# CSE 331
# Software Design & Implementation

Winter 2026

Section 10 – Final Review

# Administrivia

**Final**

- **Tuesday, 3/17, Bagley 131/154 (room assignments posted) from 12:30 - 2:20**
- Please arrive 10 minutes early
- Bring your id
- No notecards, all needed definitions will be included

# Course Evals!!

- Please fill them out!

- We appreciate the feedback
  - We do actually read them, so any suggestions will be considered!
  - Everyone should have received an email with the links


# List of General Final Topics

- Proof by Calculation
- Structural Induction
- Testing
- **Writing or reasoning on methods of a mutable/immutable ADTs**
- Reasoning about Loops
- Writing the code of a loop, given the loop idea, invariant, or spec
- Small questions on any other topics (all content is fair game)
  - Strength, Design Patterns, Polymorphism, etc…

- **Practice exams are on the course website under Course Info (Syllabus)>>Exam Mechanics**

# ADT

- **`MutableIntCursor` ADT** represents a list of integers with the ability to insert new characters at the "cursor index" within the list.
  - cursor index can be moved forward or backward

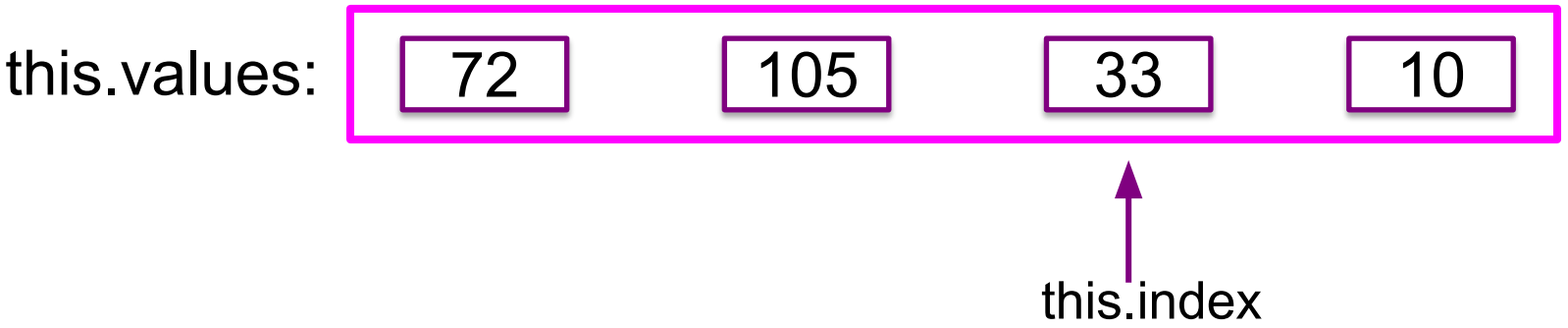
- **`LineCountingCursor`** implements `MutableIntCursor` by:
  - using the abstract state (an index and a list of values) as its concrete state
  - + records the number of newline characters (so class can easily, quickly determine the number of lines in the text)

- **Reminder**: familiar functions and implementations on last few pages of WS!

# ADT Comprehension Cursor (English)

We can think of the cursor index as representing the number of elements behind the cursor.

A cursor at index *i* is “ahead” of the first *i* elements in the array




If This.index = 2, then there are two elements behind the cursor. We can imagine it pointing to the 3rd element in the array

# ADT Comprehension Cursor

Let’s take a second to understand the ADT…

Imagine we have a LineCountingCursor, ourLCC, which is (1, [72, 105, 33, 10]).

Where is the cursor in [72, 105, 33, 10]?

Note: 72 -> H, 105 -> i, 33 -> !, 10 -> \n so this.values is saying “Hi!”

