Terminology: Range Notation

- **m..n** is a range containing **n+1** values
  - **2..5** contains 2, 3, 4, 5. Contains 5+1 = 4 values
  - **2..4** contains 2, 3, 4. Contains 4+1 = 3 values
  - **2..3** contains 2, 3. Contains 3+1 = 2 values
  - **2..2** contains 2. Contains 2+1 = 1 values
  - **2..1** contains **???

What does 2..1 contain?

A: nothing
B: 2, 1
C: 1
D: 2
E: something else

Assertions: Tracking Code State

- **assertion**: true-false statement placed in a program to assert that it is true at that point
  - Can either be a comment, or an assert command
- **invariant**: assertion supposed to "always" be true
  - If temporarily invalidated, must make it true again
  - Example: class invariants and class methods
- **loop invariant**: assertion supposed to be true before and after each iteration of the loop
- **iteration of a loop**: one execution of its body

Assertions versus Asserts

- **Assertions prevent bugs**
  - Help you keep track of what you are doing
  - Also **track down bugs**
    - Make it easier to check belief/code mismatches
  - The assert statement is a (type of) assertion
  - One you are enforcing
  - Cannot always convert a comment to an assert

Preconditions & Postconditions

- **Precondition**: assertion placed before a segment
- **Postcondition**: assertion placed after a segment

Solving a Problem

- **precondition**: # x = sum of 1..n
  - x = x + n
  - n = n + 1

- **postcondition**: # x = 0

| A: x = x + 1 |
| B: x = x + n |
| C: x = x + n+1 |
| D: None of the above |
| E: I don’t know |

What statement do you put here to make the postcondition true?

Invariants: Assertions That Do Not Change

- **Loop Invariant**: an assertion that is true before and after each iteration (execution of repetend)
  - x = 0; i <= 5
  - while i <= 5:
    - x = x + i+1
    - i = i + 1
  - # x = sum of squares of 2..5

Invariant:

- x = sum of squares of 2..i-1
  - in terms of the range of integers that have been processed so far

- # invariant

- i = 2
- i = 3
- i = 4
- i = 5

The loop processes the range 2..5

Relationship Between Two
If precondition is true, then postcondition will be true
Invariants: Assertions That Do Not Change

\( x = 0; \ i = 2 \)

# Inv: \( x = \text{sum of squares of } 2..i \)
while \( i <= 5 \):
  \( x = x + i^2 \)
  \( i = i + 1 \)

# Post: \( x = \text{sum of squares of } 2..5 \)

Integers that have been processed: \( 2, 3, 4, 5 \)
Range \( 2..i-1 \): \( 2 \)

The loop processes the range \( 2..5 \)

Invariant was always true just before test of loop condition. So it’s true when loop terminates

Designing Integer while-loops

# Process integers in \( a.b \)
# Inv: integers in \( a..k-1 \) have been processed
\( k = a \)
while \( k <= b \):
  process integer \( k \)
  \( k = k + 1 \)

# post: integers in \( a..b \) have been processed

Finding an Invariant

# Make \( b \) True if \( n \) is prime, False otherwise
\( b = \text{True} \)
\( k = 2 \)

# invariant: \( b \) is True if no int in \( 2..k-1 \) divides \( n \), False otherwise
while \( k < n \):
  # Process \( k \);
  if \( n \% k == 0 \):
    \( b = \text{False} \)
  \( k = k + 1 \)

# \( b \) is True if no int in \( 2..n-1 \) divides \( n \), False otherwise

What is the invariant?

A: \( k = 0; \ c = s[0] \)
B: \( k = 1; \ c = s[0] \)
C: \( k = 1; \ c = s[1] \)
D: \( k = 0; \ c = s[1] \)
E: None of the above

Finding an Invariant

# set \( x \) to \# adjacent equal pairs in \( s \)
\( x = 0 \)

# Inv: \( x = \# \text{adjacent equal pairs in } s[0..k-1] \)
while \( k < \text{len}(s) \):
  # Process \( k \)
  \( k = k + 1 \)

# Postcondition: \( \# \text{adjacent equal pairs in } s[0..\text{len}(s)-1] \)

What is initialization for \( k? \)

A: \( k = 0 \)
B: \( k = 1 \)
C: \( k = -1 \)
D: I don’t know

Reason carefully about initialization

# \( s \) is a string; \( \text{len}(s) \) >= 1
# Set \( c \) to largest element in \( s \)
\( c = \text{??} \)

# Inv: \( c \) is largest element in \( s[0..k-1] \)
while \( k < \text{len}(s) \):
  # Process \( k \)
  \( k = k + 1 \)

# \( c \) is largest char in \( s[0..\text{len}(s)-1] \)
Equivalent postcondition

What is the invariant?

1. What is the invariant?
2. How do we initialize \( c \) and \( k? \)