Encuentra la última posición de equilibrio

Implementa una función que resuelva el siguiente problema:

```
function numUnos(s : seq<int>) : int
ensures s == [] ==> numUnos(s) == 0
ensures s != [] && s[0] == 1 ==> numUnos(s) == 1 + numUnos(s[1..])
ensures s != [] && s[0] != 1 ==> numUnos(s) == numUnos(s[1..])

function numCeros(s : seq<int>) : int
ensures s == [] ==> numCeros(s) == 0
ensures s != [] && s[0] == 0 ==> numCeros(s) == 1 + numCeros(s[1..])
ensures s != [] && s[0] != 0 ==> numCeros(s) == numCeros(s[1..])

method xxx (v : array<int>) returns (p : int)
requires v!= null
ensures -1 <= p < v.Length
ensures numUnos(v[..p+1]) == numCeros(v[..p+1])
ensures forall k :: p < k < v.Length ==> numUnos(v[..k+1]) != numCeros(v[..k+1])
```

Requisitos de implementación.

Indicar el coste de la solución obtenida.

La función que resuelve el problema debe recibir los datos en un vector y devolver el valor de la posición (observad que puede ser -1).

Entrada

La entrada comienza con el número de casos de prueba. Cada caso de prueba tiene dos líneas. En la primera línea se indica el tamaño del vector, en la segunda los valores separados por blancos.

Salida

Para cada caso de prueba se escribe en una línea el valor de la posición pedida. Si el tamaño del vector es cero la posición debe ser -1.

Entrada de ejemplo

```
5
8
2 1 1 0 2 3 0 2
6
0 1 0 1 1 1
5
1 1 1 1 1
4
2 3 2 3
1
0
```

Salida de ejemplo

```
7
3
-1
3
-1
```

Autor: Isabel Pita.