# ADT Comprehension Cursor

Let's take a second to understand the ADT…

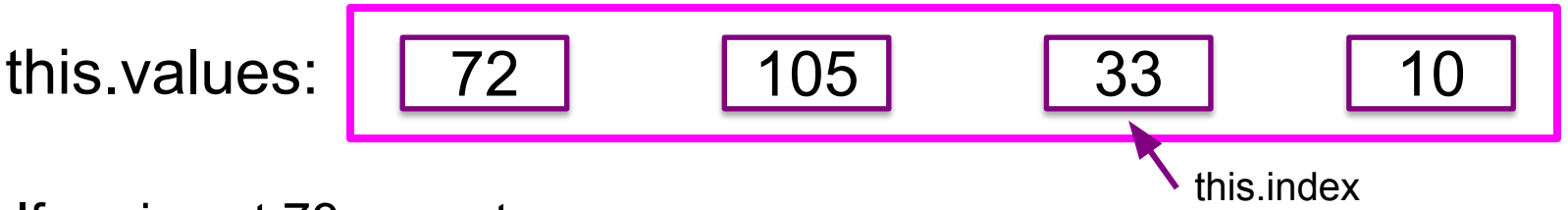
Imagine we have a LineCountingCursor, ourLCC,
which is (1, [72, 105, 33, 10]).

Where is the cursor in [72, 105, 33, 10]?

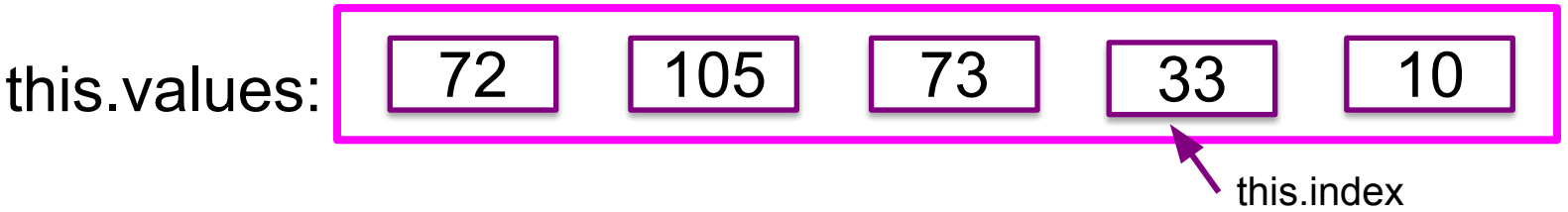
# ADT Insert Comprehension (English)

When we insert at the cursor index, we will put the new char code at that index.

Like when you type, the cursor will then move to the next index and all text after the cursor will shift to the right by one index in the array.




If we insert 73 we get:



# ADT Comprehension Insert Method

What would happen if we called ourLCC.insert(104)?

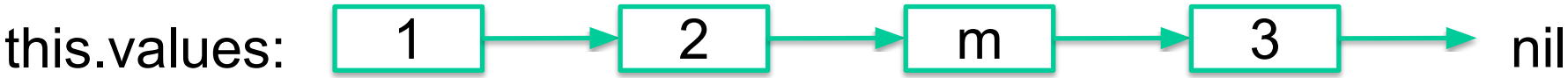
(1, [72, 105, 33])
[72, 105, 33] → (2, [72, 104, 105, 33])
[72, 104, 105, 33] This.values is now saying “Hhi!”

Looking at the effects tag and our AF, since we know obj0 = (1, [72, 105, 33])
Then obj = (1+1, concat([72], concat([104], [105, 33]))
→ obj = (2, [72, 104, 105, 33])

