Testing change to a linked list

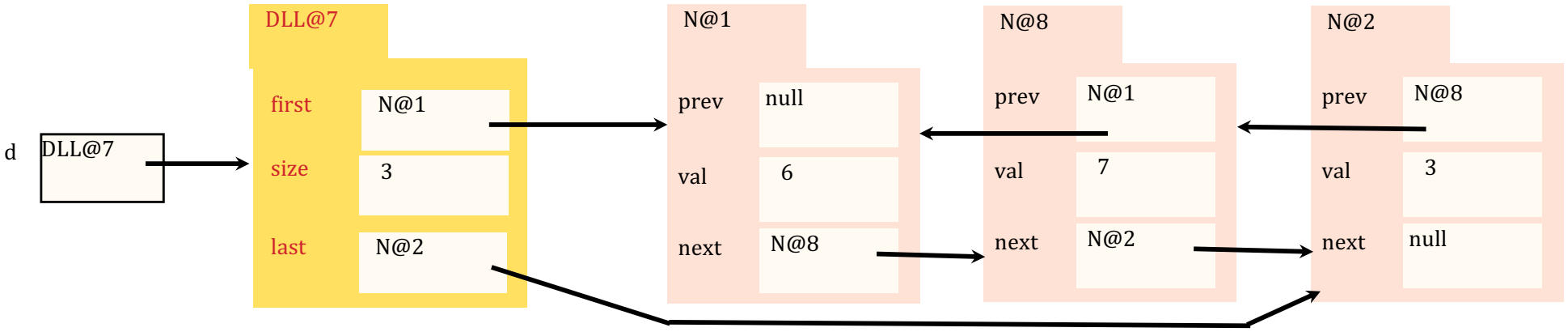
```
d.prepend(5);
assertEquals("[…]",  d.toString());  // uses first, all next fields, val fields
assertEquals("[…]",  d.gnirtSot());  // uses last, all prev fields, val fields
assertEquals(4,          d.size());  // uses field size
```

Thus, those three assertEquals calls test all fields.

You don't have to worry about how to test, TEST ALL FIELDS!

You must test all methods that change the list in this fashion.

Otherwise, points deducted



Iterative palindrome testing

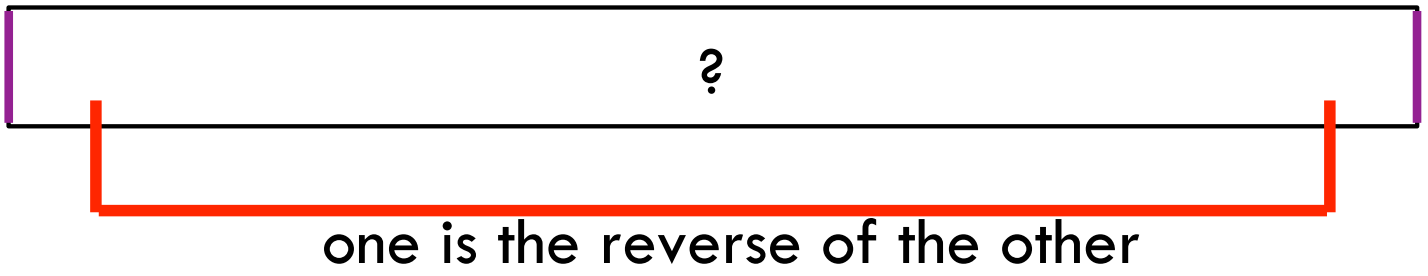
```
/** = " s is a palindrome" */
public static boolean isPal(String s) {
    if (s.length() <= 1) return true;
    // { s has at least 2 chars }
    int n= s.length()-1;
    return s.charAt(0) == s.charAt(n)  &&  isPal(s.substring(1,n));
}
```

```
/** = " b is a palindrome" */
public static boolean isPal(int[] b)
```

We demo the use of loop invariants to **develop** an iterative version of palindrome testing. We use an array instead of a String because the notation is easier in Java. That's the only reason.

