Suggestions for Assignment 2 (Parts 3-4)

Outline

Feedback from Assignment 1

How to merge stacks ensuring max size 5?

- Example about how to remove pieces from the stack ensuring that max 5 pieces are on the stack.

Some Feedback on Assignment 1

Exercise

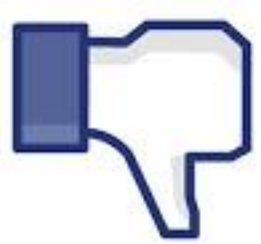
For each case identify whether it is correct or not.

```
char greeting[] = "hello";
strcpy(greeting, "hi how are you?");
```

Exercise

For each case identify whether it is correct or not.

```
char greeting[] = "hello";
strcpy(greeting, "hi how are you?");
```



Exercise

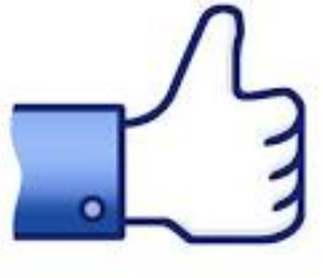
For each case identify whether it is correct or not.

```
char greeting[20] = "hello";
strcpy(greeting, "hi how are you?");
```

Exercise

For each case identify whether it is correct or not.

```
char greeting[20] = "hello";
strcpy(greeting, "hi how are you?");
```



Exercise

For each case identify whether it is correct or not.

```
char greeting[20] = "hello";
strcpy(greeting, "hi how are you?
Whats up bro");
```

Exercise

For each case identify whether it is correct or not.

```
char greeting[20] = "hello";
strcpy(greeting, "hi how are you?
Whats up bro");
```

👎

Exercise

For each case identify whether it is correct or not.

```
char *greeting;
strcpy(greeting, "hi how are you?");
```

Exercise

For each case identify whether it is correct or not.

```
char *greeting;
strcpy(greeting, "hi how are you?");
```

👎

Exercise

For each case identify whether it is correct or not.

```
char *greeting = NULL;
strcpy(greeting, "hi how are you?");
```

Exercise

For each case identify whether it is correct or not.

```
char *greeting = NULL;
strcpy(greeting, "hi how are you?");
```

👎

Exercise

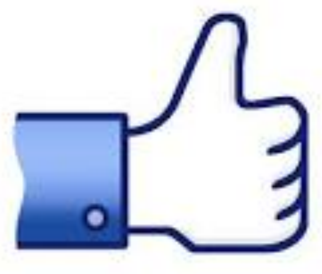
For each case identify whether it is correct or not.

```
char *greeting = (char *) malloc(20);
strcpy(greeting, "hi how are you?");
```

Exercise

For each case identify whether it is correct or not.

```
char *greeting = (char *) malloc(20);
strcpy(greeting, "hi how are you?");
```



Exercise

For each case identify whether it is correct or not.

```
char *greeting = (char *) malloc(20);
strcpy(greeting, "hi how are you?
Whats up bro");
```

Exercise

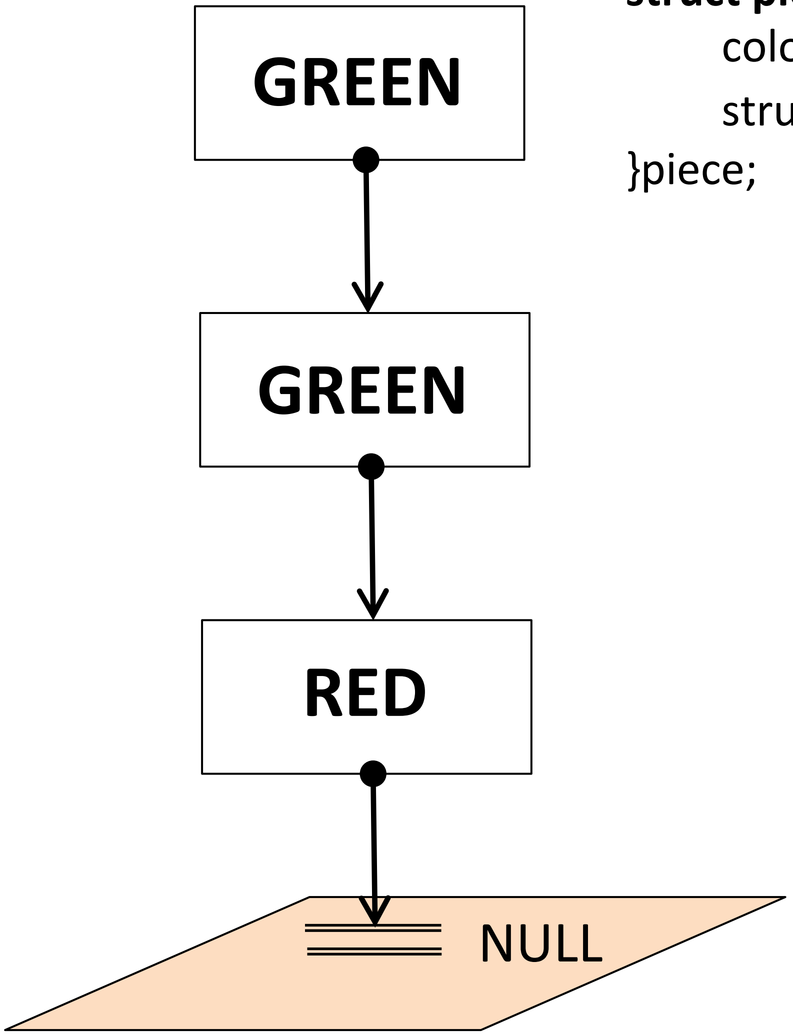
For each case identify whether it is correct or not.

```
char *greeting = (char *) malloc(20);
strcpy(greeting, "hi how are you?
Whats up bro");
```

👎

How to merge stacks ensuring that max size is 5?

Imagine that this is the stack that is on top of one of the squares (e.g., in position [2,2]) of your board



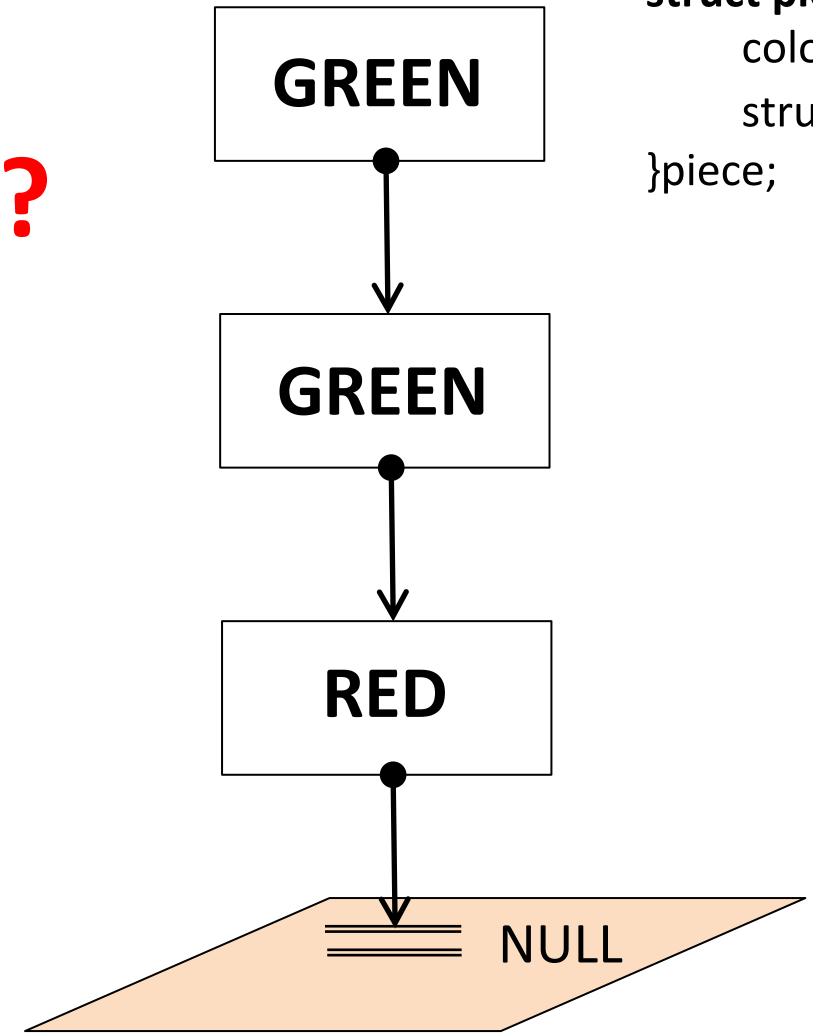
```
struct piece {
        color p_color;
        struct piece * next;
}piece;
```

```
struct square {
        square_type type;
        piece * stack;
        int num_pieces;
}square;
```

Imagine that this is the stack that is on top of one of the squares (e.g., in position [2,2]) of your board

```
square board[8][8];
```

board[2][2].stack ?

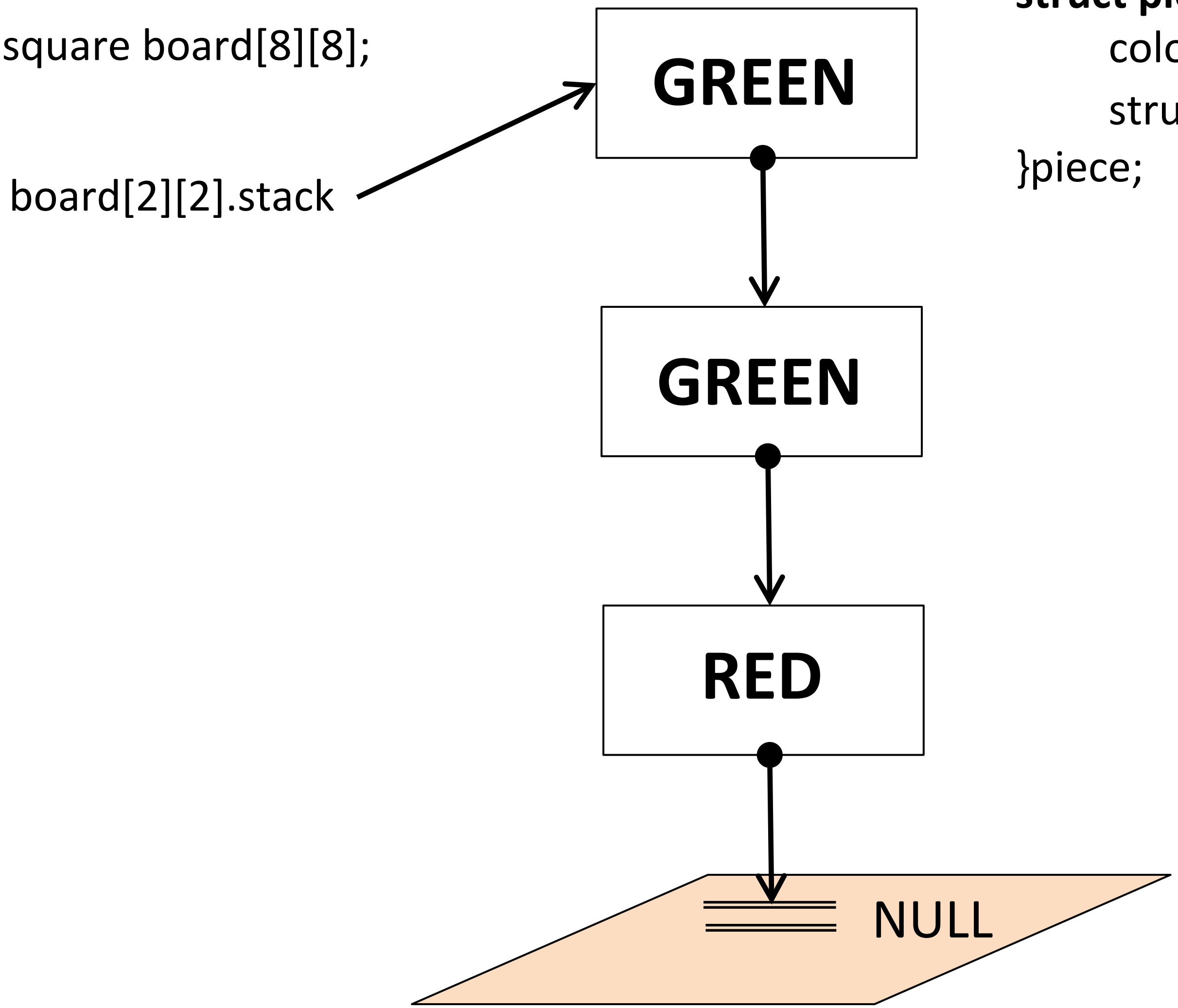


```
struct piece {
        color p_color;
        struct piece * next;
}piece;
```

```
struct square {
        square_type type;
        piece * stack;
        int num_pieces;
}square;
```

Imagine that this is the stack that is on top of one of the squares (e.g., in position [2,2]) of your board

```
square board[8][8];
```

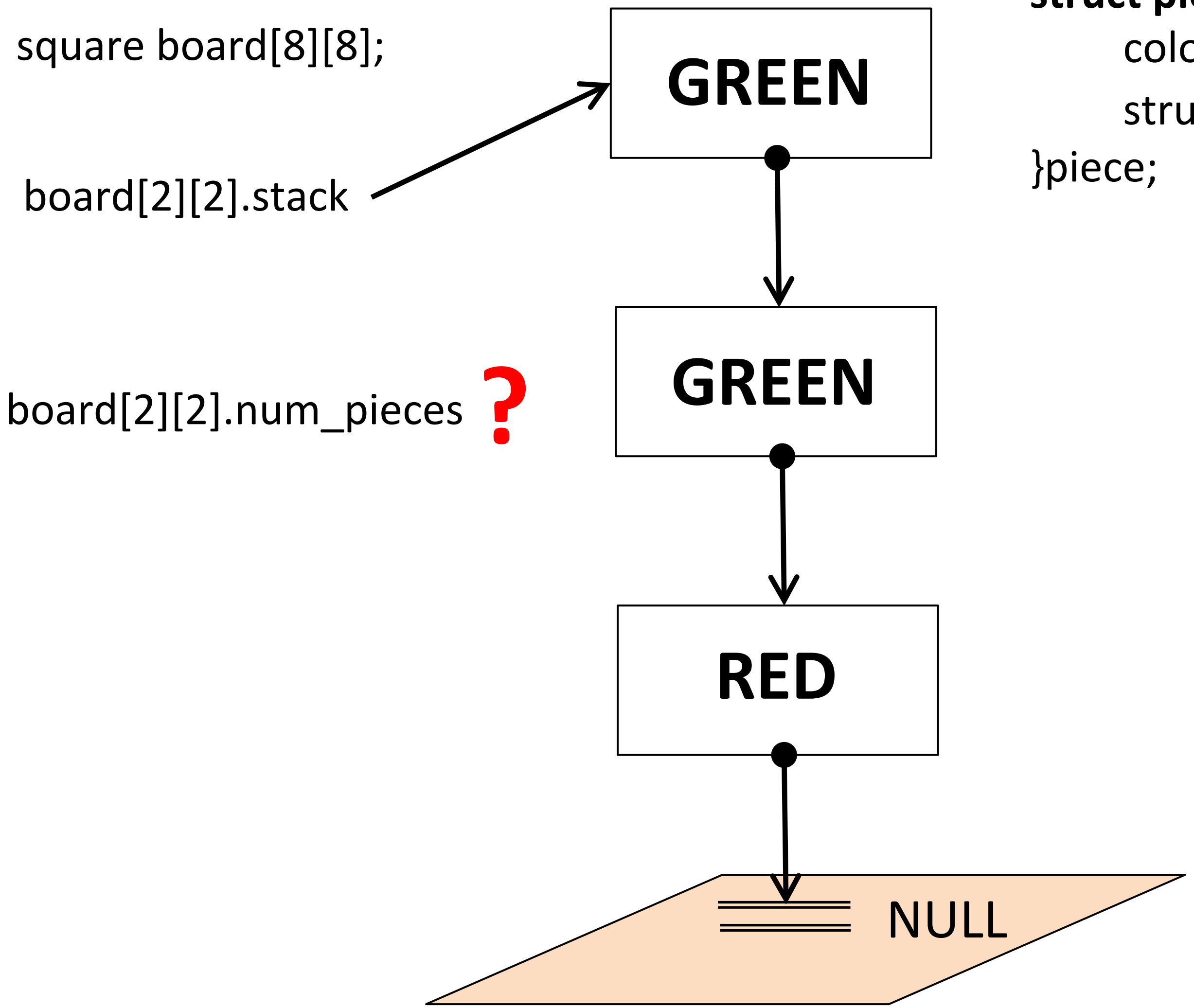


```
struct piece {
        color p_color;
        struct piece * next;
}piece;
```

```
struct square {
        square_type type;
        piece * stack;
        int num_pieces;
}square;
```