Our RI still holds because 0 <= 1 <= 3 → 0 <= 2 <= 4

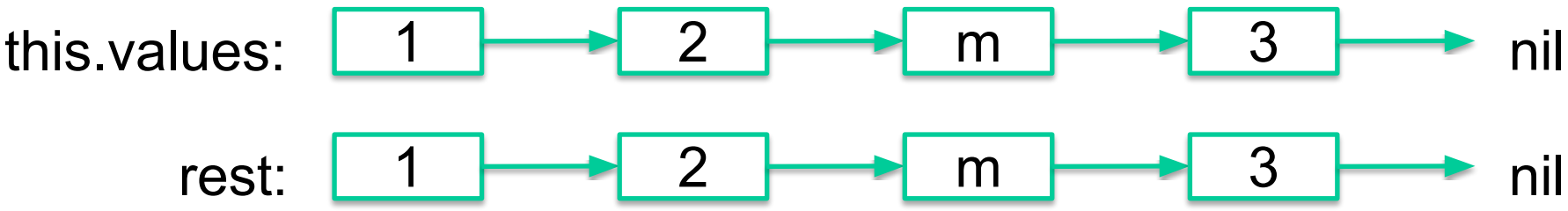
# Problem 2

```
// Inv: this.values = concat(rev(skipped), rest) and
//      contains(m, skipped) = false
```



# Problem 2

```
// Inv: this.values = concat(rev(skipped), rest) and
//      contains(m, skipped) = false
```

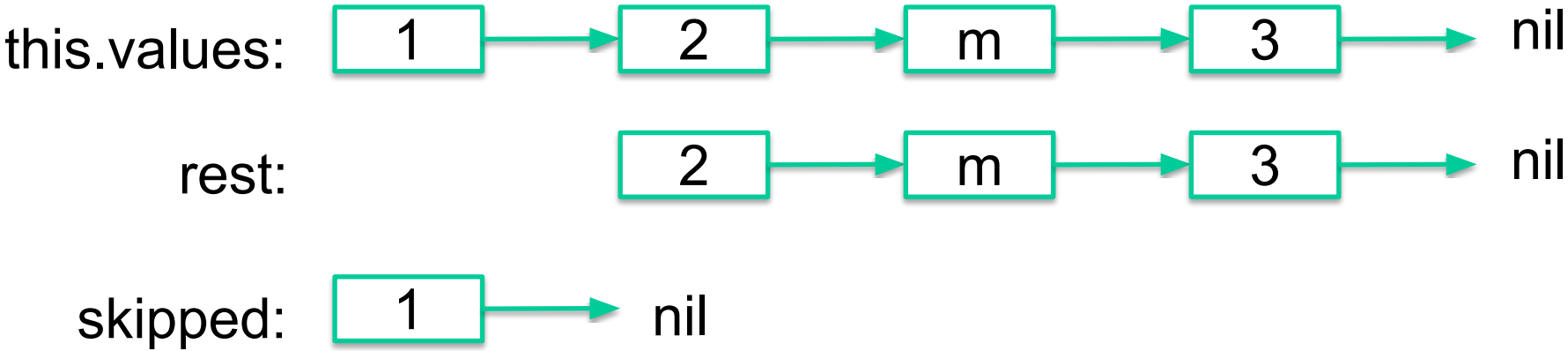



skipped: nil

Easiest way to satisfy the invariant

# Problem 2

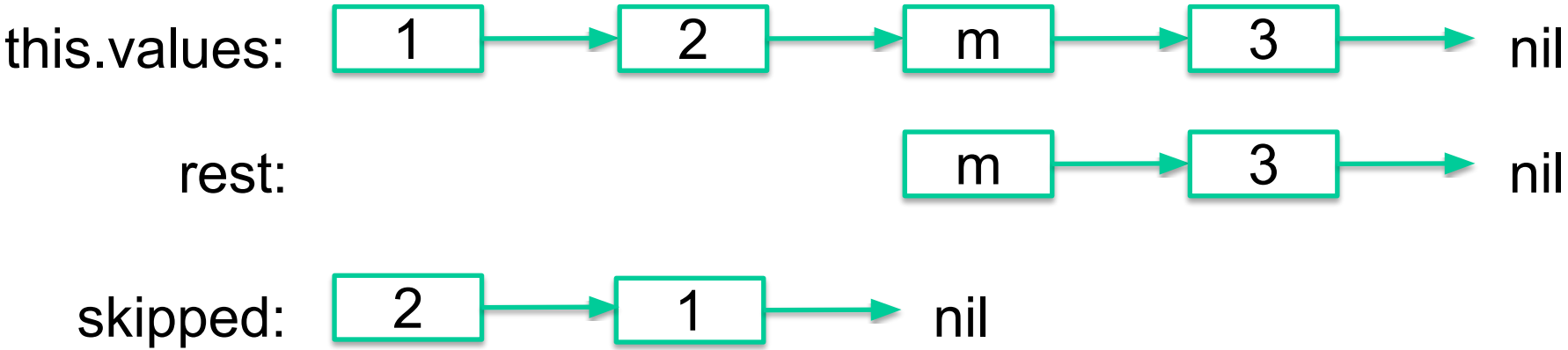
```
// Inv: this.values = concat(rev(skipped), rest) and
//      contains(m, skipped) = false
```