Iterative palindrome testing

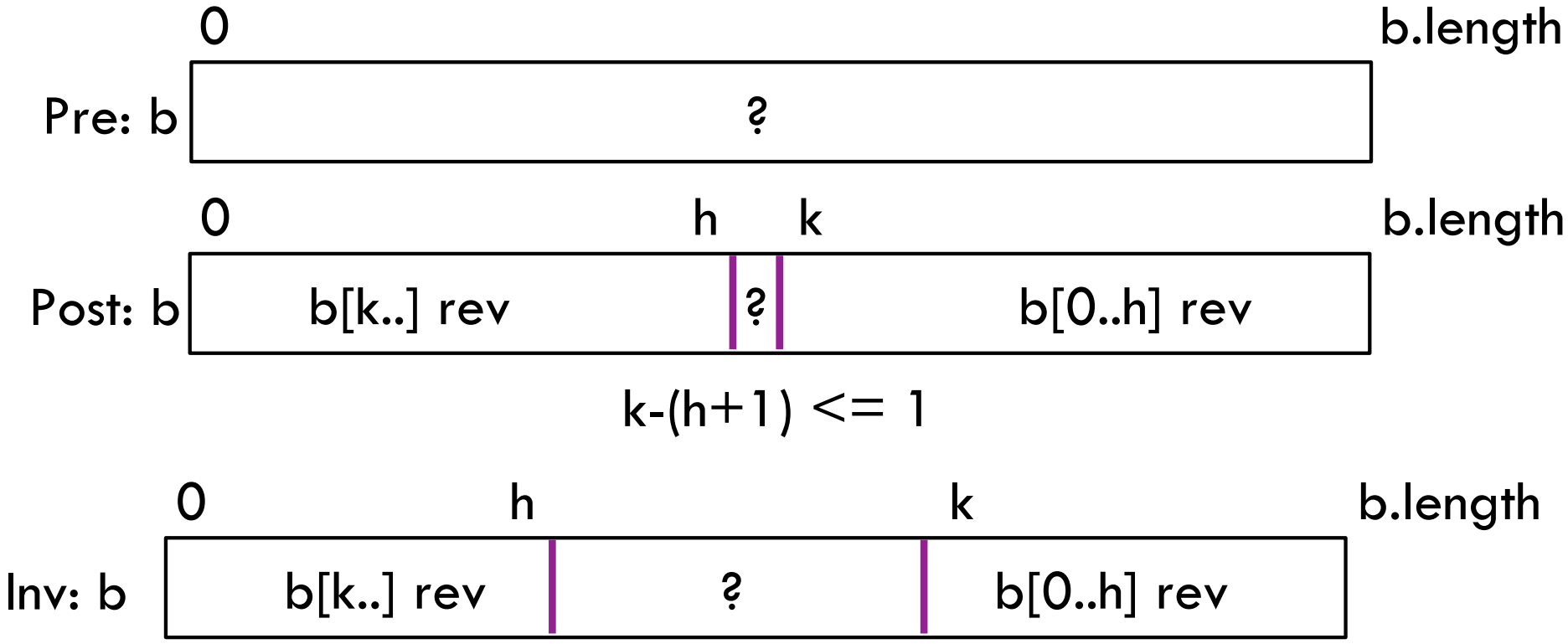
```
/** = " s is a palindrome" */
public static boolean isPal(String s) {
    if (s.length() <= 1) return true;
    // { s has at least 2 chars }
    int n= s.length()-1;
    return s.charAt(0) == s.charAt(n)  &&  isPal(s.substring(1,n));
}
```



Recursive isPal compares successive outer pairs.

Iterative palindrome testing

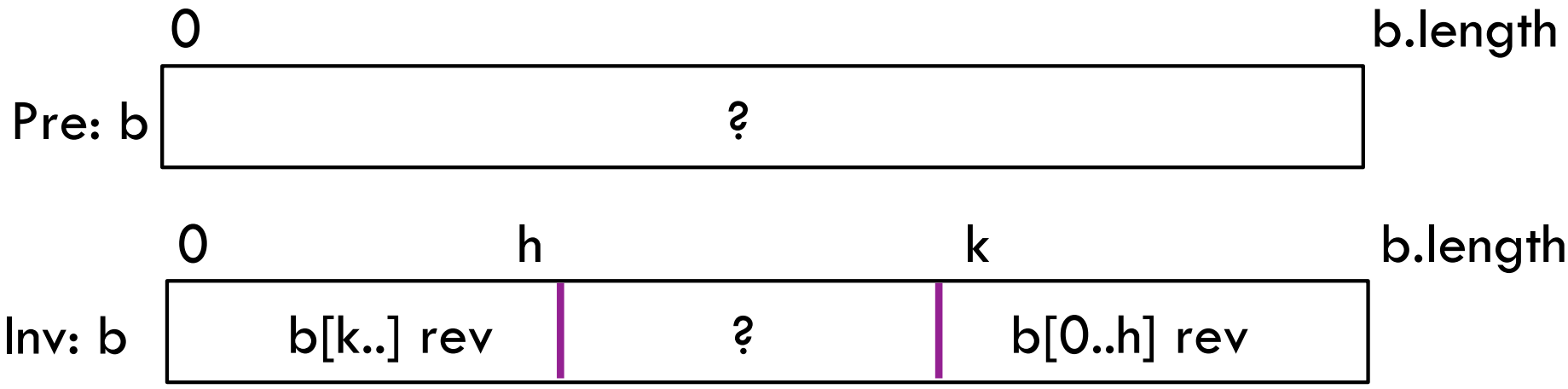
```
/** = " b is a palindrome" */
public static boolean isPal(char[] b) {


}
```