Imagine that this is the stack that is on top of one of the squares (e.g., in position [2,2]) of your board

square board[8][8];

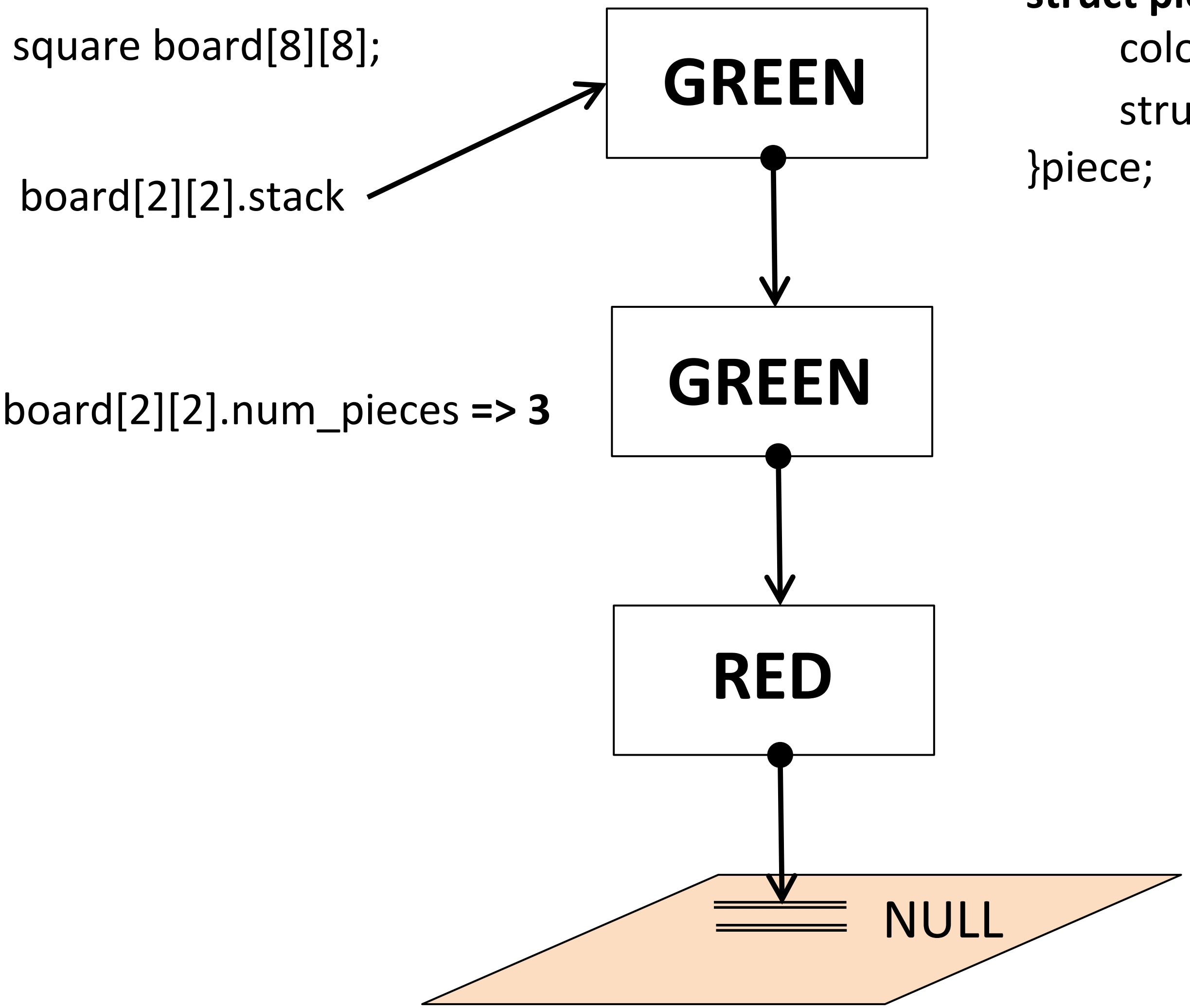


```
struct piece {
	color p_color;
	struct piece * next;
}piece;
```

```
struct square {
	square_type type;
	piece * stack;
	int num_pieces;
}square;
```

Imagine that this is the stack that is on top of one of the squares (e.g., in position [2,2]) of your board

square board[8][8];

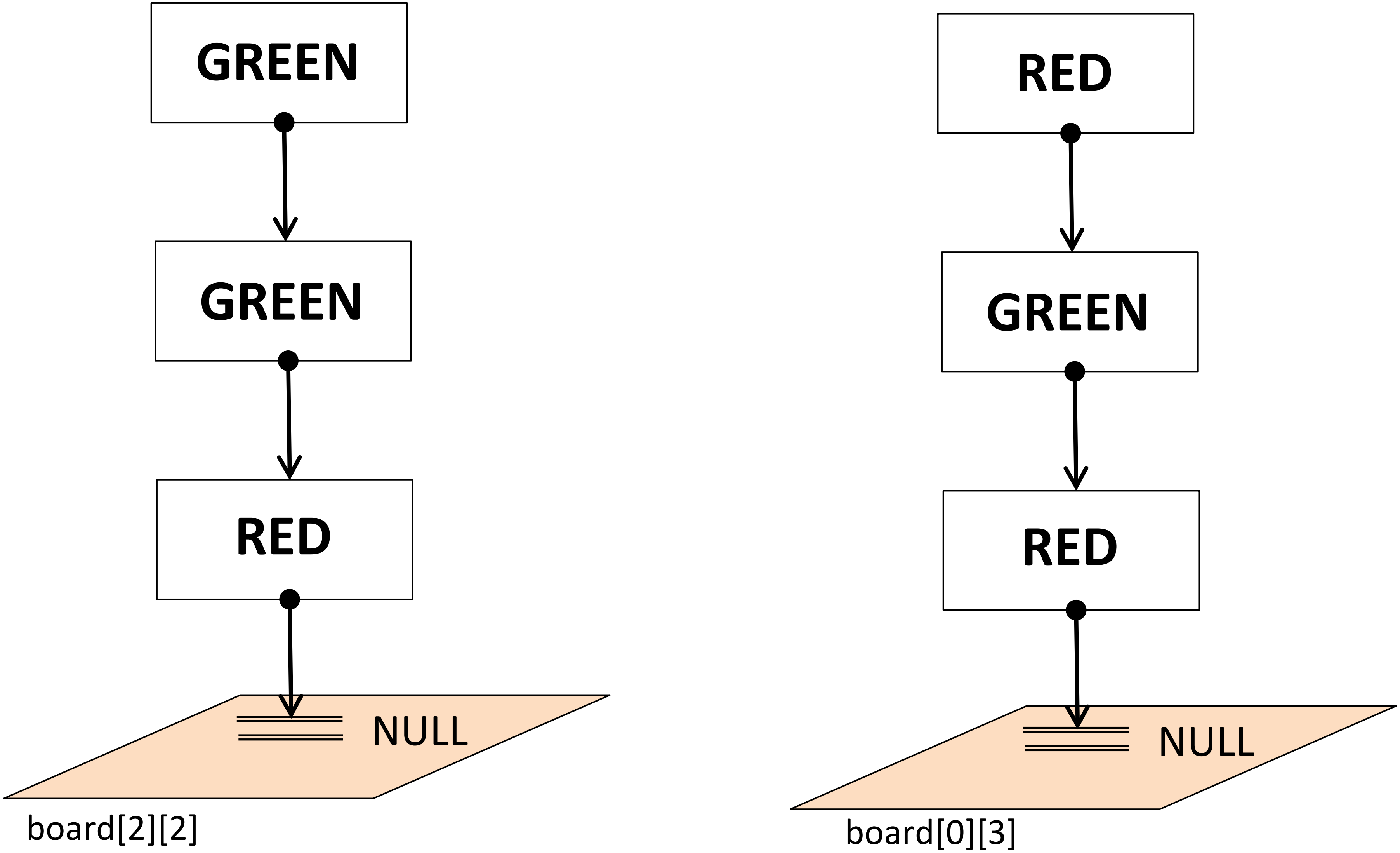


board[2][2].num_pieces **=> 3**

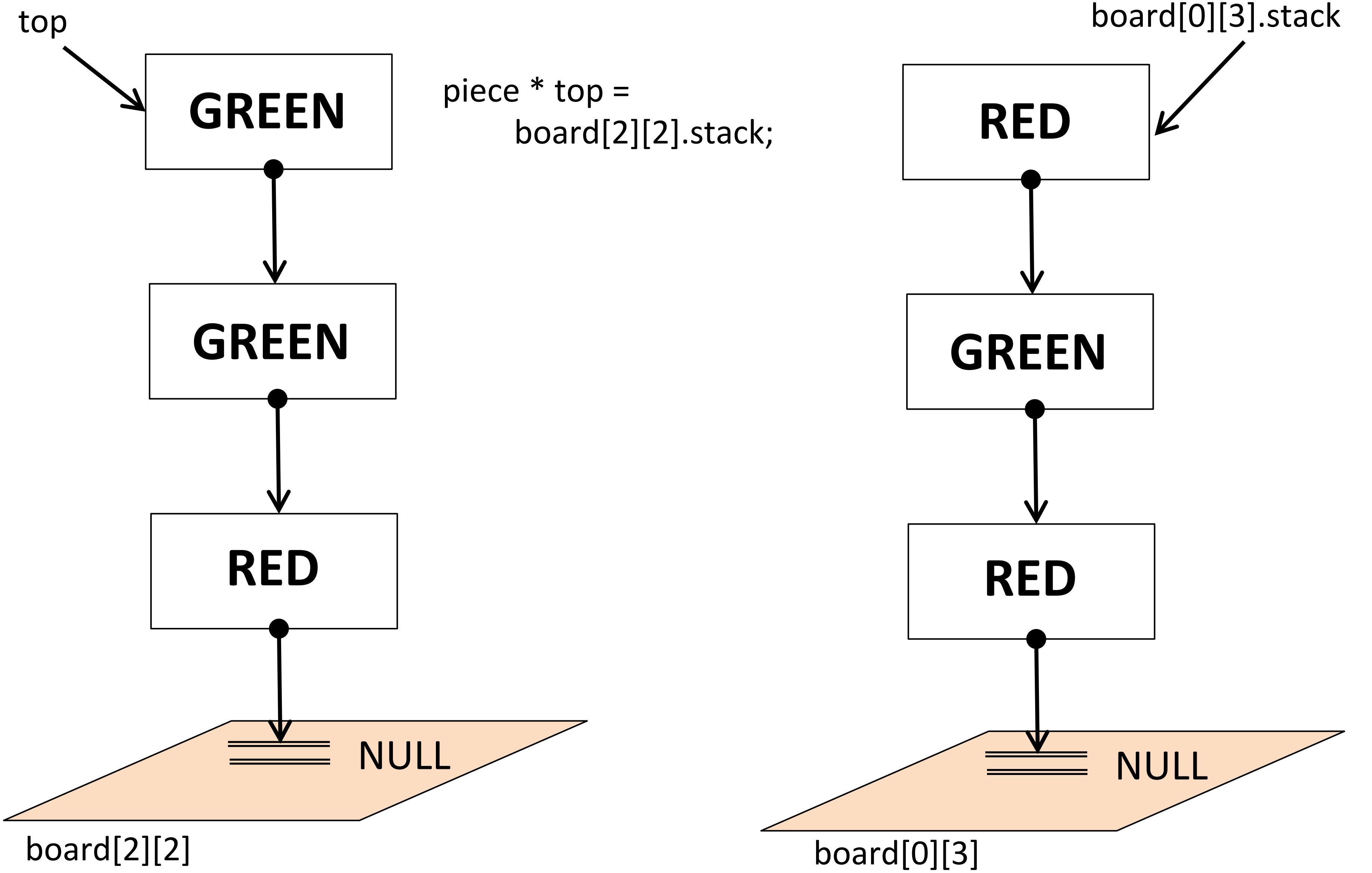
```
struct piece {
        color p_color;
        struct piece * next;
}piece;
```

```
struct square {
        square_type type;
        piece * stack;
        int num_pieces;
}square;
```

Imagine that we want to move the stack in square [2,2] on square [0,3]

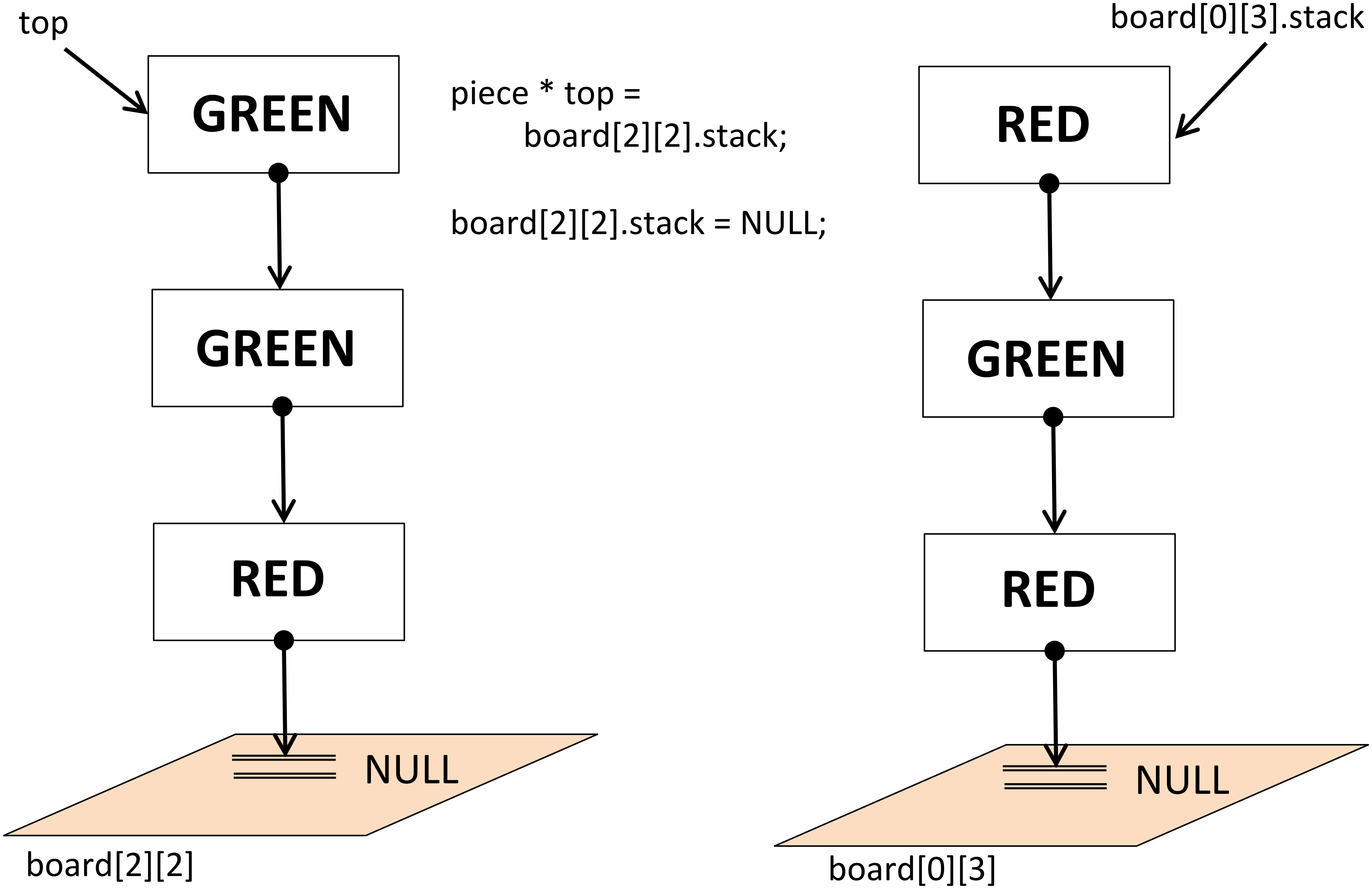


Imagine that we want to move the stack in square [2,2] on square [0,3]



