EJERCICIOS PRUEBAS TESTS DE SOFTWARE
Daniel Cabezas Recio 03937658N ITT

countPositive

-Código inicial ==>

```
/**
* Counts positive elements in array
*
* @param x array to search
* @return number of positive elements in x
* @throws NullPointerException if x is null
*/
public static int countPositive (int[] x)
{
	int count = 0;

	for (int i=0; i < x.length; i++)
	{
		if (x[i] >= 0)
		{
			count++;
		}
	}
	return count;
}
```

1. En este caso el fallo lo tenemos en el if ya que comprueba que el número sea mayor o igual a cero, de esta forma considera que el cero sería un número positivo. Para que sea correcto habría que comprobar únicamente que el número sea mayor que 0.

2. Para este ejercicio no hay ningún caso en el que no se ejecute el código que tiene el fallo , ya que al introducir cualquier número se producirá el error.

3. Si ponemos un array con diferentes números en el que no haya un cero, sacará lo esperado ya que ejecuta el código sin cambiar el estado. De forma que al no haber ningún cero, no va entrar al if cuando no deba, y no se producirá el error de estado.

4. No sería posible que hubiera un error en el estado sin disfunción, ya que al haber error de estado el contador de números positivos se incrementa dando lugar a la disfunción.

5. No sería posible.

6. Hecho.

lastZero

-Código inicial ==>

```
/**
* Find LAST index of zero
*
* @param x array to search
* @return index of last 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;
}
```

1. En este caso el fallo se encuentra en el bucle for ya que empieza recorriendo el array desde el principio y en cuanto encuentra el primer cero ya se para y sale de la función, luego está contemplando que ha encontrado el primer cero y no el último. Lo que tendríamos que hacer sería recorrer el array desde el final hacia el principio de forma que el primer cero que encuentre sea realmente el último que es lo queremos.

2. Para este ejercicio no hay ningún caso en el que no se ejecute el código que tiene el fallo , ya que al introducir cualquier número se producirá el error.

3. No hay caso posible, se tendría que ejecutar el código y entonces se vería el error.

4. Poniendo un array de números positivos sin cero, tendríamos el error en el estado ya que se recorre el bucle al revés, pero sin disfunción puesto que no hemos puesto en el array ningún cero.

5. El primer estado sería i=0 y esto es un error, puesto que debería empezar por el último elemento del array, con i=x.length-1.

6. Hecho.

findLast

-Código inicial==>

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

1. En este caso el error se encuentra en el bucle for a la hora de poner las condiciones. No se contempla el caso de que i=0, luego no llega al cero. Tendríamos que hacer que el bucle llegara hasta i>=0.

2. Para este ejercicio no hay ningún caso en el que no se ejecute el código que tiene el fallo , ya que al introducir cualquier número se producirá el error.

3. Por ejemplo un array [2,4,5,6]. Si buscamos el número 5 vemos que el índice del último elemento buscado es 2. Sale un resultado correcto a pesar de ejecutar el código con error pero no afecta al estado. Entra en el bucle pero como encuentra el número antes de llega a 0 no sigue.

4. No sería posible que hubiera un error en el estado sin disfunción, el bucle no tiene en cuenta la primera posición de los números introducidos devolviendo -1 porque no comprueba ese número.

5. No sería posible.

6. Hecho.

oddOrPos

-Código inicial==>

```
/**
* Count odd or positive elements in an array
*
* @param x array to search
* @return count of odd or positive elements 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;
}
```

1. En este caso el error se encuentra dentro del bucle for, concretamente en el if. La forma que tiene de encontrar los números impares es bajo la condición x[i]%2==1, pero no sería correcta. Para poder corregir el error deberíamos poner en su lugar la condición x[i]%2.

2. Para este ejercicio no hay ningún caso en el que no se ejecute el código que tiene el fallo , ya que al introducir cualquier número se producirá el error.

3. Los ejemplos serían donde no hay impares negativos.

4. No sería posible que hubiera un error en el estado sin disfunción,ya que habiendo error en el if, el contador no entrará, de forma que no se sumará cuando si tocara, dando lugar a error.

5. No sería posible.

6. Hecho.