Iterative palindrome testing

```
/** = " b is a palindrome" */
public static boolean isPal(char[] b) {
        int h= -1;  int k= b.length;
}
```

Start? Make invariant true?



Iterative palindrome testing

```
/** = " b is a palindrome" */
public static boolean isPal(char[] b) {
    int h= -1;  int k= b.length;
    while (      h < k-2        ) {
    }
}
```

When k-(h+1) <= 1

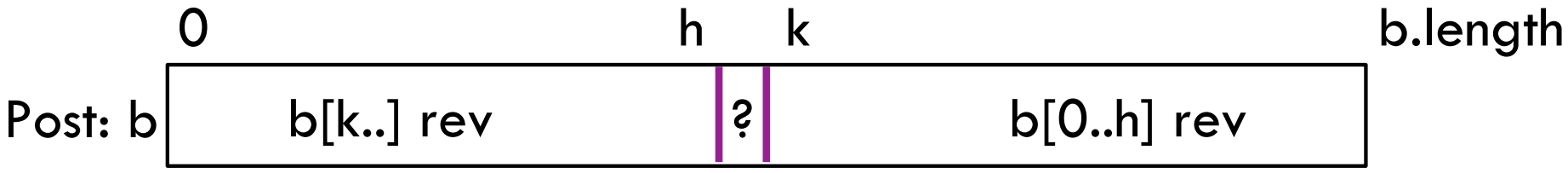
Continue when k-(h+1) > 1

Continue when k-h-1 > 1

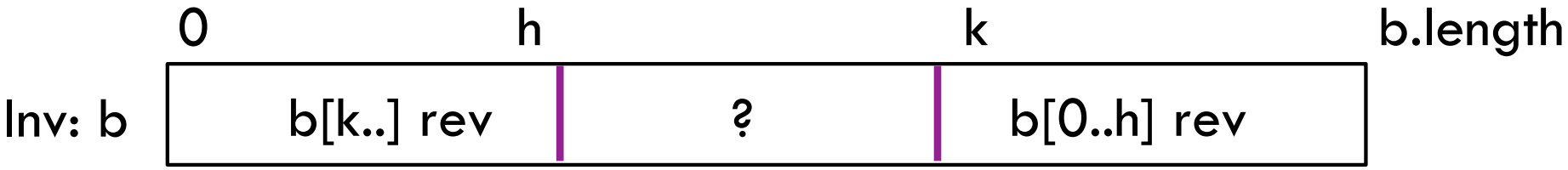
Continue when k-2 > h

Continue when h < k-2

Stop? When is postcondition true?



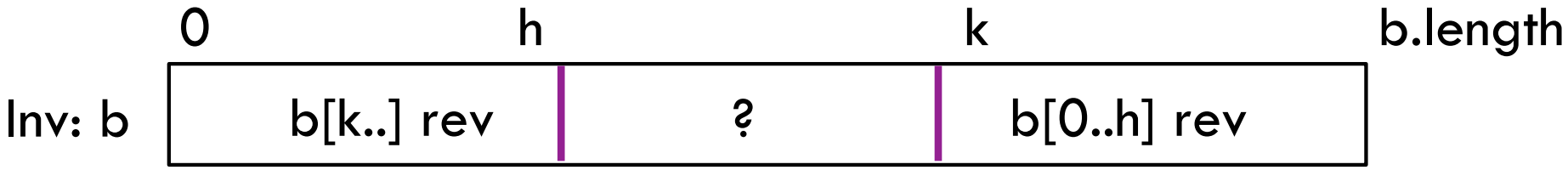
k-(h+1) <= 1



Iterative palindrome testing

```
/** = " b is a palindrome" */
public static boolean isPal(char[] b) {
    int h= -1;  int k= b.length;
    while ( h < k-2 ) {
        h= h+1; k= k-1;
    }
}
```