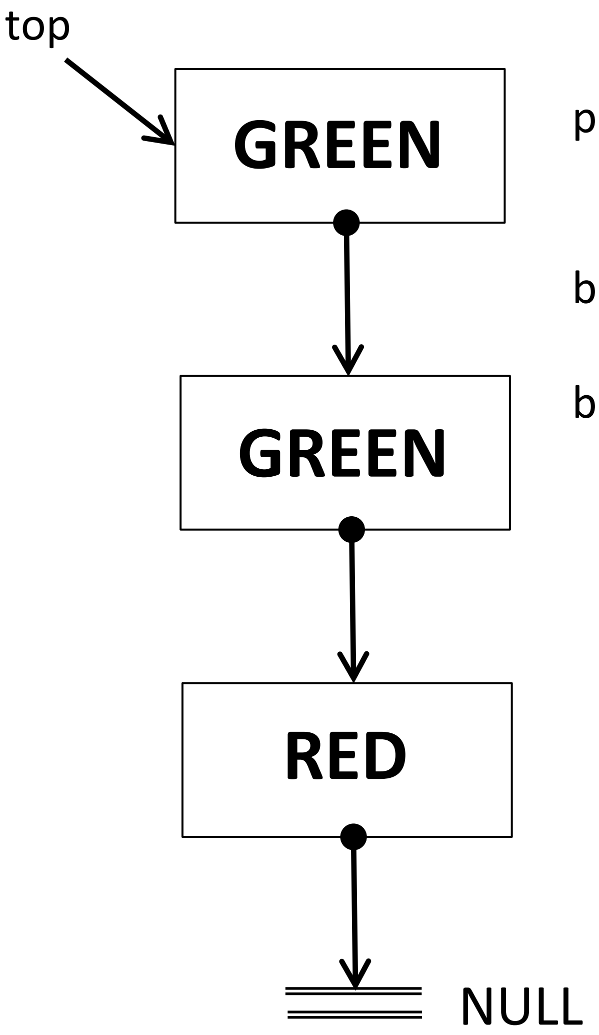
```
piece * top =
        board[2][2].stack;
```

Imagine that we want to move the stack in square [2,2] on square [0,3]

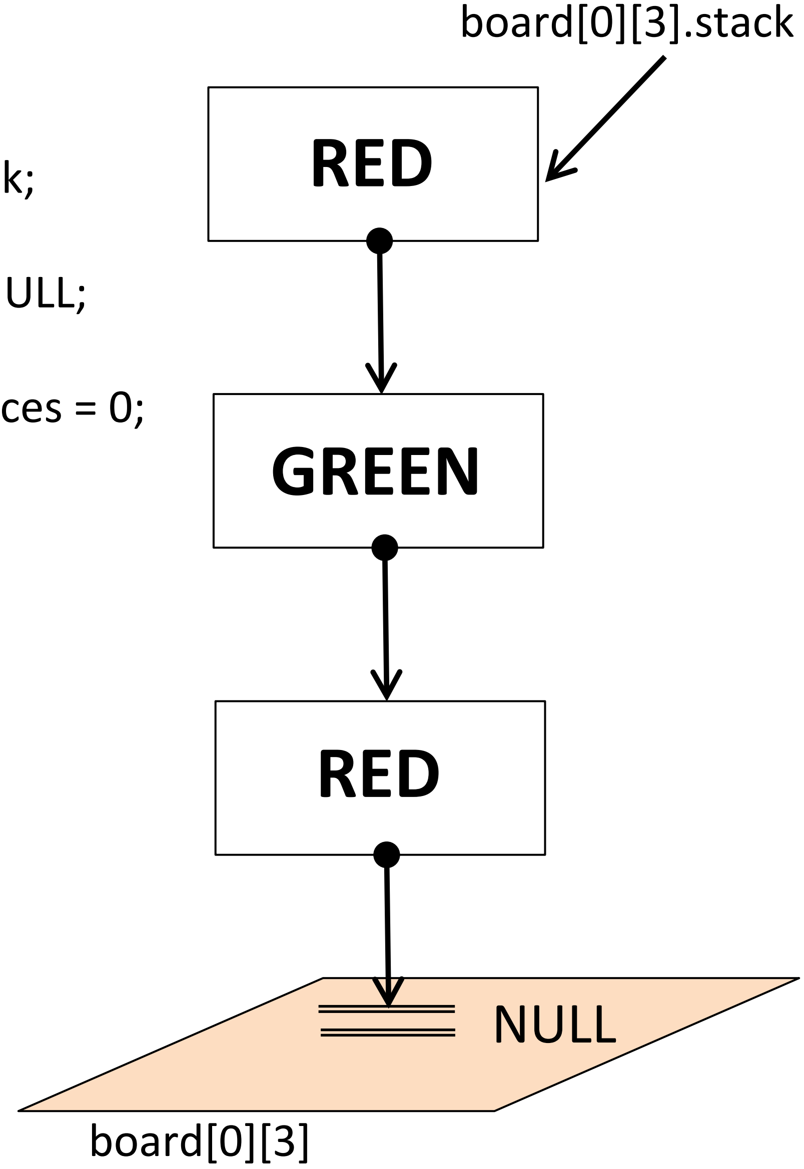


```
piece * top =
        board[2][2].stack;

board[2][2].stack = NULL;
```

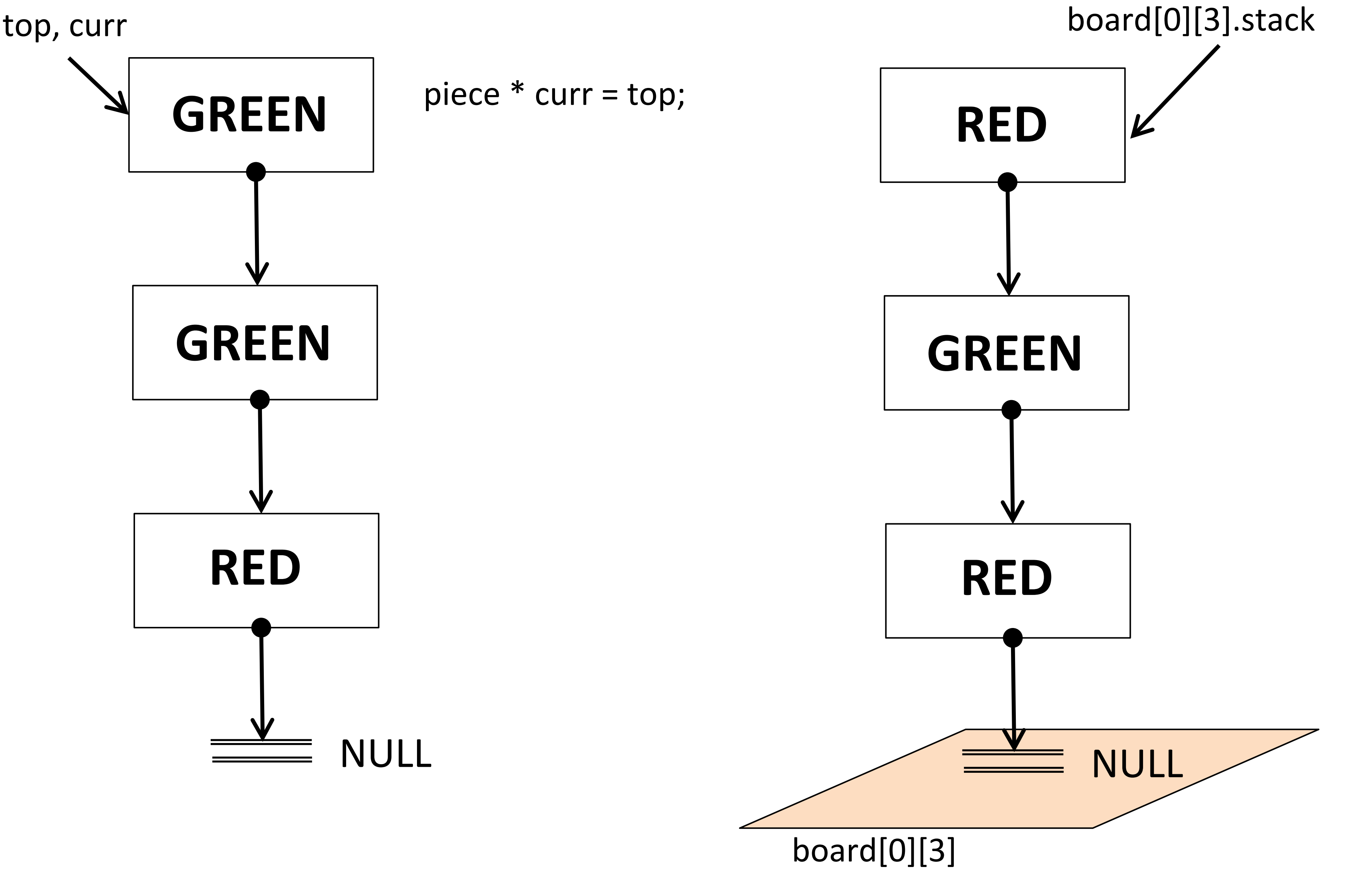
Imagine that we want to move the stack in square [2,2] on square [0,3]



```
piece * top =
        board[2][2].stack;

board[2][2].stack = NULL;

board[2][2].num_pieces = 0;
```

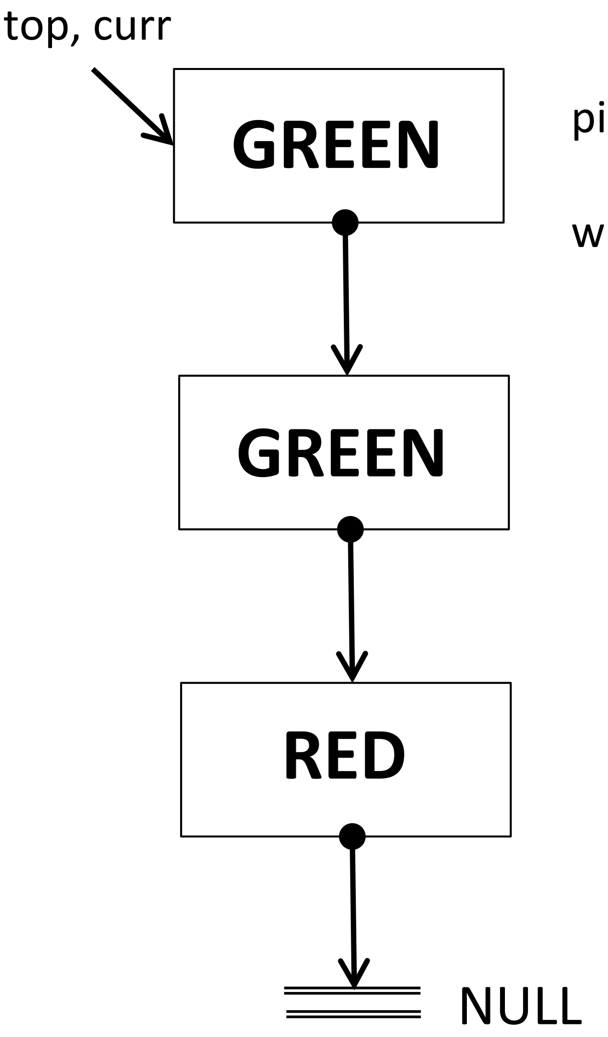


Imagine that we want to move the stack in square [2,2] on square [0,3]

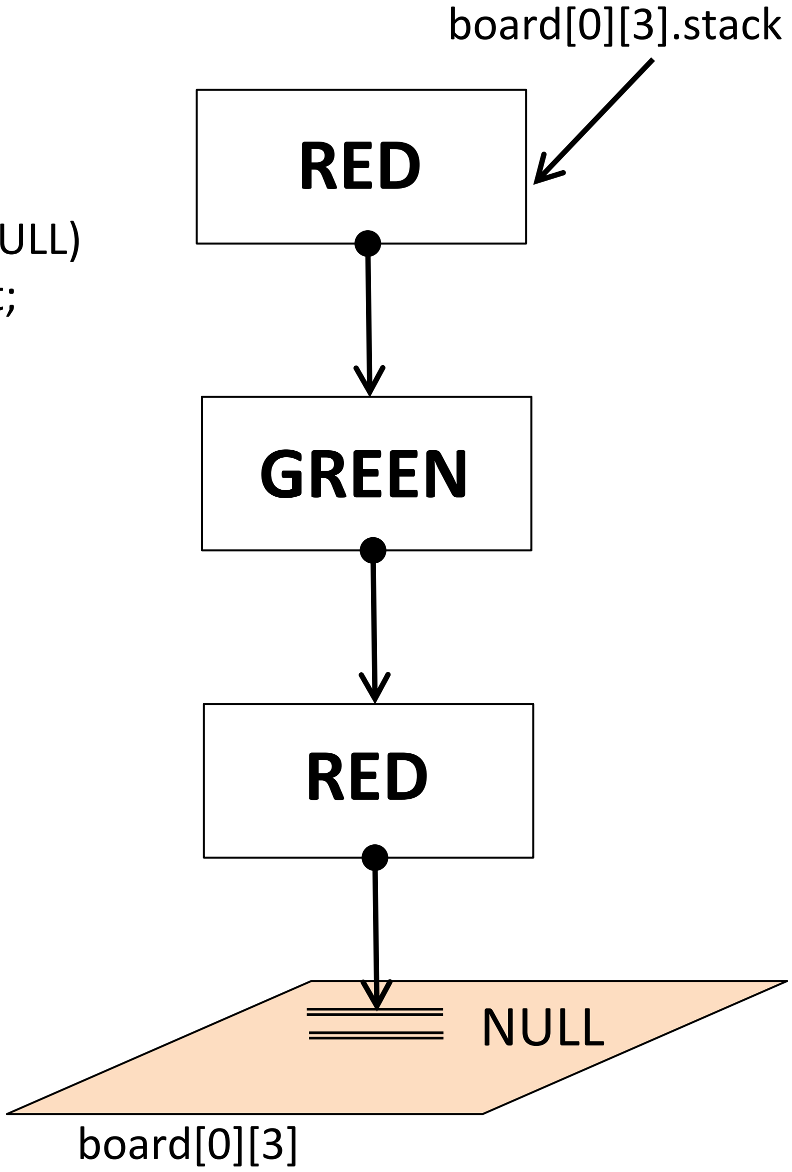


piece * curr = top;

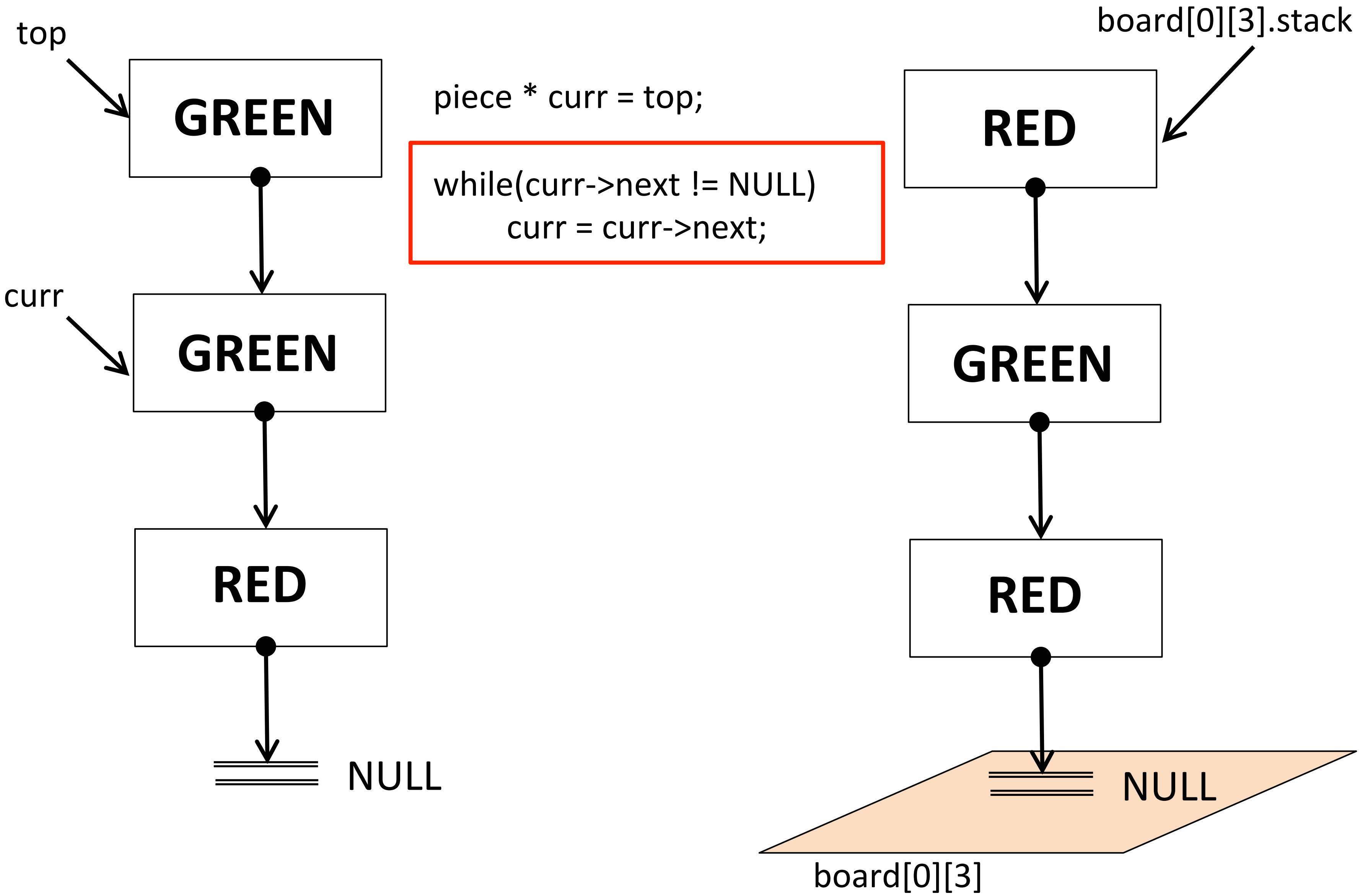
Imagine that we want to move the stack in square [2,2] on square [0,3]



```
piece * curr = top;

while(curr->next != NULL)
        curr = curr->next;
```

