir ningún valor cuando ejecutamos el programa, ya que en este caso nos salta el mensaje avisándonos que hay que introducir valores y el programa se acaba. En todos los casos en los que introducimos algún valor el código con el error se ejecuta.

3. LastZero 0
The last index of zero is: 0

Este caso ejecuta el código con error porque entra en el bucle, pero no produce un error en el estado porque al tener sólo un valor el array, el bucle sólo da una vuelta, así que da igual que el bucle vaya hacia delante o hacia atrás, i va a ser lo mismo en ese momento.

4. LastZero 1 2 3 4
The last index of zero is: -1

Aquí se ejecuta el código con error (entra en el bucle) y se produce error en el estado (el bucle se recorre al revés), pero no provoca ninguna disfunción ya que no hay ningún cero y el bucle no se para en ningún sitio (ni correcto ni incorrecto).

5. Cuando introducimos los valores en la entrada, el programa los trata y llama a la función lastZero. En esta función, en el bucle, el primer estado es i=0. Esto es un error en el estado, ya que debería empezar por el último elemento del array, es decir, i=x.length-1.

6. En el bucle: *for (int i = x.length-1; i >= 0; i--)*
Ejecuto las pruebas → OK (3 test)

CountPositive

1. El fallo está dentro del bucle, en el if. El if comprueba si el número es mayor o igual a 0, y de esta manera considera que el 0 es un número positivo. Debería comprobar sólo si el número es mayor que 0.

2. El único caso en el que no se ejecute el código que tiene el fallo es no introducir ningún valor cuando ejecutamos el programa, ya que en este caso nos salta el mensaje avisándonos que hay que introducir valores y el programa se acaba. En todos los casos en los que introducimos algún valor el código con el error se ejecuta.

3. CountPositive 1 -2 3
Number of positive number is: 2

En este caso no introducimos ningún 0, que es el valor en discordia. Así, se ejecuta el código que contiene el fallo (la línea del if), pero como ningún valor es 0, no entra en el if cuando no debe, y por tanto no se produce error en el estado (que el contador del programa estuviera dentro del if cuando el valor es 0).

4. No es posible provocar un error en el estado sin producir una disfunción, ya que cuando se produce un error en el estado (entra en el if siendo 0) el contador de números positivos aumenta, por lo que se produce la disfunción.

5. No es posible un caso anterior.

6. En el if de countPositive: *if (x[i] > 0)*
Ejecuto las pruebas → OK (2 test)

OddOrPos

1. El fallo está en la función OddOrPos, dentro del bucle, en el if. Para buscar los números que son impares, utiliza *x[i]%2 == 1.* Lo que ocurre con esta sentencia, es que al calcular el resto de un número negativo, dicho resto da también negativo (-1), por lo que no lo considera impar. Para solucionarlo, podemos hacer el valor absoluto de *x[i]%2*.

2. El único caso en el que no se ejecute el código que tiene el fallo es no introducir ningún valor cuando ejecutamos el programa, ya que en este caso nos salta el mensaje avisándonos que hay que introducir valores y el programa se acaba. En todos los casos en los que introducimos algún valor el código con el error se ejecuta.

3. OddOrPos -4 -2 0 1 3 4
Number of elements that are either odd or positive is: 3

En este caso, el código con fallo se ejecuta para comprobar si los números son impares o positivos, pero no se produce error de estado porque no comprueba ningún número impar negativo.

4. En este caso, no es posible que se produzca un error de estado sin que se produzca una disfunción, ya que si se produce el error en el if, el contador de programa no va a entrar en el if y por tanto no se va a sumar el contador cuando sí debería sumarse, y por tanto la salida del programa va a ser errónea.

5. No hay caso de prueba posible.

6. En la función OddOrPos, en el if del bucle: *if (Math.abs(x[i]%2) == 1 || x[i] > 0)*
Ejecutando las pruebas → OK (2 test)