While rest.hd != m (need to check rest != nil first),
remove and append rest.hd to skipped
(cons adds to front which reverses the list which matches the invariant)

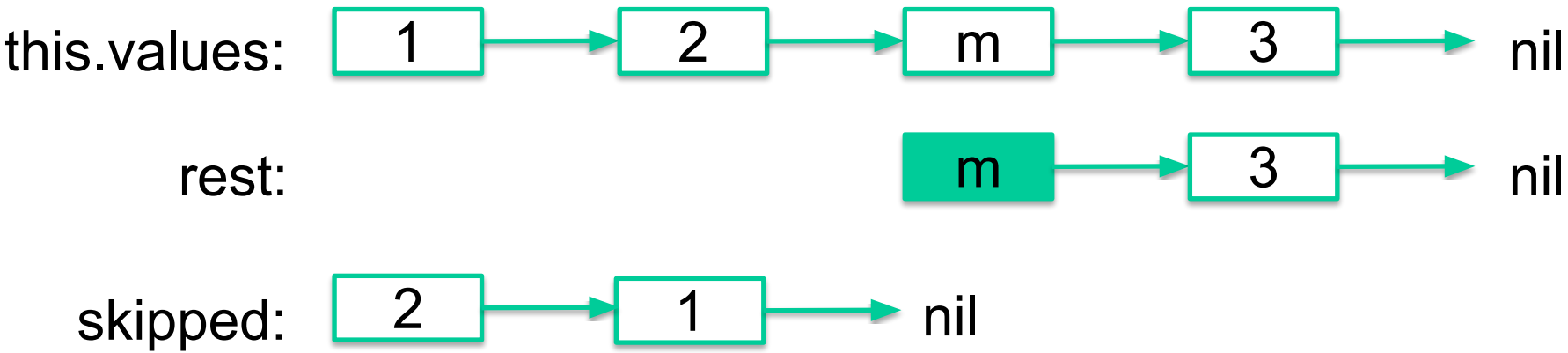
# Problem 2

```
// Inv: this.values = concat(rev(skipped), rest) and
//      contains(m, skipped) = false
```



# Problem 2

```
// Inv: this.values = concat(rev(skipped), rest) and
//      contains(m, skipped) = false
```

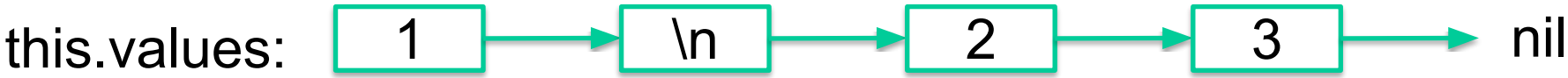



When we exit the loop

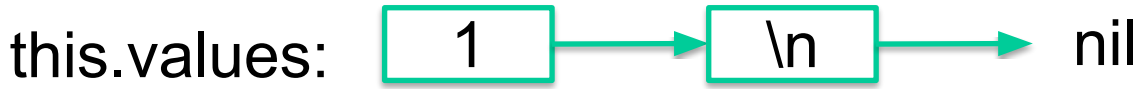
- If rest = nil then we didn't find m
- Otherwise, Index of m is the length of the skipped list