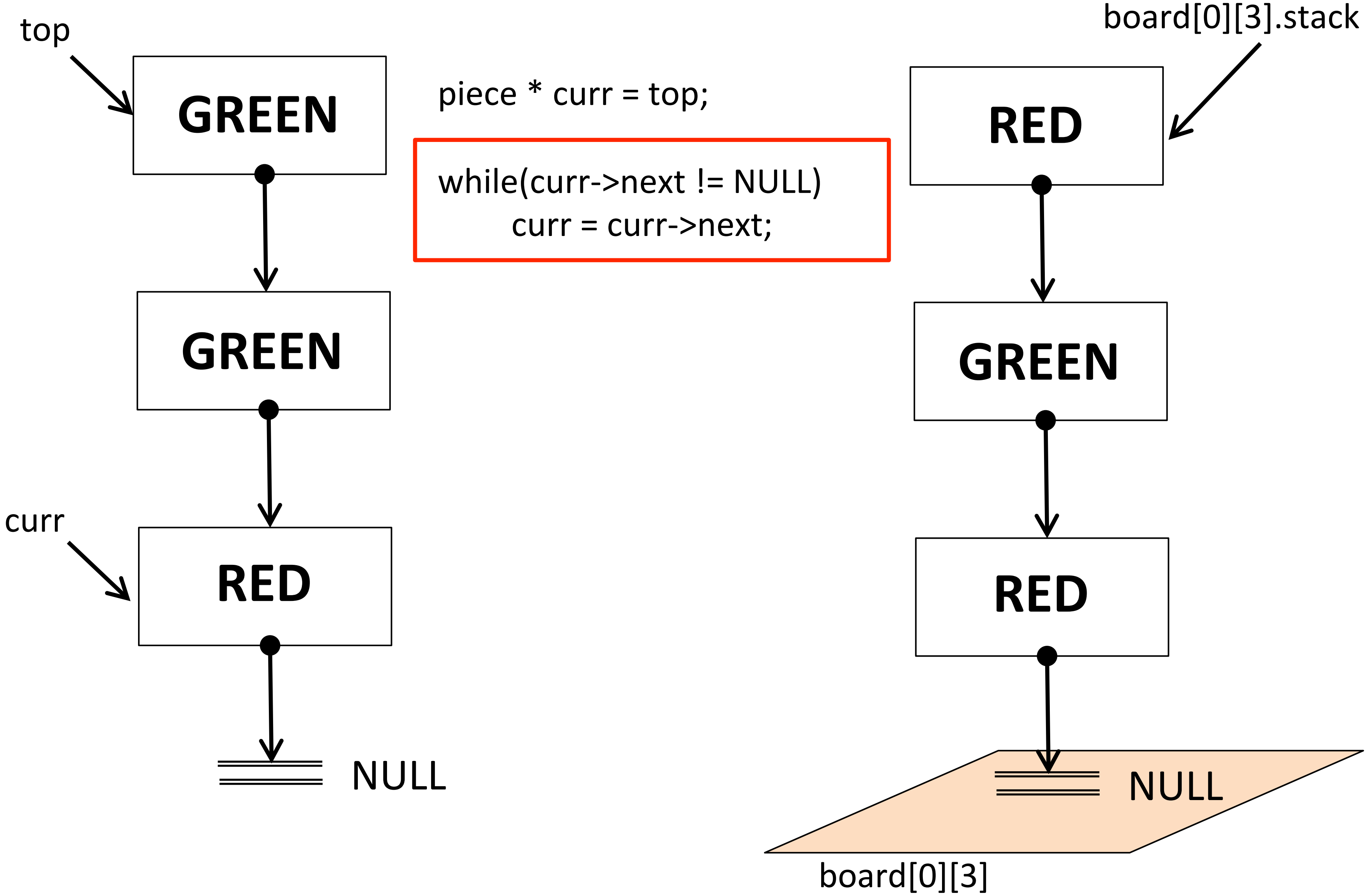


Imagine that we want to move the stack in square [2,2] on square [0,3]



```
piece * curr = top;

while(curr->next != NULL)
        curr = curr->next;
```

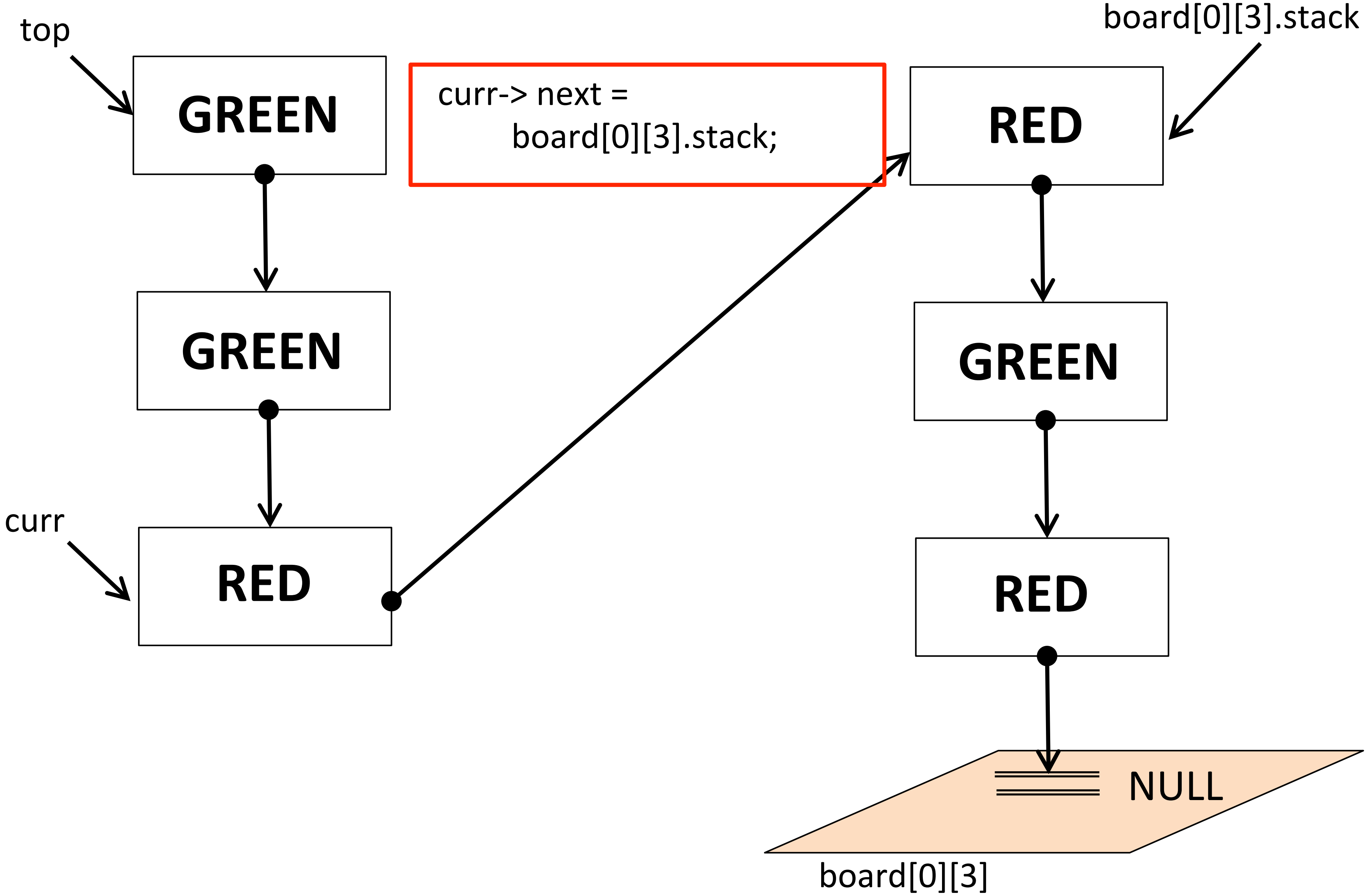
Imagine that we want to move the stack in square [2,2] on square [0,3]



```
piece * curr = top;
```

```
while(curr->next != NULL)
        curr = curr->next;
```

Imagine that we want to move the stack in square [2,2] on square [0,3]



```
curr-> next =
        board[0][3].stack;
```

