Andrei Buciulea.

1. Explica el fallo (qué está mal en el código). Describe de manera precisa el fallo y propón una modificación al código que lo arregle.

FindLast.java: se puede observar que al ejecutar el programa si el elemento buscado es el primero no es capaz de encontrarlo y devuelve index -1. Esto se debe a que en la sentencia del bucle for no tiene en cuenta el valor de i = 0; *for (int i=x.length-1; i>0; i--)*. Una solución para este fallo sería incluir el valor de i=0; *for (int i=x.length-1; i >= 0; i--).*

Lastzero.java: busca el último cero de una cadena. En este caso el bucle for encuentra el primer cero en vez de el último. Esto se debe a que está mal escrita la condición el for:
for (int i = 0; i < x.length; i++). Para solucionar dicho fallo es necesario cambiar la condicion y la inicialización de la variable i: *for (int i = x.length-1; i >= 0; i--)*

CountPositive.java: el programa cuenta los números positivos que hay en una cadena. En este caso cuenta como numeros positivos los ceros *(x[i] >= 0)*, cuado los ceros no son ni positivos ni negativos. Una solución a este fallo es: *(x[i] > 0)*, es decir, cualquier número mayor que cero es positivo.

OddOrPos.java: el programa cuenta el número total de elementos de una cadena que son positivos o impares. Viendo el código se observa que sólo tiene en cuenta los números impares positivos: *if x[i]%2 == 1 || x[i] > 0).* Para solucionar el fallo y tener en cuenta también los números negativos es necesarios hacer el módulo del número antes de ver si es impar: *if (abs(x[i])%2 == 1 || x[i] > 0)*

2. Proporciona, si ello es posible, un caso de prueba que no ejecute el código que tiene el fallo. Si no es posible, explica por qué.

FindLast.java: no es posible poner un caso de prueba ya que el fallo está en la conición del for(*i>0;*).

Lastzero.java: no es posible poner un caso de prueba ya que el fallo está en la conición del for(*int i = x.length-1; i >= 0; i*--).

CountPositive.java: un caso de prueba que no ejecute el código sería una cadena vacía ya que en ese caso no se cumple la condición de que i < x.length, por lo tanto no se entraría en el bucle para ejecutar el codigo que tiene fallo(*(x[i] >= 0*).

OddOrPos.java: un caso de prueba que no ejecute el código sería una cadena vacía ya que en ese caso no se cumple la condición de que i < x.length, por lo tanto no se entraría en el bucle para ejecutar el codigo que tiene fallo*(if x[i]%2 == 1 || x[i] > 0)*.

3. Si es posible, proporciona un caso de prueba que ejecuta el fallo que hay en el código, pero que no provoque un error en el estado. Si no se puede, explica por qué.

FindLast.java: cualquier caso de prueba en el que el número buscado no ocupe la posición 0 será un caso que no provoque error en el estado.

Lastzero.java: cualquier caso de prueba que sólo contenga un cero no provocará un error en el estado.No estoy muy seguro. Es posible que cualquier prueba provoque un caso de error debido a que se empieza a buscas el último cero por el principio de la cadena en vez de por el final?.

CountPositive.java: cualquier prueba que no contenga ceros será una prueba que no provocará error en el estado.

OddOrPos.java: cualquier prueba que no tenga números negativos impares será una prueba que no provocará error en el estado.

4. Si es posible, proporciona un caso de prueba que provoque un error en el estado, pero que no acabe provocando una disfunción en el comportamiento del programa. No olvides que el contador de programa forma parte, junto a las variables, del estado del programa. Si no es posible, explica por qué.

FindLast.java: Ejemplo de prueba: [1,2,3,4,5,6] Find(1). Esta prueba provoca un error en el estado y además provoca una disfunción en el comportamiento del programa. Si se tiene un estado de error eso provoca una disfunción en el comportamiento del programa obteniendo resultados erroneos.

Lastzero.java: Ejemplo de prueba: [1,2,3,4,0,1,3].Este ejemplo de prueba tiene un error en el estado debido a que busca el último cero empezando por el principio de la cadena pero no provoca una disfunción en el comportamiento del programa ya que devuelve el valor esperado.

CountPositive.java: No es posible porporcionar un caso de prueba que no provoque disfunción en el comportamiento del programa.En el momento que haya un cero en la cadena, se provoca un estado de error lo que acaba provocando una disfunción en el funcionamiento del programa que cuenta el cero como número positivo.

OddOrPos.java: No es posible porporcionar un caso de prueba que no provoque disfunción en el comportamiento del programa. Si en la cadena hay números negativos impares se provoca un estado de error y además una disfunción en el comportamiento del programa.

5. Para el caso de prueba del anterior apartado, describe el primero de los estados erróneos. Describe detalladamente todo el estado (todas las variables, incluyendo el contador de programa).

Lastzero.java: Ejemplo de prueba: [1,2,3,4,0,1,3].

En el ejemplo de Lastzero tenemos una cadena de 7 dígitos y queremos encontrar la posición del último cero. El contador del programa empieza en i = 0 hasta i = 6. En cada vuelta comprueba si el dígito de la posición i es un cero o no. El error en el estado se produce en el contador del bucle ya que debería empezar en i = 6 y acabar en i=0. El ejemplo propuesto provoca un error pero no una disfunción en el funcionamiento del programa debido a que sólo hay un cero.