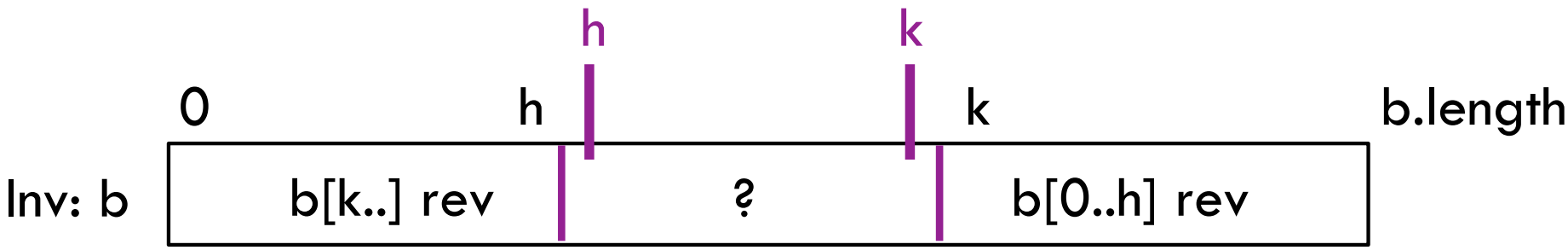
How to make progress toward termination?



Iterative palindrome testing

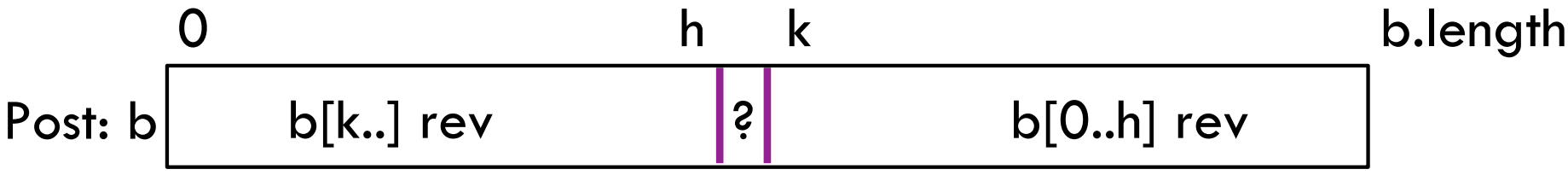
```
/** = " b is a palindrome" */
public static boolean isPal(char[] b) {
    int h= -1;  int k= b.length;
    while ( h < k-2 ) {
        h= h+1; k= k-1;
        if (b[h] != b[k]) return false;
    }
    return true;
}
```

How to make to make invariant true again?

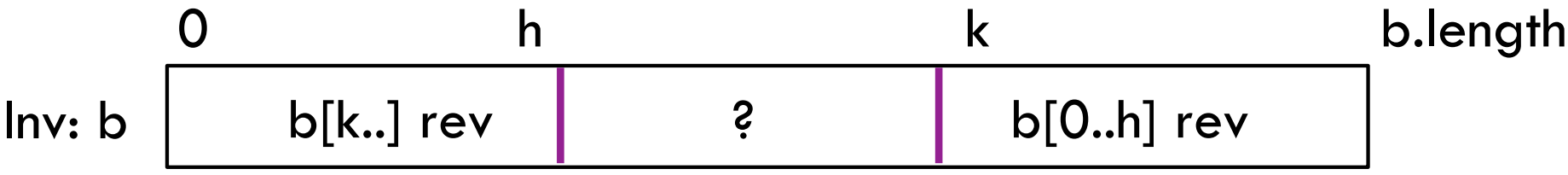


Iterative palindrome testing

```
// Store "array b is a palindrome" in variable isPal
int h= -1;  int k= b.length; isPal= true;
while ( isPal  &&  h < k-2 ) {
      h= h+1; k= k-1;
      if (b[h] != b[k]) isPal= false;
}
```



k-(h+1) <= 1 && isPal = (b[0..h] is reverse of b[k..])



isPal = (b[0..h] is reverse of b[k..])