# removeNextLine Example

Imagine we have the list:

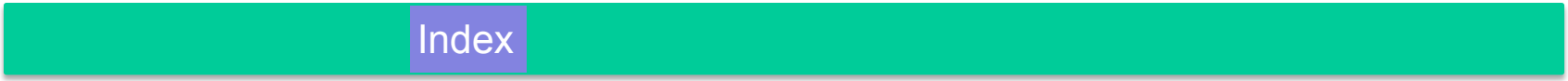



Assume our cursor is at index = 0, what would this.values be after the call LCC.removeNextLine()?



# Problem 3

```
// Removes the line of text after the one containing the cursor index
void removeNextLine() {
```



# Problem 3

```
// Removes the line of text after the one containing the cursor index
void removeNextLine() {
```

# Problem 3 Cases

Now that we see how our values list looks after we split it.

How many different cases do we have?

2 Cases (each time we split):

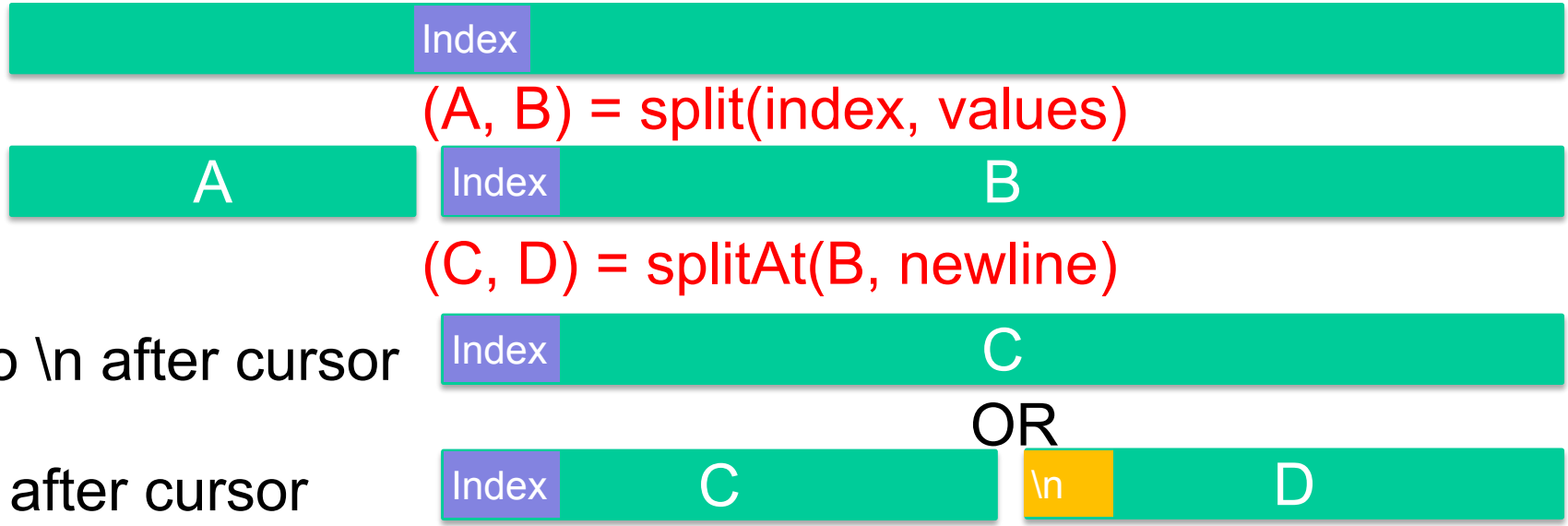
No ‘\n’ after the cursor

>= 1 ‘\n’ after the cursor

Now let’s draw out what we would do in each case…

# Problem 3

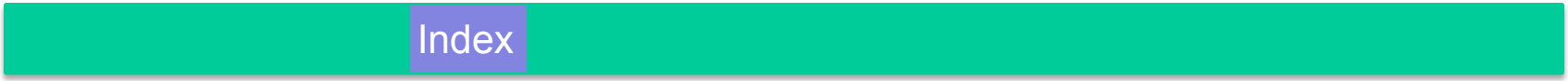
```
// Removes the line of text after the one containing the cursor index
void removeNextLine() {
```



