Recall: Horizontal Notation

Example of an assertion about an sequence \( b \). It asserts that:
1. \( b[0..k-1] \) is sorted (i.e. its values are in ascending order)
2. Everything in \( b[0..k-1] \) is ≤ everything in \( b[k]. \)

Given index \( h \) of the first element of a segment and index \( k \) of the element that follows that segment, the number of values in the segment is \( k - h \).

\[ b[h..k-1] \text{ has } k - h \text{ elements in it.} \]

Partition Algorithm Implementation

```python
def partition(b, h, k):
    
    # Moves to end of block.
    _swap(b, h, t)
    j = h

    # Sorts b[h..t] block around a pivot \( b[h] \),
    # only rearrange part of the list, not all
    while i < j:
        if b[i] < x:
            i += 1
            # post: \( (\) \)
        else:
            j -= 1
            # post: \( (\) \)

    return i
```

Dutch National Flag Algorithm

```python
def dutch(b, h, k):
    
    # Move partition points as a tuple (i,j,k)
    t = h
    while t < i:
        if b[t] < 0:
            _swap(b, t, j)
            t += 1
        elif b[t] == 0:
            i += 1
            _swap(b, j, j + 1)
        else:
            _swap(b, i, t)
            i += 1
            j = j + 1
    return (0, j)
```

Dutch National Flag Variant

- Sequence of integer values
  - 'red' = negatives, 'white' = 0, 'blues' = positive
  - Only rearrange part of the list, not all
Flag of Mauritius

• Now we have four colors!
  * Negatives: ‘red’ = odd, ‘purple’ = even
  * Positives: ‘yellow’ = odd, ‘green’ = even

### Linear Search

```python
def linear_search(b, c, h):
    # pre: b
    # invariant: c is not in b[0..i-1]
    while i < len(b) and b[i] == c:
        i = i + 1
    # post: e is not in b[0..i-1]
    # l = i - len(b) or b[i] == c
    return i if i < len(b) else -1
```

### Binary Search

• Vague: Look for v in sorted sequence segment b[h..k].

• Better:
  * Precondition: b[h..k-1] is sorted (in ascending order).
  * Postcondition: b[h..i] <= v and v < b[i+1..k-1]

• Below, the array is in non-descending order:

```
def binary_search(b, c, h):
    # pre: b
    # post: b
    # invariant: c is not in b[h..k]
    # l = i - len(b) or b[i] == c
    return i if i < len(b) else -1
```