top

GREEN

GREEN

curr

RED

board[0][3].stack

RED

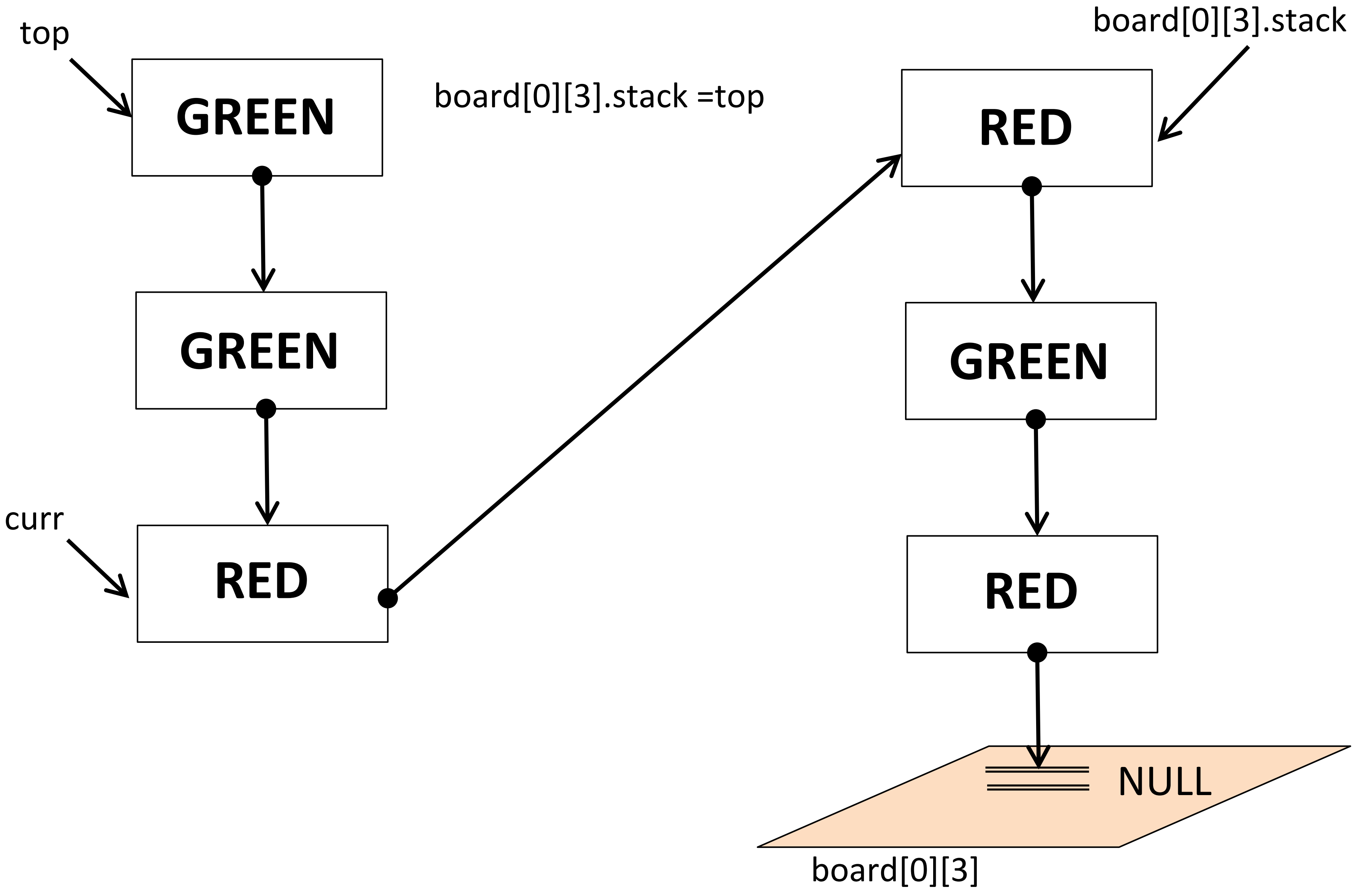
GREEN

RED

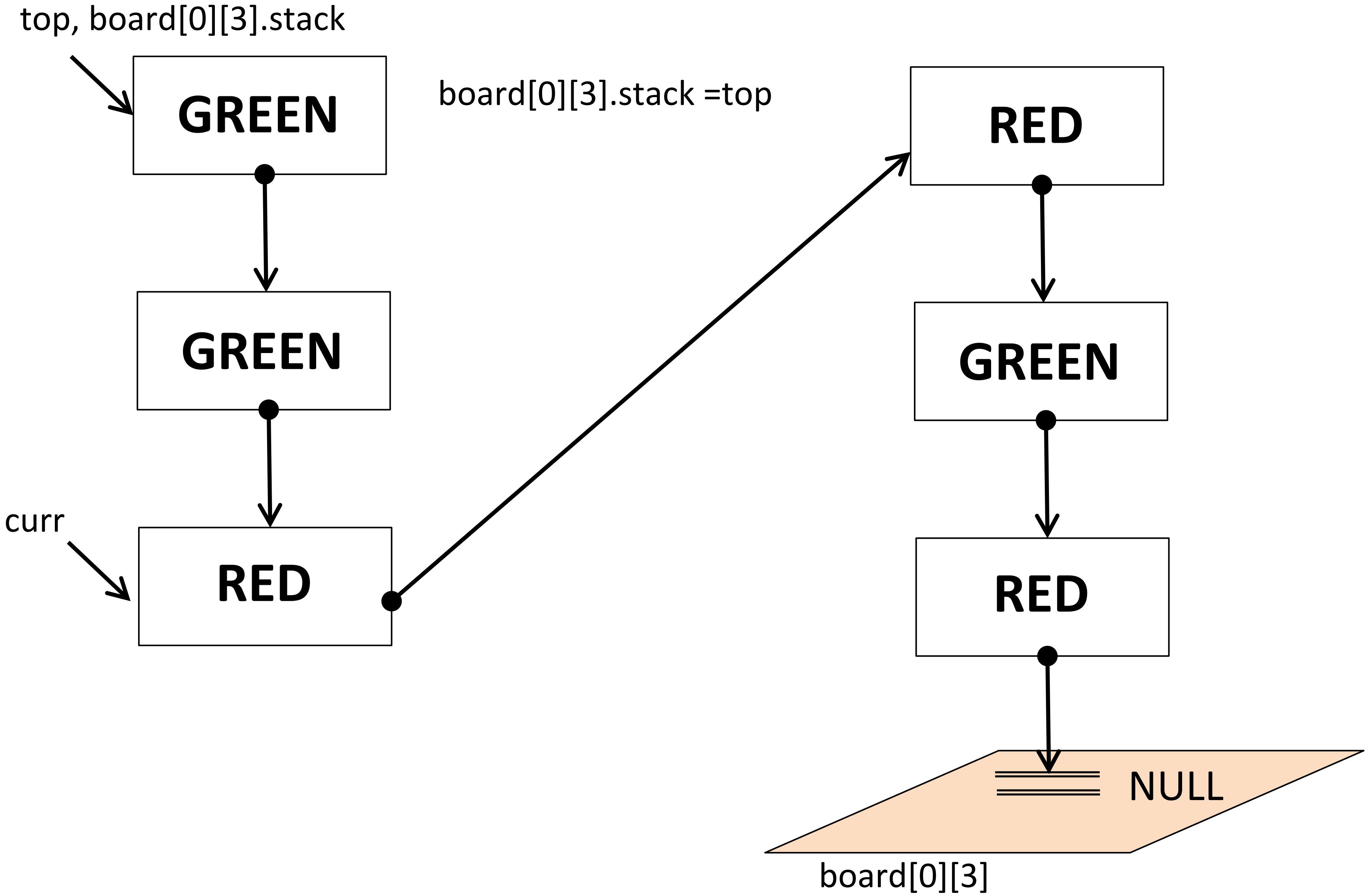
NULL

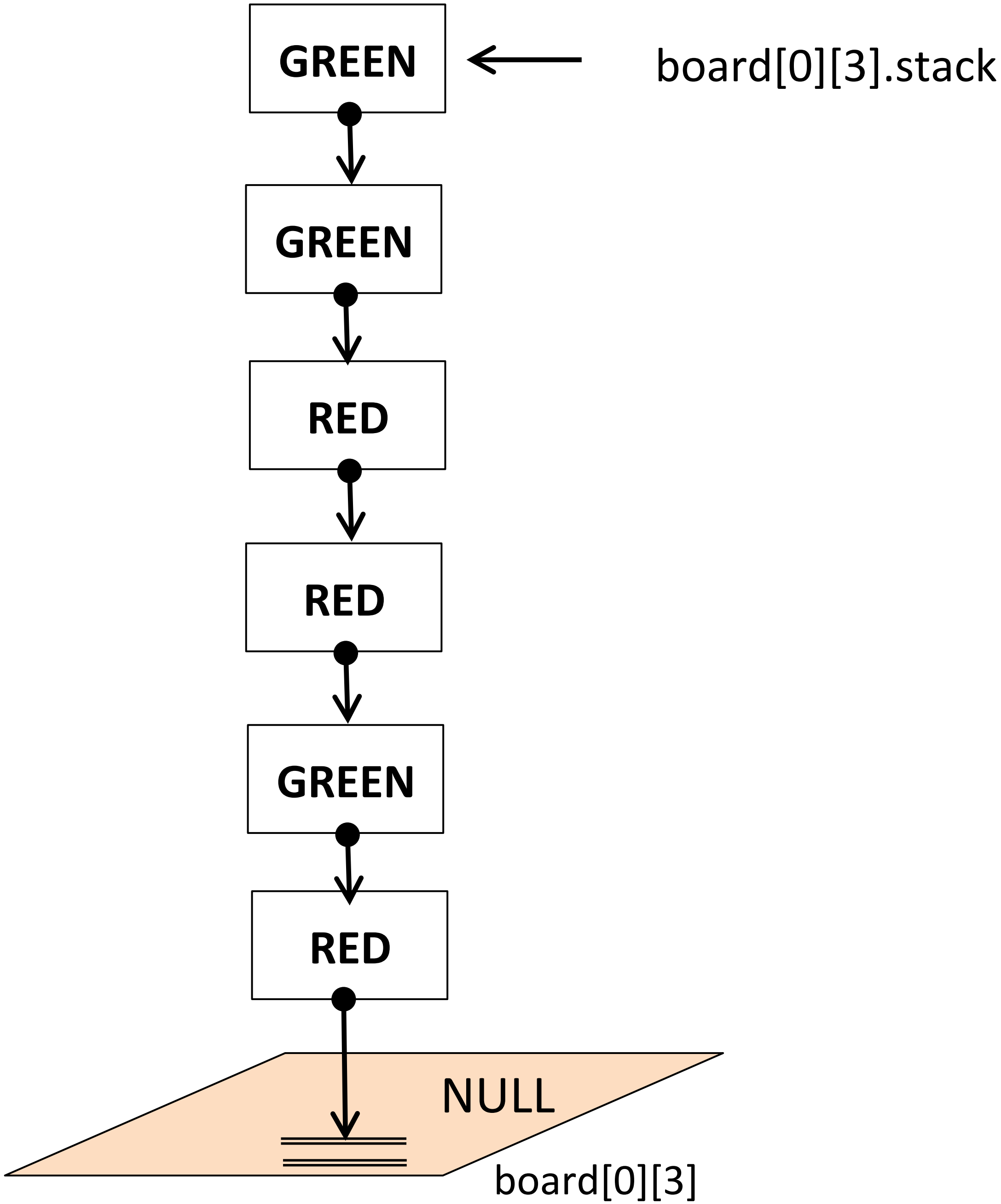
board[0][3]

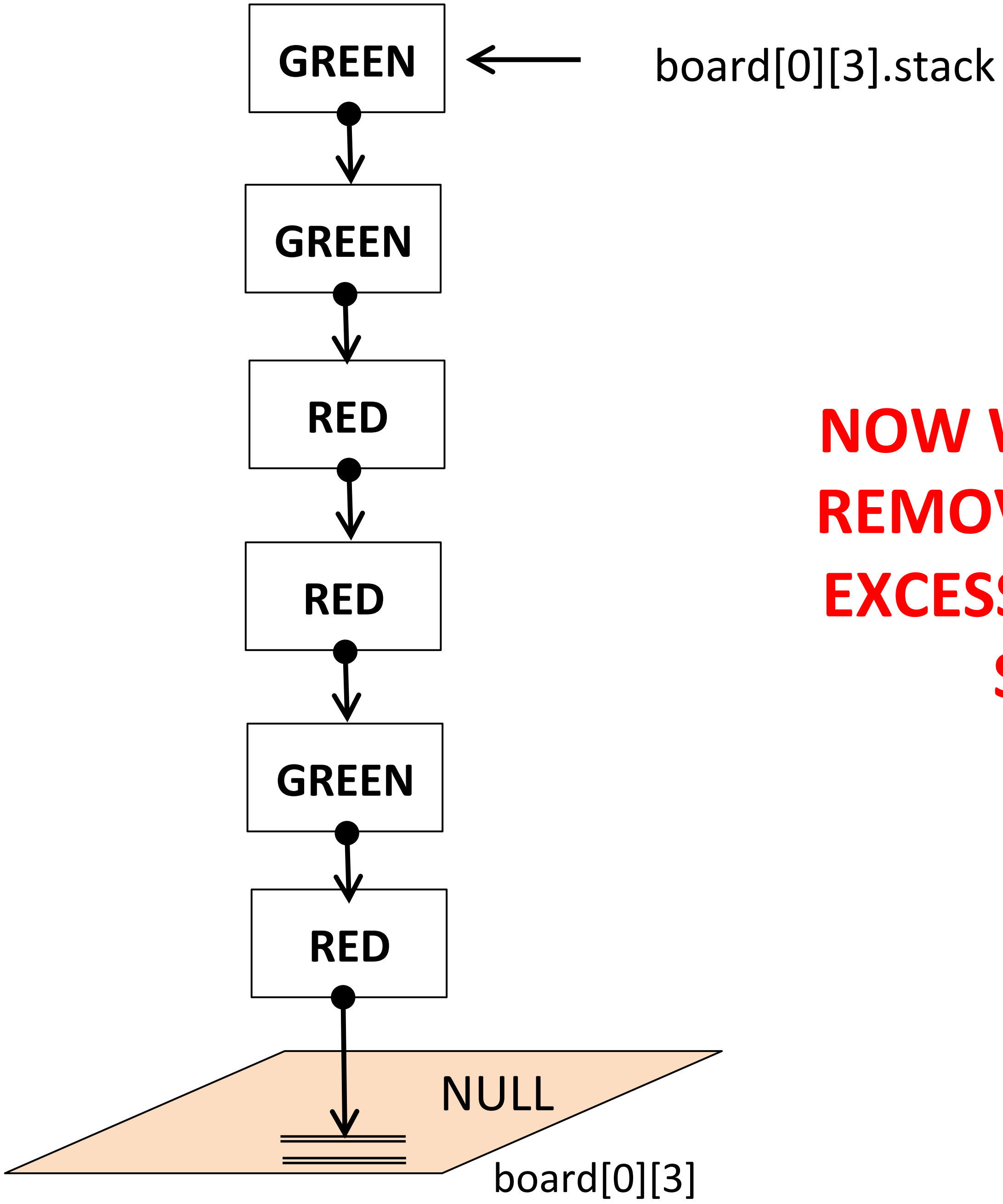
Imagine that we want to move the stack in square [2,2] on square [0,3]



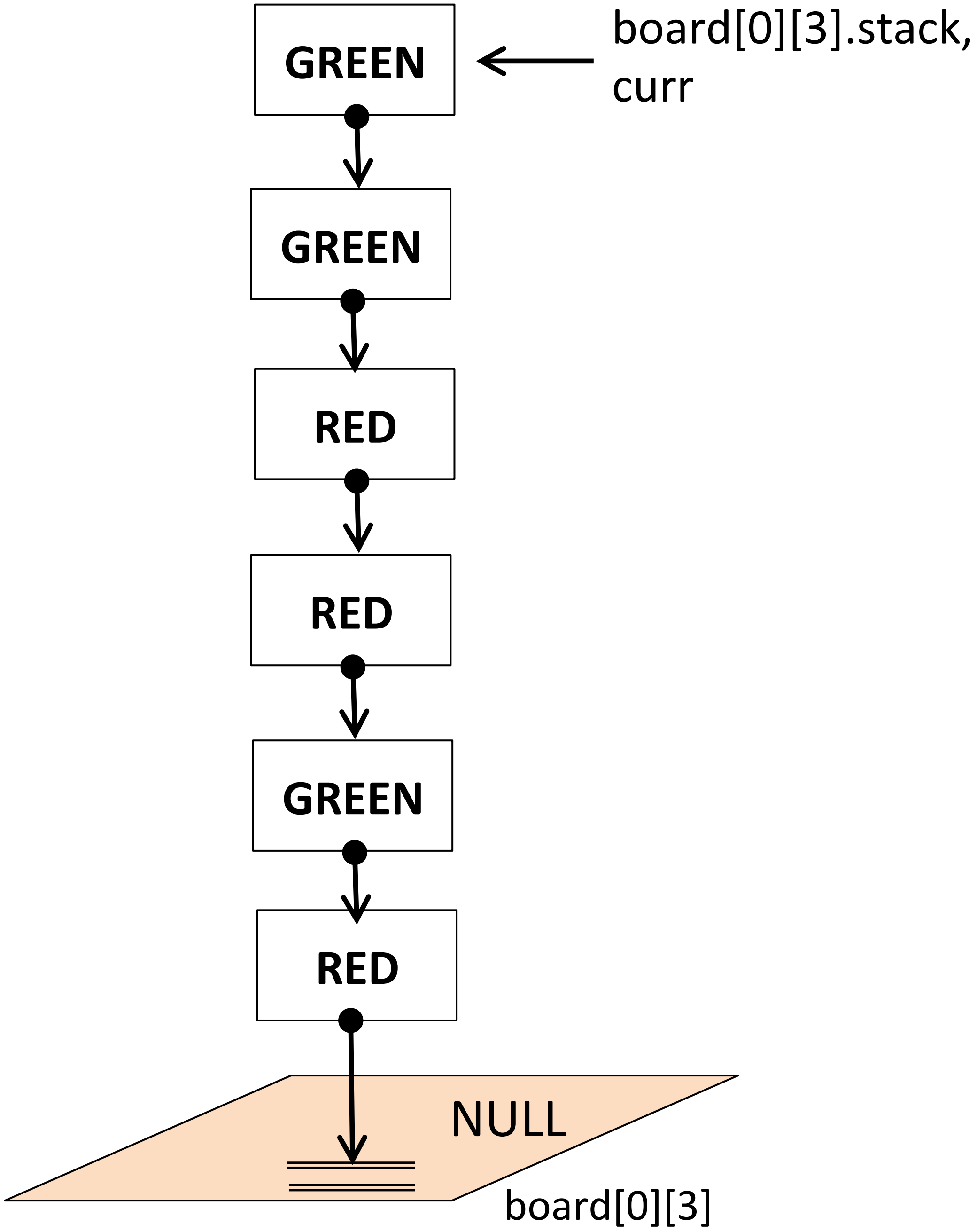
Imagine that we want to move the stack in square [2,2] on square [0,3]




GREEN
board[0][3].stack
GREEN
RED
RED
GREEN
RED
NULL
board[0][3]



