Developing Loops from Invariants

Developing a Loop on a Range of Integers

- Given a range of integers a..b to process.
- Possible alternatives
  - Could use a for-loop: `for x in range(a,b+1):`
  - Or could use a while-loop: `x = a; while x <= b:`
  - Which one you can use will be specified
- But does not remove the need for invariants
  - **Invariants**: assertion supposed to be true before and after each iteration of the loop

Developing an Integer Loop (a)

Suppose you are trying to implement the command

Process a..b

Write the command as a postcondition:

post: a..b has been processed.

Developing an Integer Loop (b)

Set-up using while:

```
while k <= b:
    # Process k
    k = k + 1
# post: a..b has been processed.
```

Developing an Integer Loop (c)

Add the invariant:

```
# invariant: a..k-1 has been processed
while k <= b:
    # Process k
    k = k + 1
# post: a..b has been processed.
```

Note it is post condition with the loop variable

Developing an Integer Loop (d)

Fix the initialization:

Has to handle the loop variable (and others)

```
init to make invariant true
# invariant: a..k-1 has been processed
while k <= b:
    # Process k
    k = k + 1
# post: a..b has been processed.
```

Developing an Integer Loop (e)

Figure out how to "Process k":

```
init to make invariant true
# invariant: a..k-1 has been processed
while k <= b:
    # Process k
    implementation of "Process k"
    k = k + 1
# post: a..b has been processed.
```

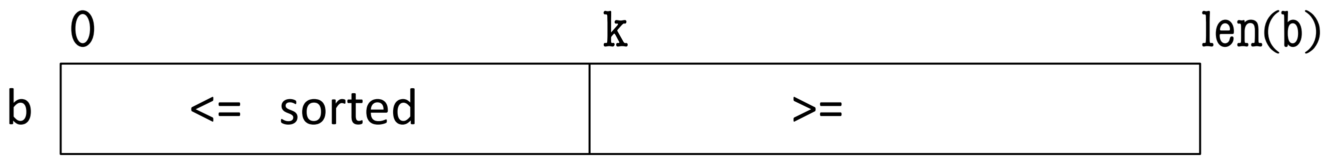
Range

- Pay attention to range:

  a..b or a+1..b or a…b-1 or …

- This affects the loop condition!
  - Range a..b-1, has condition k < b
  - Range a..b, has condition k <= b
- Note that a..a-1 denotes an empty range
  - There are no values in it
- a..b how many elements? b – a + 1

Horizontal Notation for Sequences

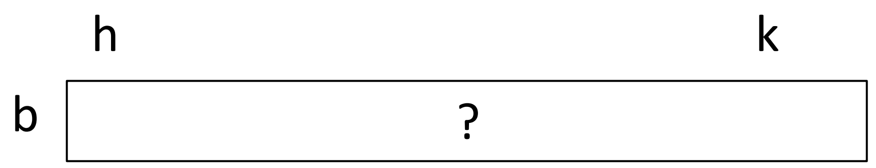


Example of an assertion about an sequence b. It asserts that:

1. b[0..k–1] is sorted (i.e. its values are in ascending order)
2. Everything in b[0..k–1] is $\leq$ everything in b[k..len(b)–1]

Algorithm Inputs

- We may specify that the list in the algorithm is
  - b[0..len(b)-1] or
  - a segment b[h..k] or
  - a segment b[m..n-1]
- **Work with whatever is given!**



- Remember formula for # of values in an array segment
  - Following – First
  - e.g. the number of values in b[h..k] is k+1–h.

Example Question, Fall 2013 Final

	0	k
pre: b	sorted	

	0	h	k
post: b	Unchanged, values in b[h+1..k]	b[0..k] w/o duplicates	

	0	p	h	k
inv: b	???	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates	

- **Example**:
  - Input [1, 2, 2, 2, 4, 4, 4]
  - Output [1, 2, 2, 2, 1, 2, 4]

Solution to Fall 2013 Final

	0 … p	… h	… k
inv: b	unchanged	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates

```
# Assume 0 <= k, so the list segment has at least one element
p =
h =
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while          :
```

Solution to Fall 2013 Final

	0	p	h	k
inv: b	unchanged	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates	

```
# Assume 0 <= k, so the list segment has at least one element
p = k-1
h = k-1
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while          :
```

Solution to Fall 2013 Final

	0	p	h	k
inv: b	unchanged	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates	

```
# Assume 0 <= k, so the list segment has at least one element
p = k-1
h = k-1
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while 0 <= p:
```

Solution to Fall 2013 Final

	0	p	h	k
inv: b	unchanged		Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates

```
# Assume 0 <= k, so the list segment has at least one element
p = k-1
h = k-1
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while 0 <= p:
    if b[p] != b[p+1]:
        b[h] = b[p]
        h = h-1
    p = p-1
```

DOs and DON'Ts #1

- DO use variables given in the invariant.
- DON'T use other variables.

```
# invariant: b[h..] contains the sum of c[h..] and d[k..],
# except that the carry into position k-1 is in 'carry'
while ____________ :
    # Okay to use b, c, d, h, k, and carry
    # Anything else should be 'local' to while
```

DOs and DON'Ts #2

DO double check corner cases!