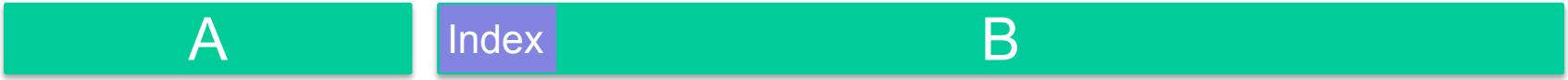

hi

# Problem 3

```
// Removes the line of text after the one containing the cursor index
void removeNextLine() {
```



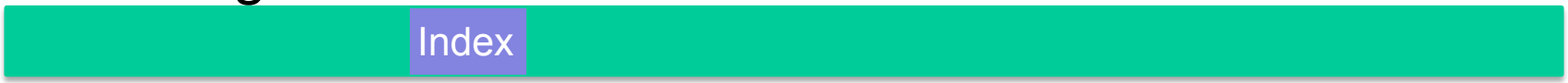

(A, B) = split(index, values)




(C, D) = splitAt(B, newline)

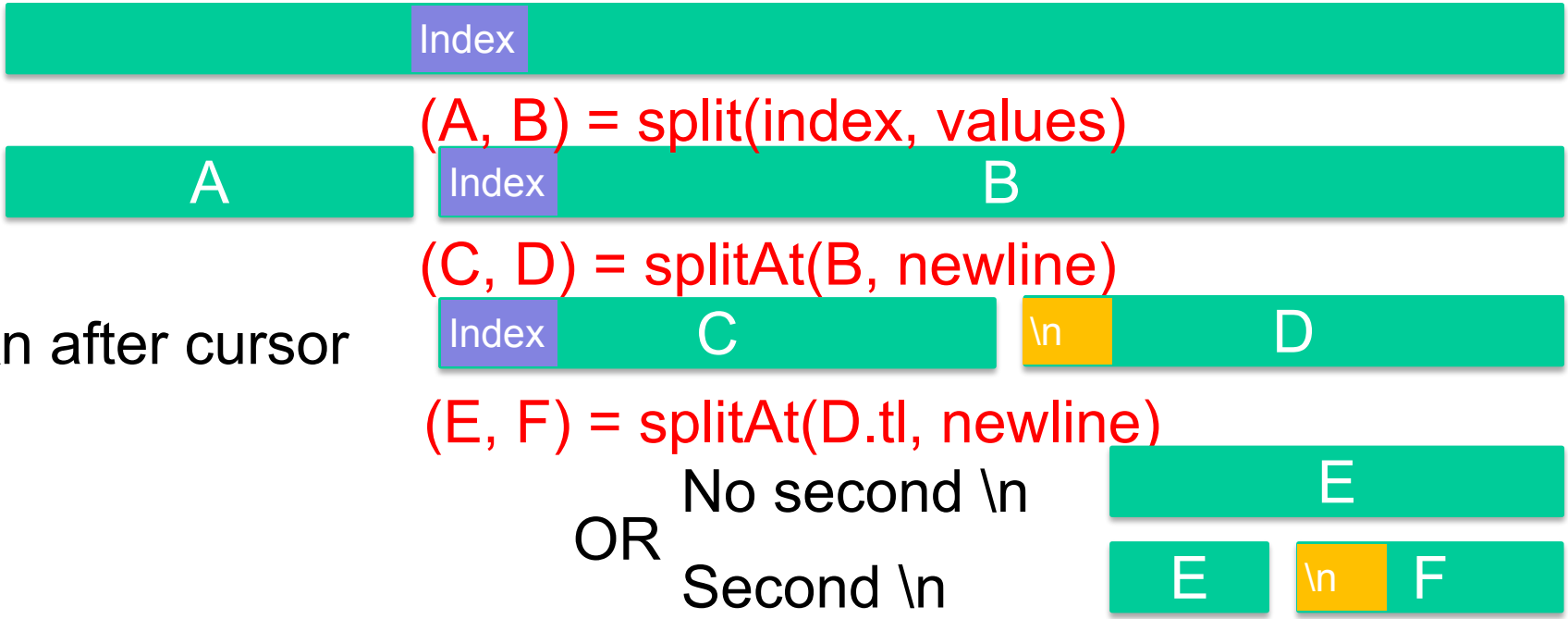
No \n after cursor


No change:



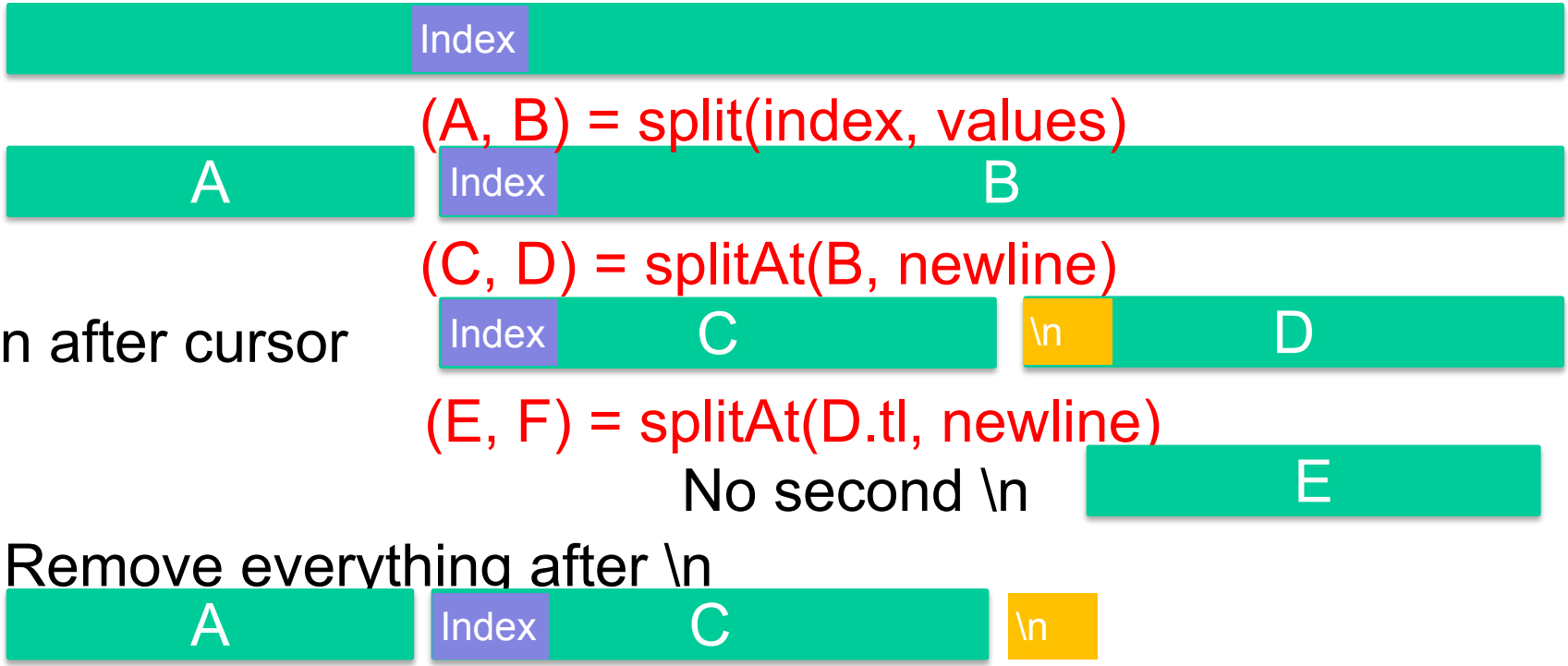
# Problem 3

```
// Removes the line of text after the one containing the cursor index
void removeNextLine() {
```