NOW WE NEED TO REMOVE PIECES IN EXCESS FROM THE STACK



board[0][3].stack,
curr

GREEN

GREEN

RED

RED

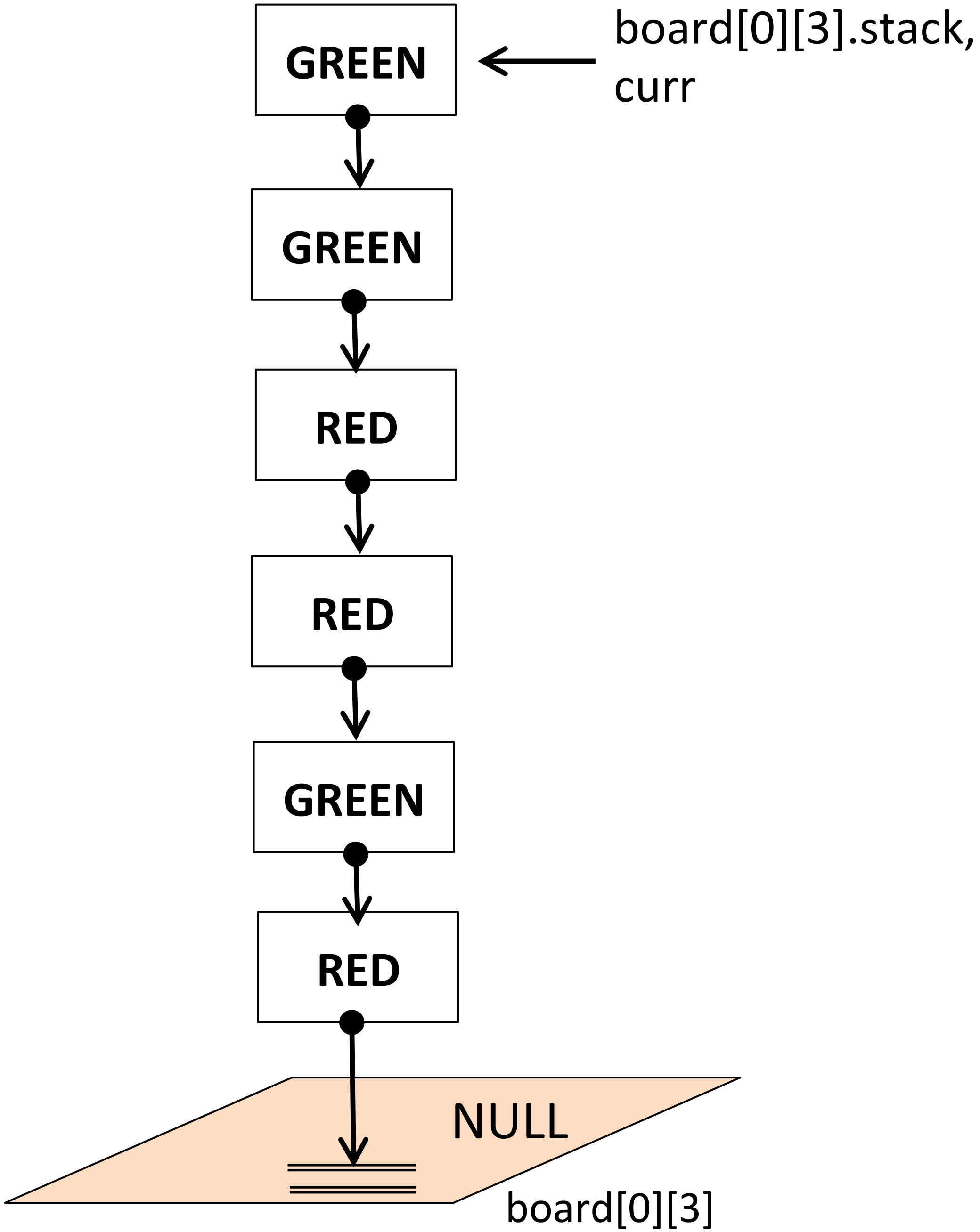
GREEN

RED

NULL

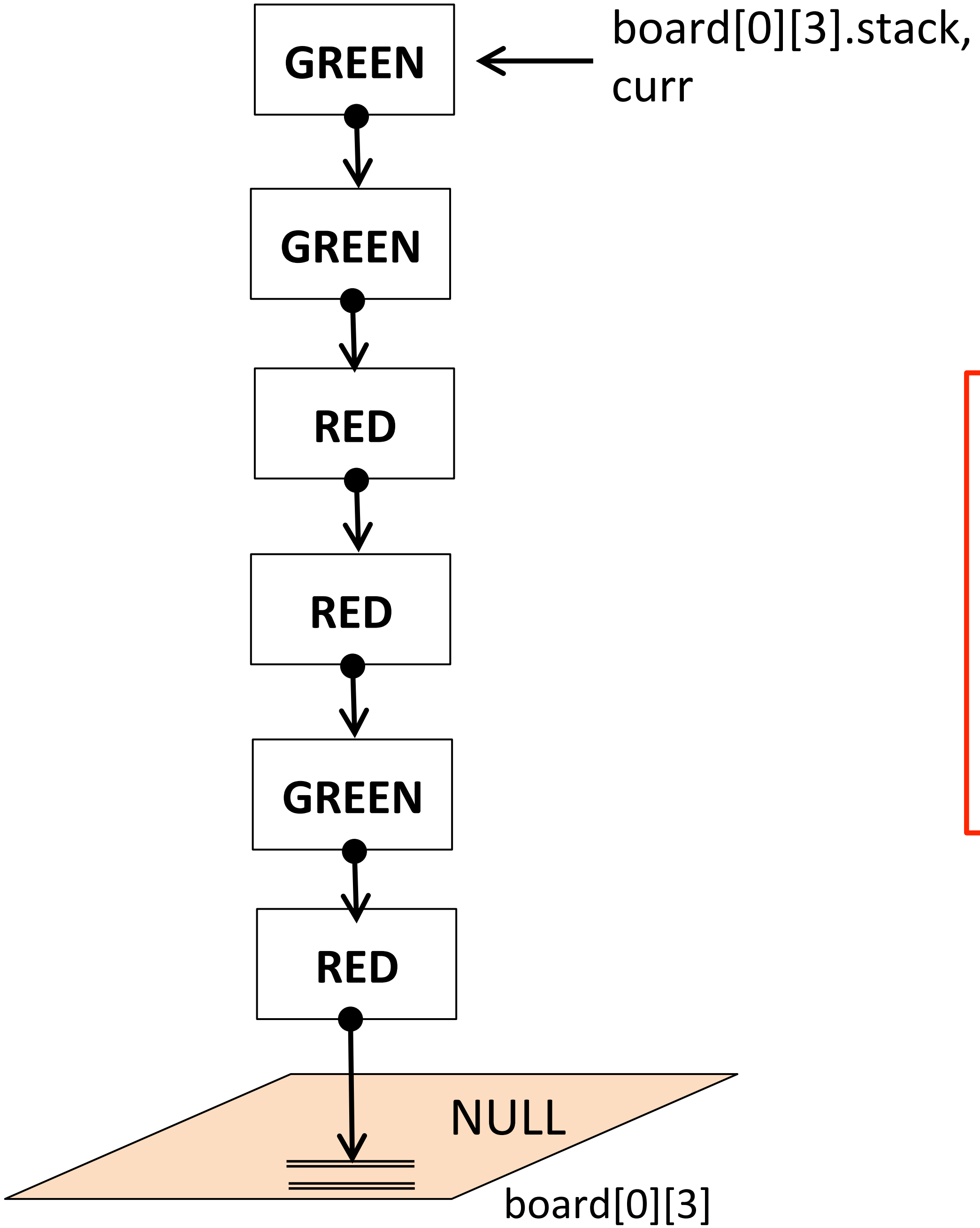
board[0][3]

```
int count = 1;
piece *last = NULL;
```



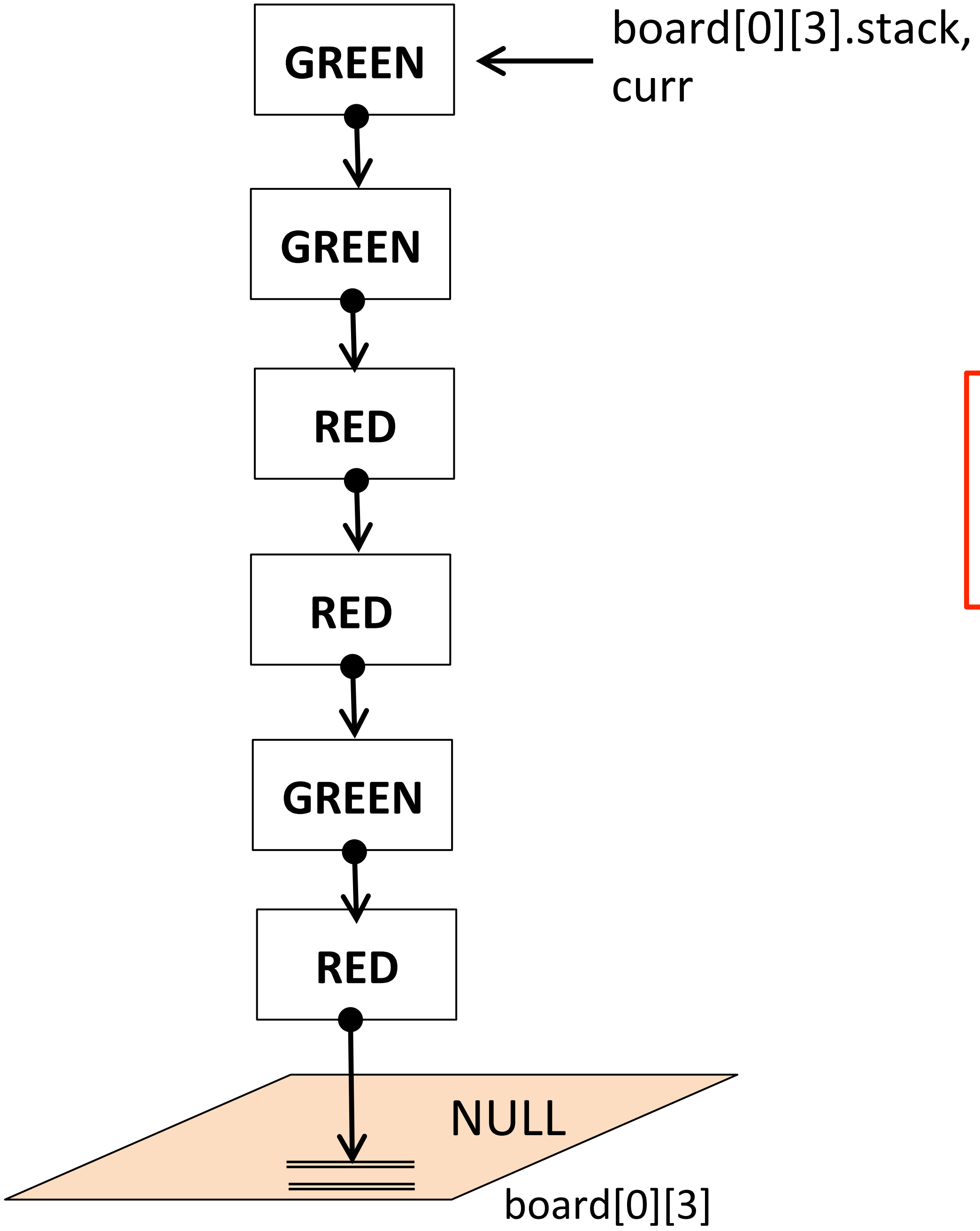
```
int count = 1;

piece *last = NULL;
```

```
while(curr != NULL){
        if(count < 5){
                curr = curr -> next;
                count++;
        } else {
                last = curr;
        }
}
```



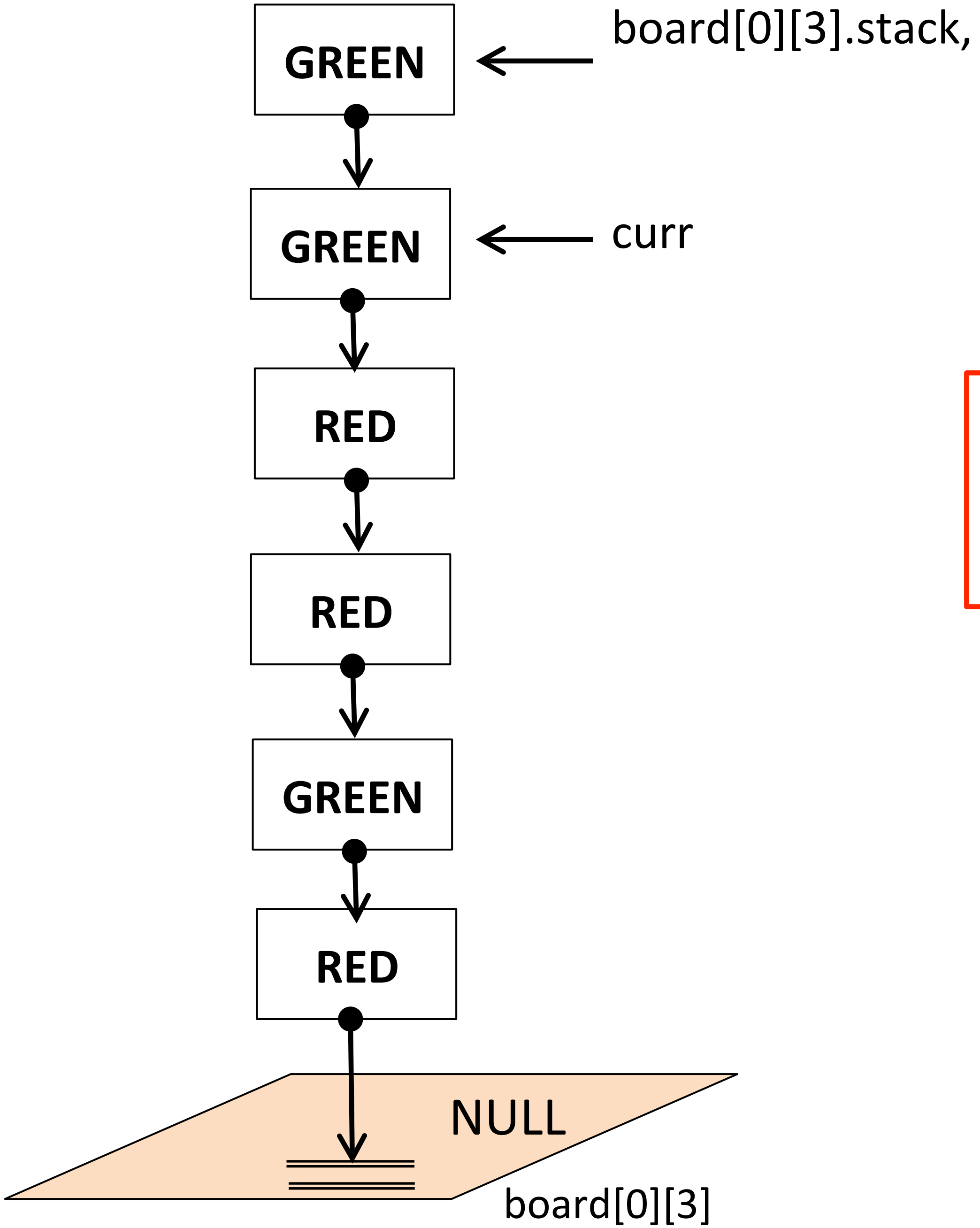
```
int count = 1;

piece *last = NULL;
```

```
while(curr != NULL){
        if(count < 5){
                curr = curr -> next;
                count++;
        } else {
                last = curr;
        }
}
```



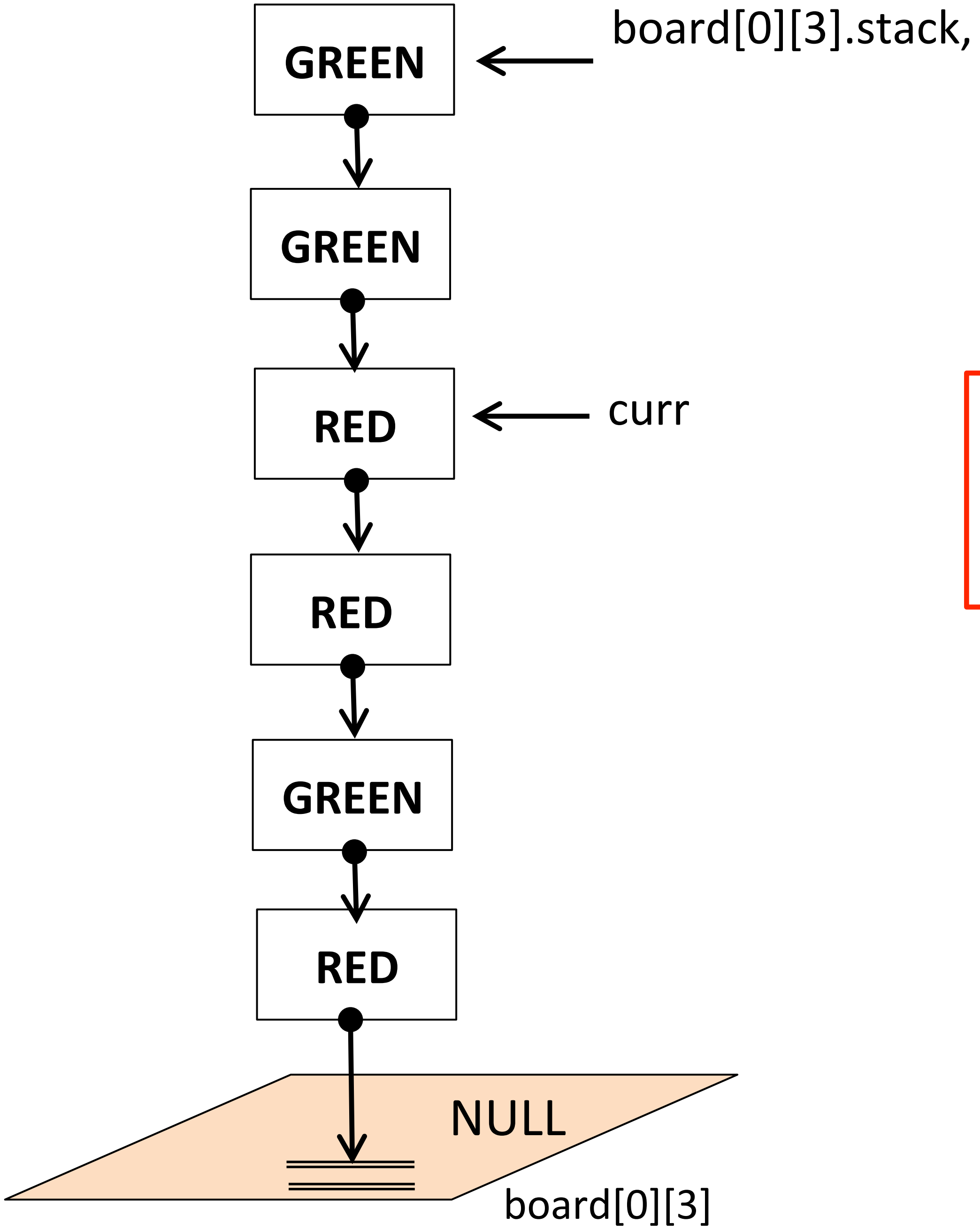
```
count = 1;

last = NULL;
```

```
while(curr != NULL){
        if(count < 5){
                curr = curr -> next;
                count++;
        } else {
                last = curr;
        }
}
```



```
count = 2;

last = NULL;
```

```
while(curr != NULL){
        if(count < 5){
                curr = curr -> next;
                count++;
        } else {
                last = curr;
        }
}
```