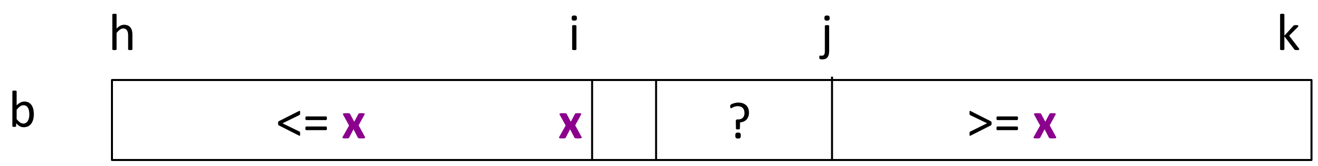
- h = len(c)
- while h > 0:
  - What will happen when h=1 and h=len(c)?
  - If you use h in c (e.g. c[h]) can you possibly get an error?

```
# invariant: b[h..] contains the sum of c[h..] and d[k..],
# except that the carry into position k-1 is in 'carry'
while h > 0:
    ...
```

Range is off by 1.
How do you know?

DOs and DON'Ts #3

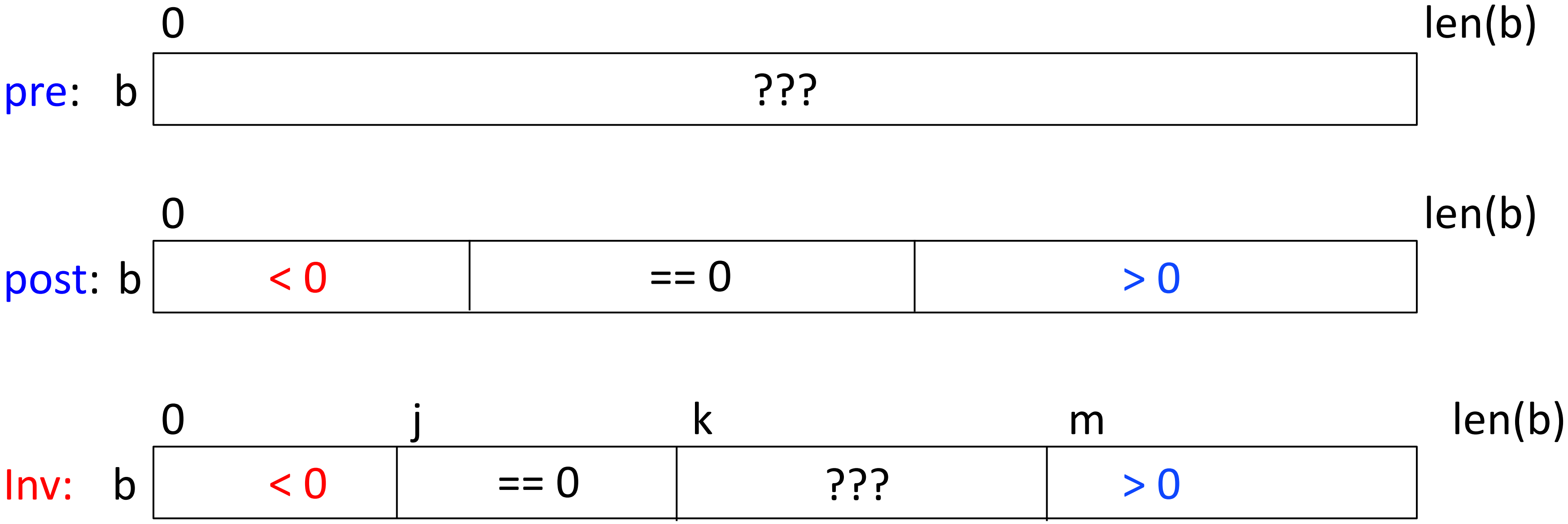
- DON'T put variables directly above vertical line.



  - Where is j?
  - Is it unknown or >= x?

Dutch National Flag

- Sequence of 0..n-1 of red, white, blue colors
  Arrange to put reds first, then whites, then blues
- Input is the list b of integers
- Modifies the list according to the invariant.



Dutch National Flag

Inv: b

0	j	k	m	len(b)
< 0	== 0	???	> 0	

```
def dutch_national_flag(b):
    j = 0; k = 0; m = len(b)
    while k < m:
        if b[k] == 0:
            k = k + 1
        elif b[k] > 0:
            _swap(b, k, m-1)
            m = m – 1
        else:   # b[k] < 0
            _swap(b, k, j)
            k = k + 1
            j = j + 1
```

Dutch National Flag

	0	j	k	m	len(b)
Inv: b	< 0	== 0	???	> 0	

```
def dutch_national_flag(b):
    j = 0; k = 0; m = len(b)
    while k < m:
        if b[k] == 0:
            k = k + 1
        elif b[k] > 0:
            _swap(b, k, m-1)
            m = m − 1
        else:   # b[k] < 0
            _swap(b, k, j)
            k = k + 1
            j = j + 1
```

dutch_national_flag([3,-1,5,-2,0])

k, j					m
3	-1	5	-2	0	

k, j				m
0	-1	5	-2	3

j	k			m
0	-1	5	-2	3

	j	k		m
-1	0	5	-2	3

	j	k	m	
-1	0	-2	5	3

		j	k, m	
-1	-2	0	5	3

Questions?