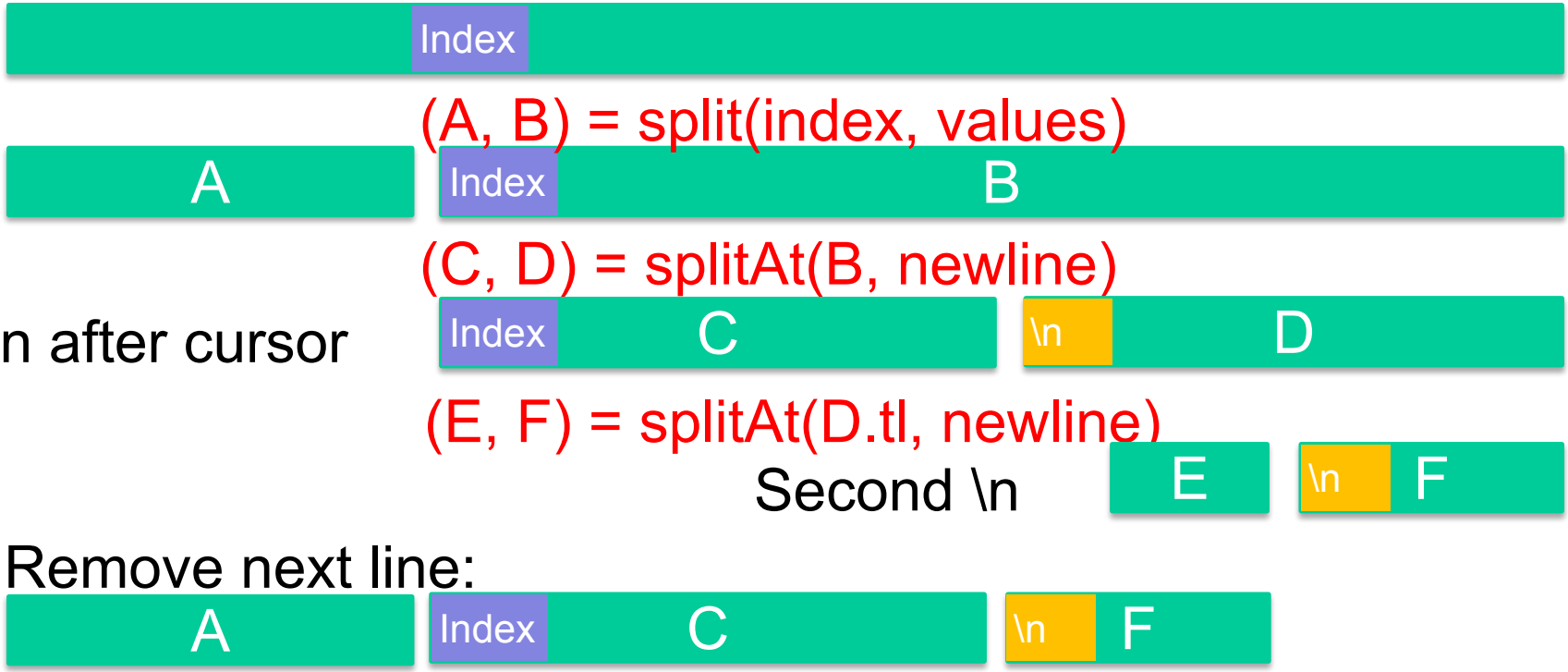
# Problem 3

```
// Removes the line of text after the one containing the cursor index
void removeNextLine() {
```



# Problem 3

```
// Removes the line of text after the one containing the cursor index
void removeNextLine() {
```




GOOD
LUCK!