GREEN

GREEN

RED

RED

GREEN

RED

NULL

board[0][3].stack,

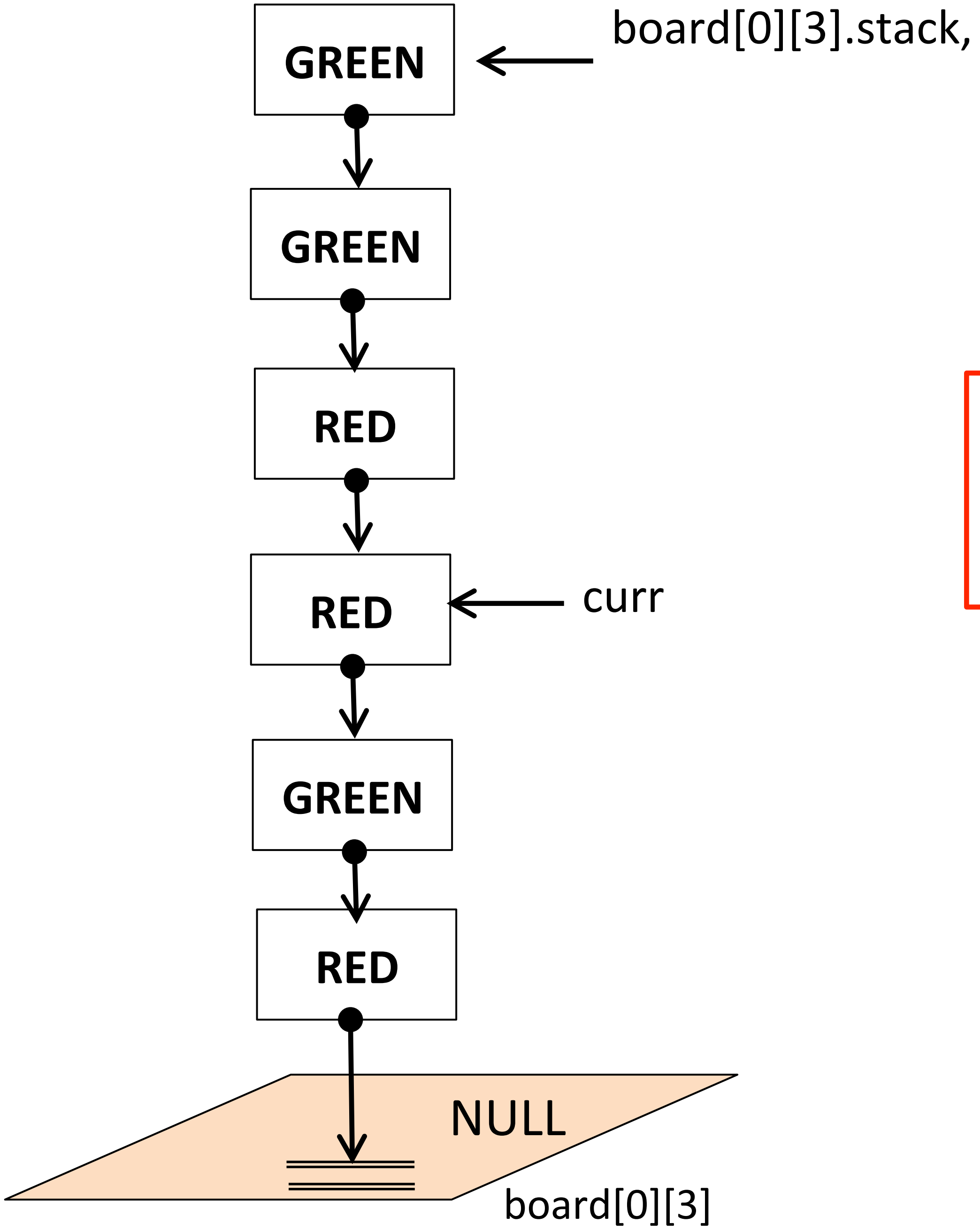
curr

board[0][3]

count = 3;

last = NULL;

```
while(curr != NULL){
        if(count < 5){
                curr = curr -> next;
                count++;
        } else {
                last = curr;
        }
}
```



GREEN

GREEN

RED

RED

GREEN

RED

NULL

board[0][3].stack,

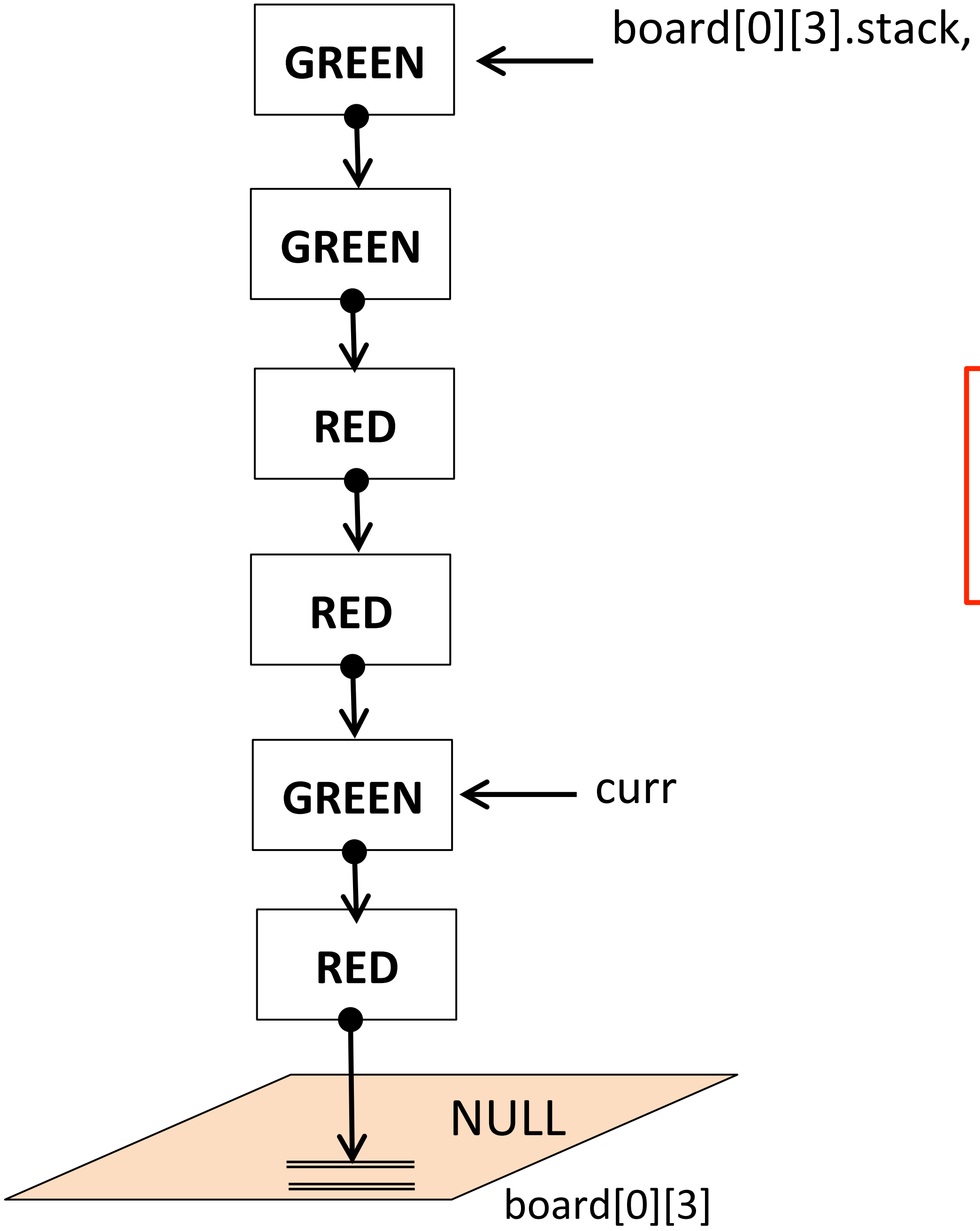
curr

board[0][3]

count = 4;

last = NULL;

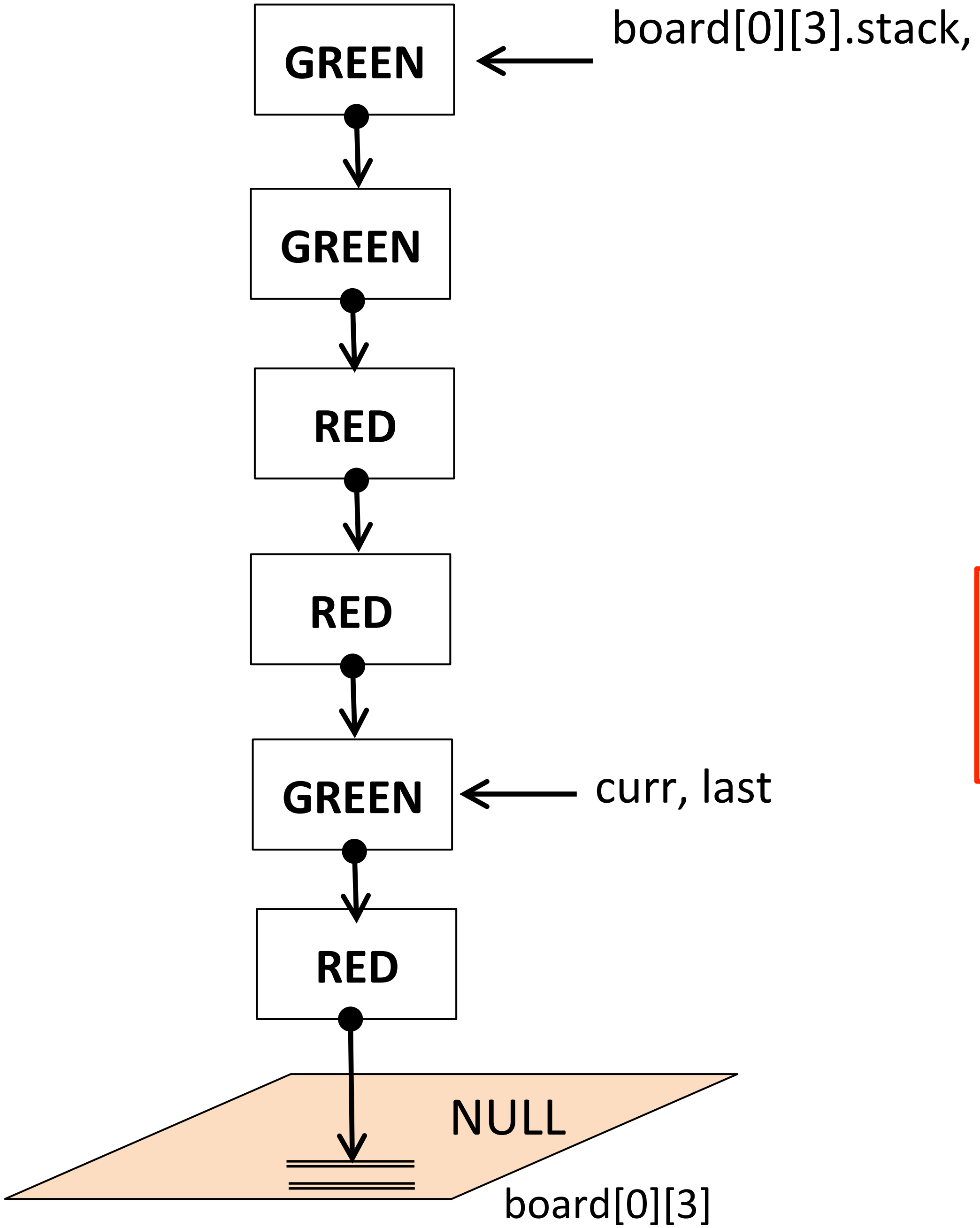
```
while(curr != NULL){
    if(count < 5){
        curr = curr -> next;
        count++;
    } else {
        last = curr;
    }
}
```



count = 5;

last = NULL;

```
while(curr != NULL){
    if(count < 5){
        curr = curr -> next;
        count++;
    } else {
        last = curr;
    }
}
```



GREEN

GREEN

RED

RED

GREEN

RED

NULL

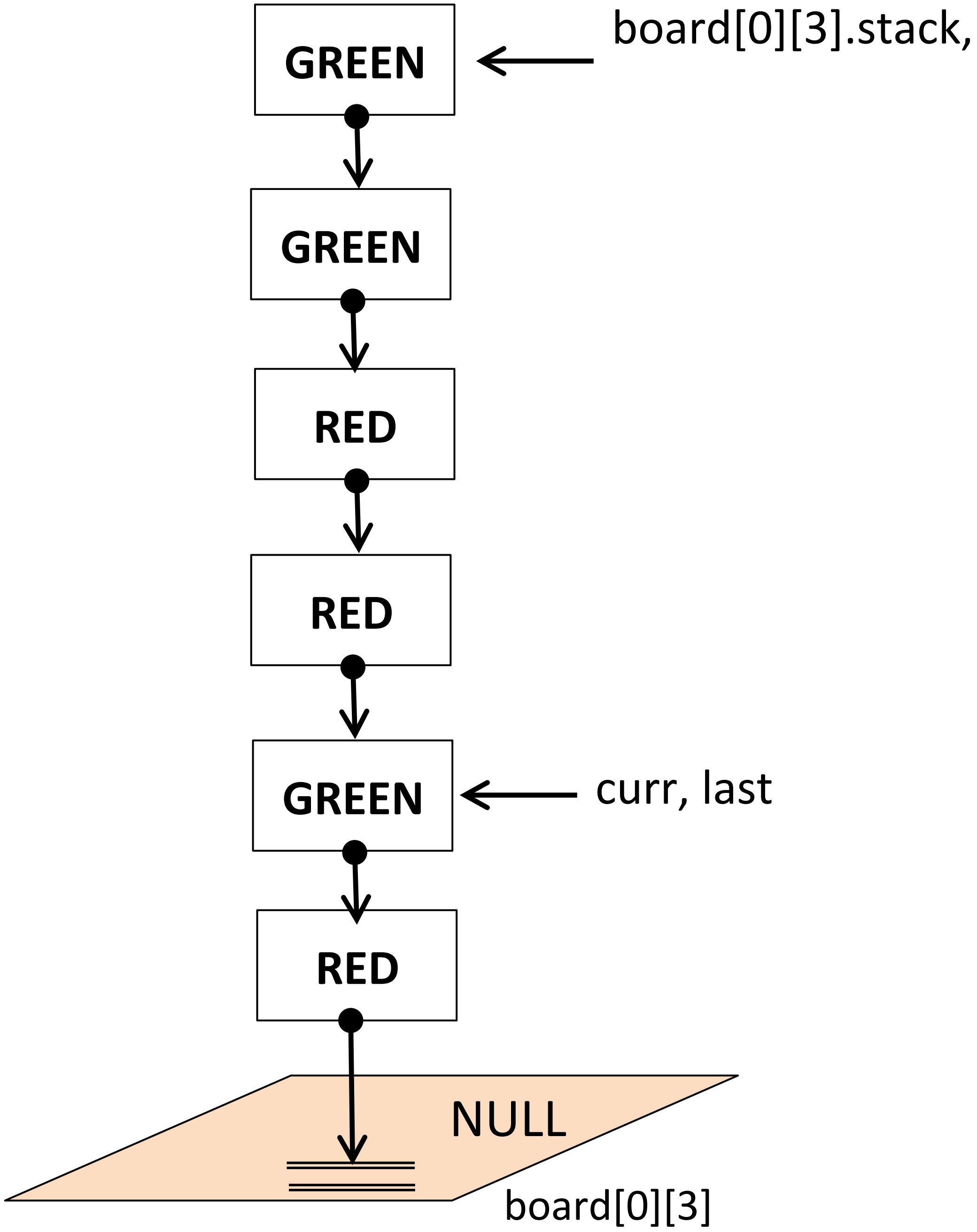
board[0][3].stack,

curr, last

board[0][3]

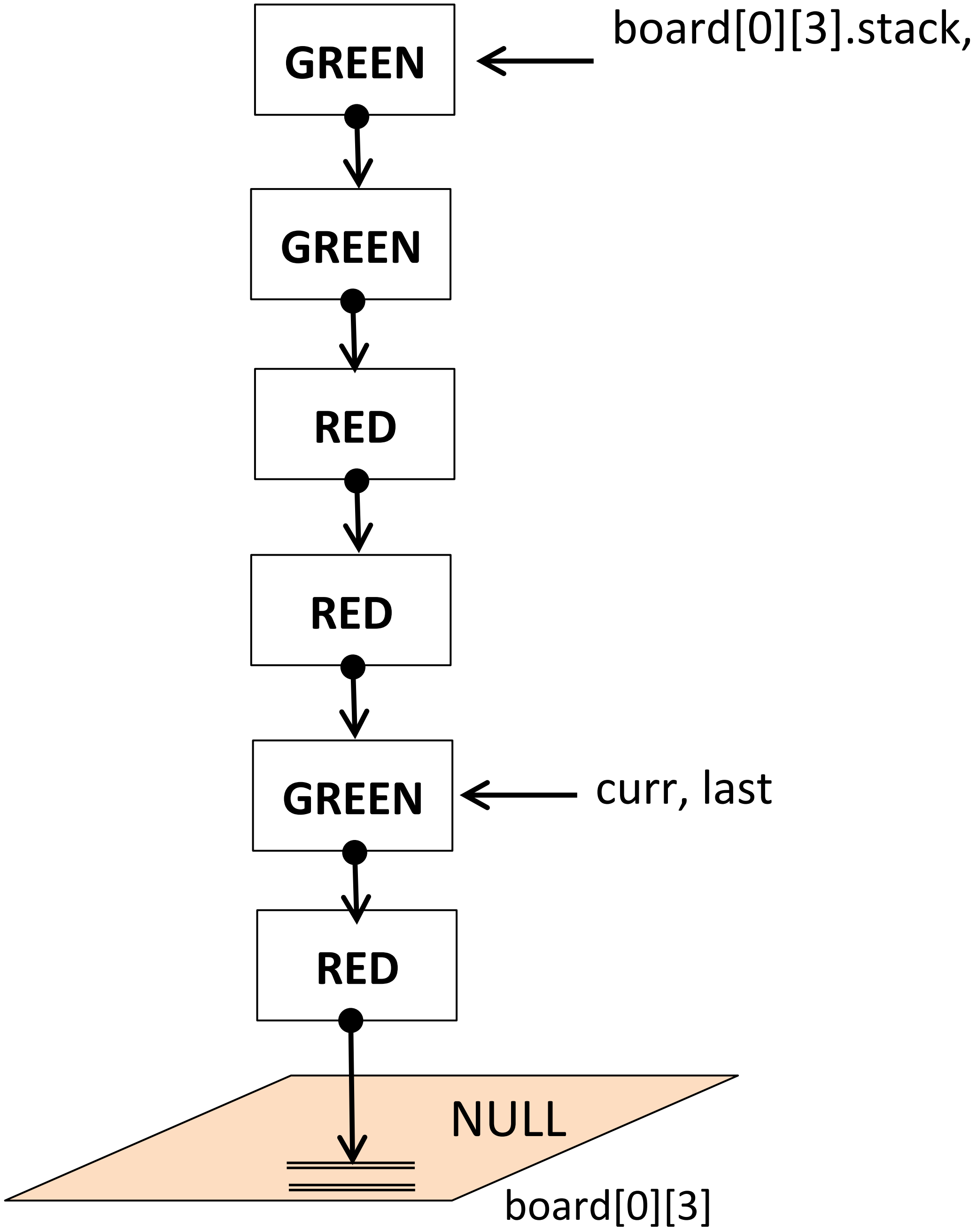
count = 5;

```
while(curr != NULL){
        if(count < 5){
                curr = curr -> next;
                count++;
        } else {
                last = curr;
        }
}
```



