Ejecricios JUnit Luis Álvarez Jerónimo

CountPositive

1) En el código se indica que el cero es positivo, cuando en verdad no debería contarse como tal, la modificación debería ser esta:

x[i] >= 0 ----------> x[i] > 0

2)
```
@Test public void arraySinCeros()
    {
      int arr[] = {-1, 3, 1, 3};
      assertEquals("El Array no tiene ceros", 3, CountPositive.CountPositive(arr));
    }
```

Se espera que se obtengan 3 números positivos y se obtienen

3) No se puede porque cada vez que haya un cero a provocar un error de estado

4)
```
@Test public void arrayConCeros()
    {
      int arr[] = {-1, 3, 1, 0};
      assertEquals("El Array contiene ceros", 2, CountPositive.CountPositive(arr));
    }
```

Se espera que haya 2 y sin embargo cuenta el cero como positivo

5) Cuenta 3, 1 y 0 como positivos entonces no se obtiene el resultado esperado

6) Se ejecuta perfectamente sustituyendo x[i] >= 0 ----------> x[i] > 0

LastZero

1) El error está en que detecta el primer cero y no el último, el cambio sería utilizar una variable donde asignar la última posición donde se ha encontrado un cero, para así al salir del bucle tener única y exclusivamente la última posición

2)
```
@Test public void noZero()
  {
    int arr[] = {1, 1, 3};
    assertEquals("No Zero", 0, LastZero.LastZero(arr));
  }
```

Al no haber cero, no ejecuta el error de salir a la primera

3)
```
@Test public void oneZero()
  {
    int arr[] = {0, 1, 3};
    assertEquals("One Zero", 0, LastZero.LastZero(arr));
  }
```

En este caso, el error se producirá porque saldrá siempre al primer cero que encuentre pero al haber solo uno no creará error de estado

4)
```
@Test public void multipleZeroes()
  {
    int arr[] = {0, 1, 0};
    assertEquals("Multiple zeroes", 2, LastZero.LastZero(arr));
  }
```

5)Aquí debería decir que el último cero se encuentra en la posición 2 pero al salir del bucle al primer cero que encuentra, el programa retornará la posición 0 como ultimo cero encontrado

6) Funciona al realizar los cambios

FindLast

1) El error está en el bucle, ya que nunca mira la primera posición por el i>0, simplemente hay que sustituirlo por i>=0
2)
```
@Test public void lastOccurrence()
  {
    int arr[] = {3, 2, 5};
    int y = 2;
    assertEquals("Last occurence ", 1, FindLast.FindLast(arr, y));
  }
```
3)
```
@Test public void noOccurrence()
  {
    int arr[] = {3, 4, 5};
    int y = 2;
    assertEquals("Last occurence ", -1, FindLast.FindLast(arr, y));
  }
```
Al no haber 2, tampoco mirará la primera posición pero no dará un error de estado

4)
```
@Test public void lastOccurrenceInFirstPosition()
  {
    int arr[] = {2 3, 5};
    int y = 2;
    assertEquals("Last occurence in the first position ", 0, FindLast.FindLast(arr, y));
  }
```
5) Ejecutará el fallo y además no detectará el 2 en la primera posición por lo que en el momento en el que pase del 3, saldrá del bucle y, por lo tanto la posición será -1, es decir, que no hay un 2 en la lista
6) Si realizas los cambios designados en el primer apartado, todos los tests funcionan como deben

OddOrPositive

1)El error se encuentra al no contemplar la existencia de números negativos impares, se soluciona utilizando el valor absoluto del resto de la división entre 2 para realizar la comparación
2)
```
@Test public void Numbers()
  {
    int arr[] = {3, 2, 0, 1, 4};
    assertEquals("Positive numbers in array", 4, OddOrPositive.oddOrPos(arr));
  }
```
Si no utilizas números negativos impares no ejecuta el error
3)
```
@Test public void negativeNumbers()
  {
    int arr[] = {-4, -2, 0, 1, 4};
    assertEquals("Negative numbers in array", 2, OddOrPositive.oddOrPos(arr));
  }
```
Ejecuta el error porque no está comparando bien los restos de los números negativos, pero al no ser ninguno impar, no detecta este fallo en un error de estado

4)@Test public void negativeOddNumbers()
  {
    int arr[] = {-3, -2, 0, 1, 4};
    assertEquals("Negative odd numbers in array", 3, OddOrPositive.oddOrPos(arr));
  }
5) Al haber un número negativo impar, cuando llega a este número calcula su resto, como es distinto de 1 (es decir, -1) no lo detecta como número impar, siendo este resultado erróneo
6) Si corriges utilizando el valor absoluto del resto, todos los tests anteriores funcionan