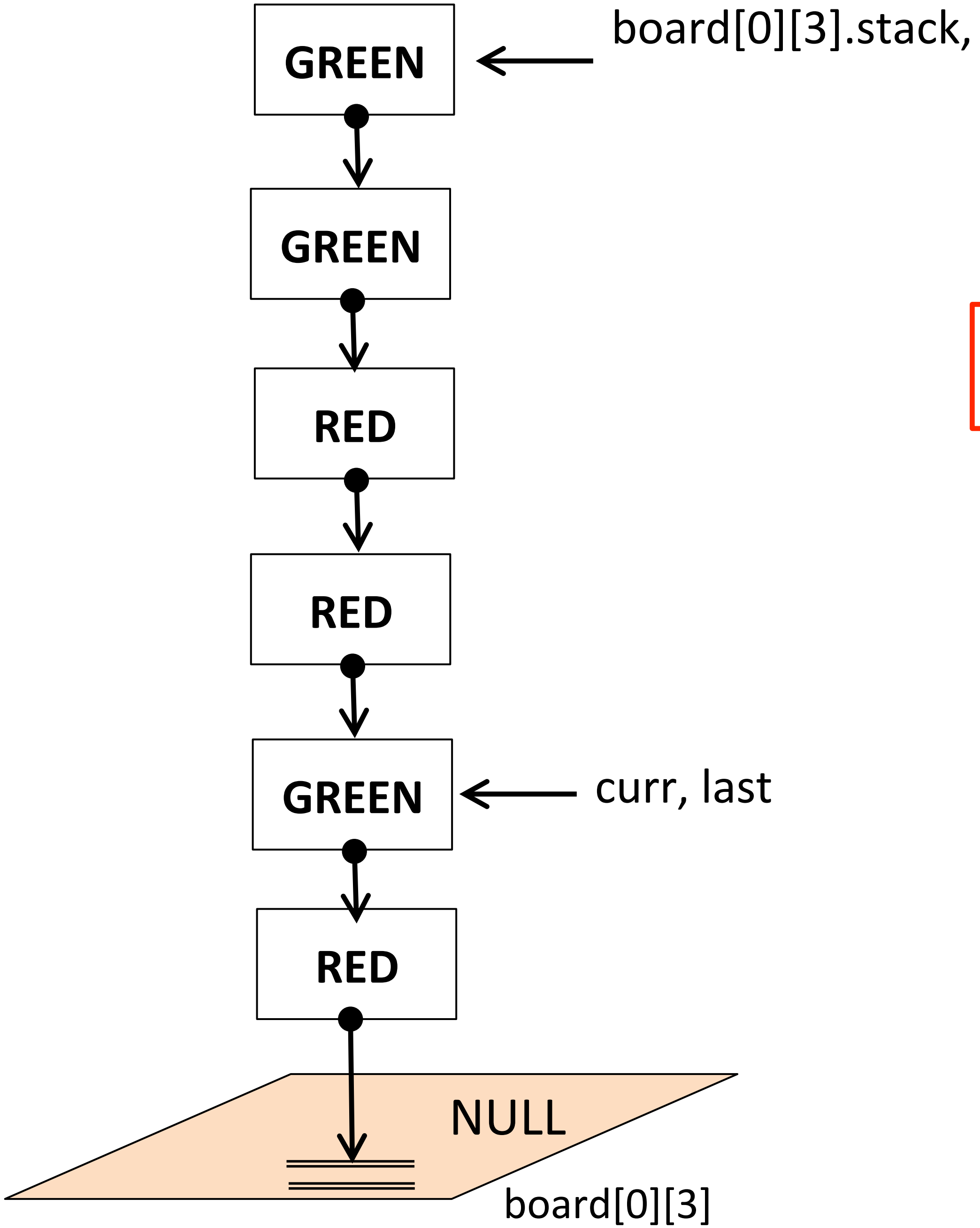
count = 5;

```
while(curr != NULL){
        if(count < 5){
                curr = curr -> next;
                count++;
        } else {
                last = curr;
        }
}
```



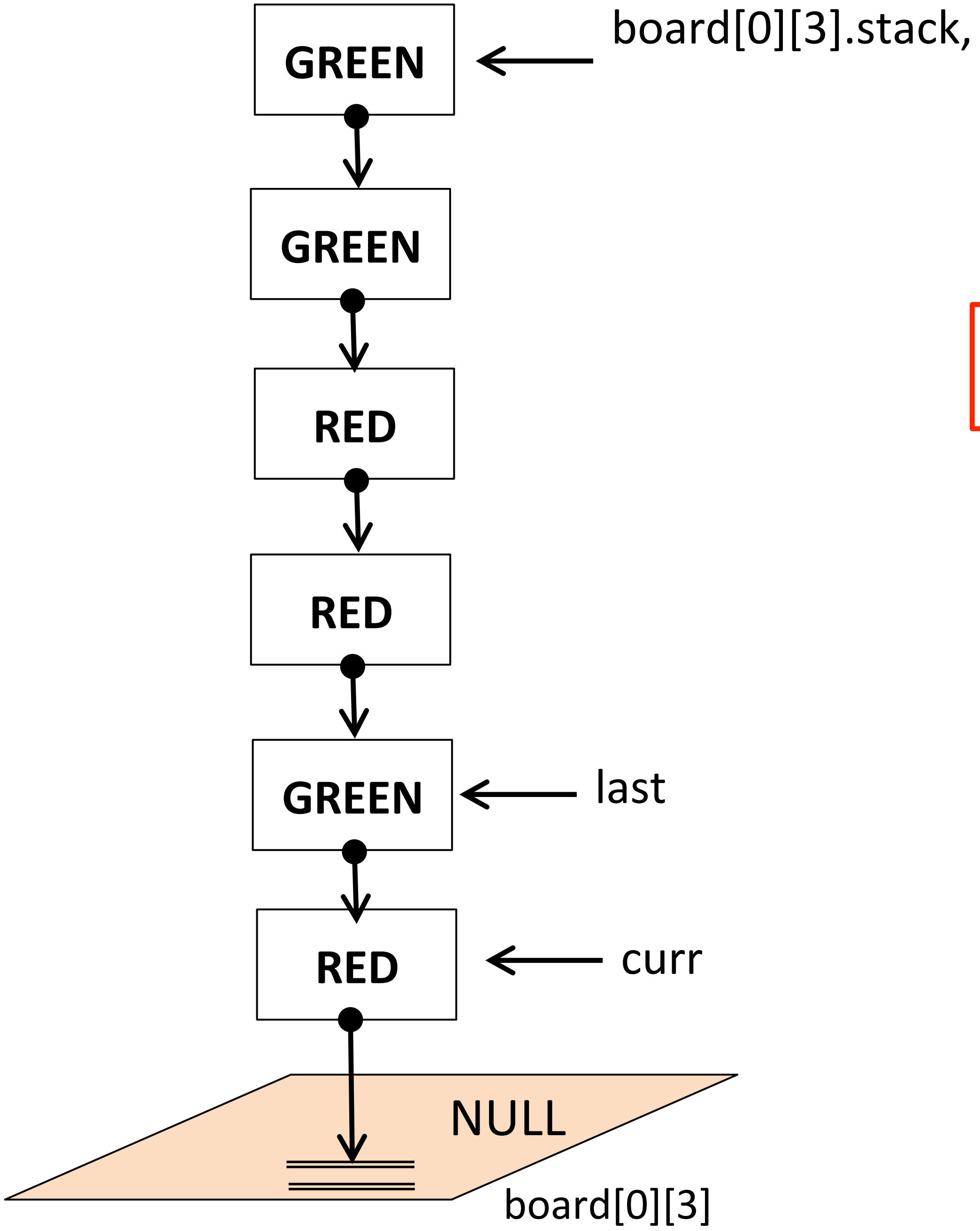
count = 5;

```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;
        }
```



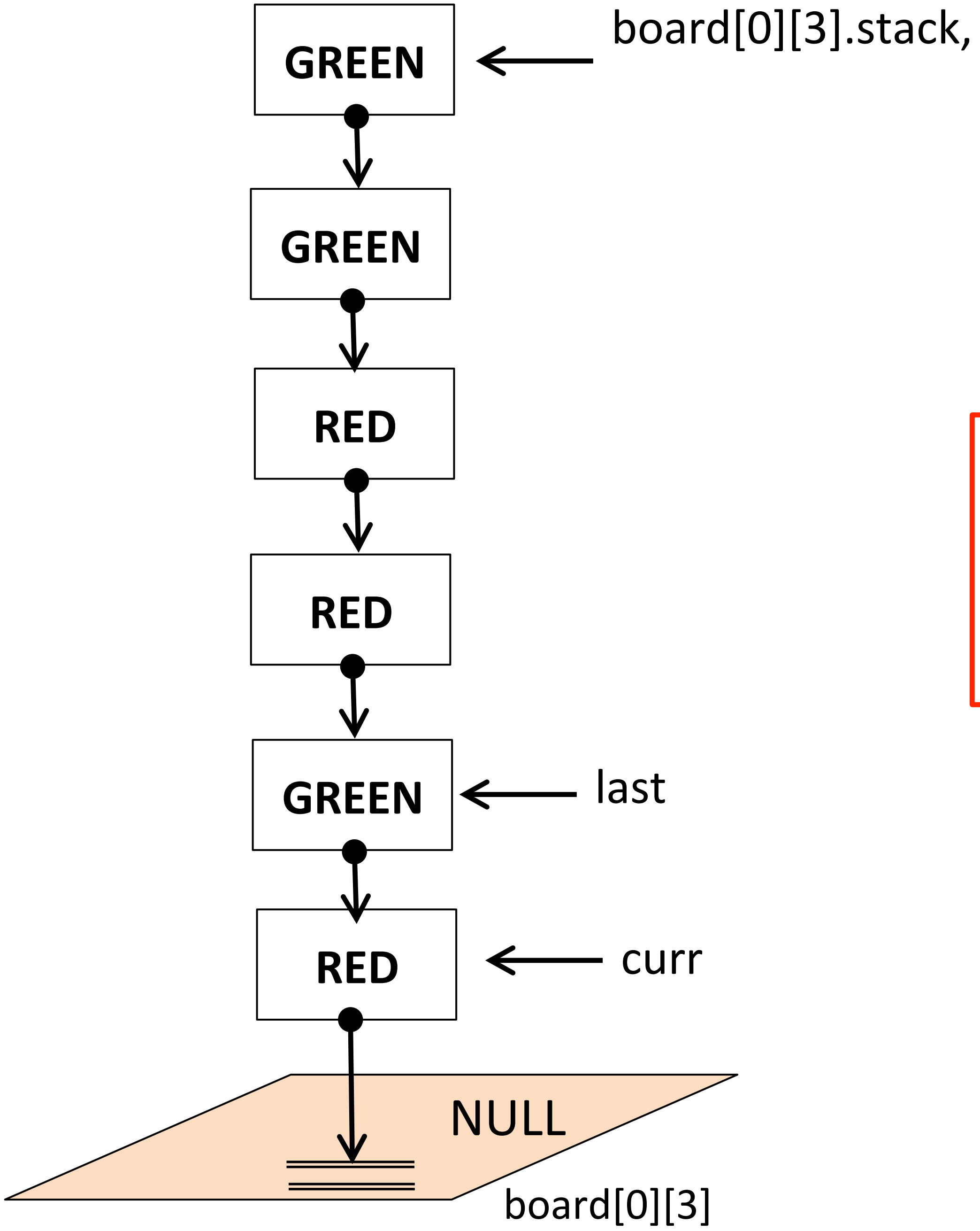
count = 5;

```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```



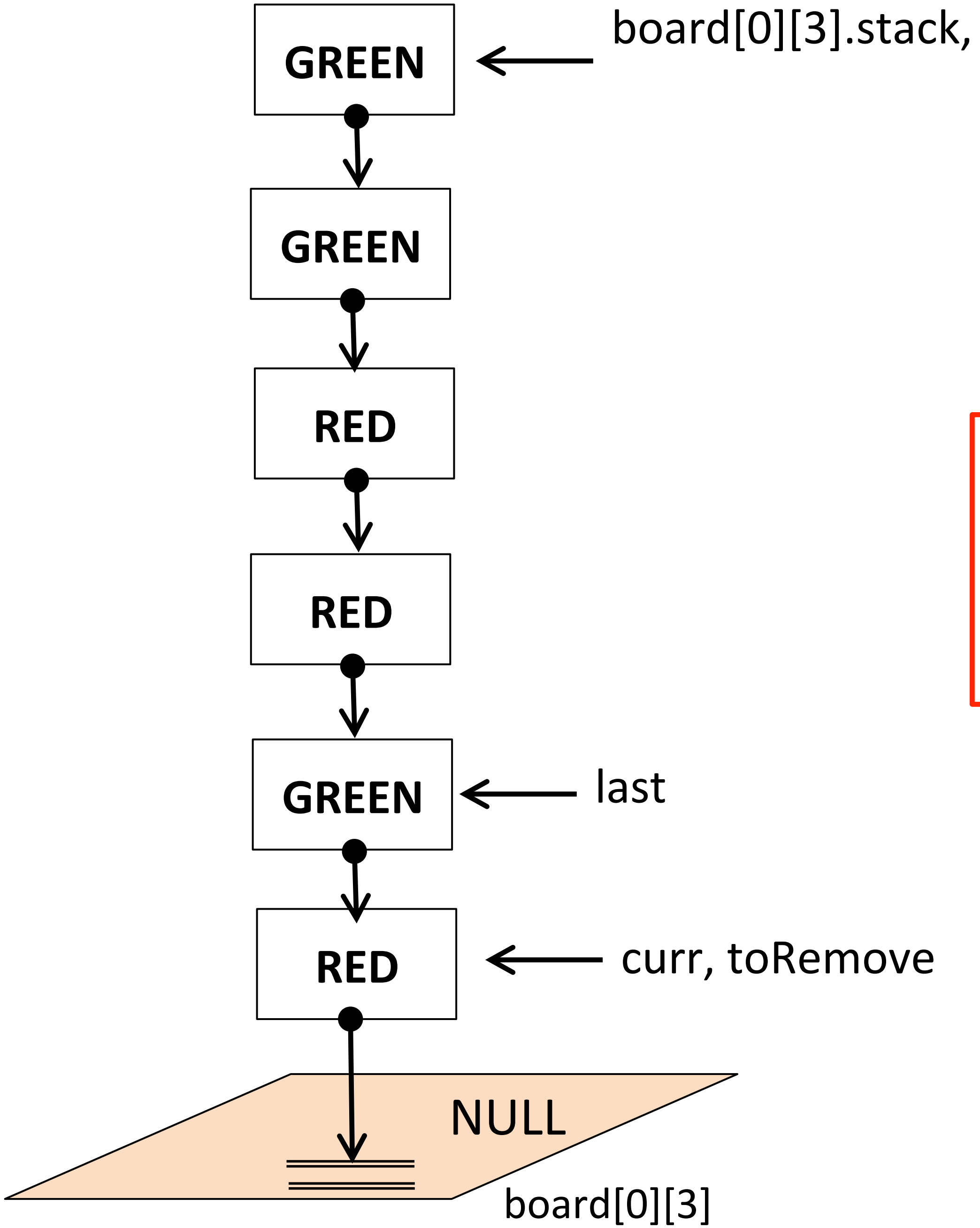
count = 5;

```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```



count = 5;

```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```



GREEN

GREEN

RED

RED

GREEN

RED

NULL

board[0][3].stack,

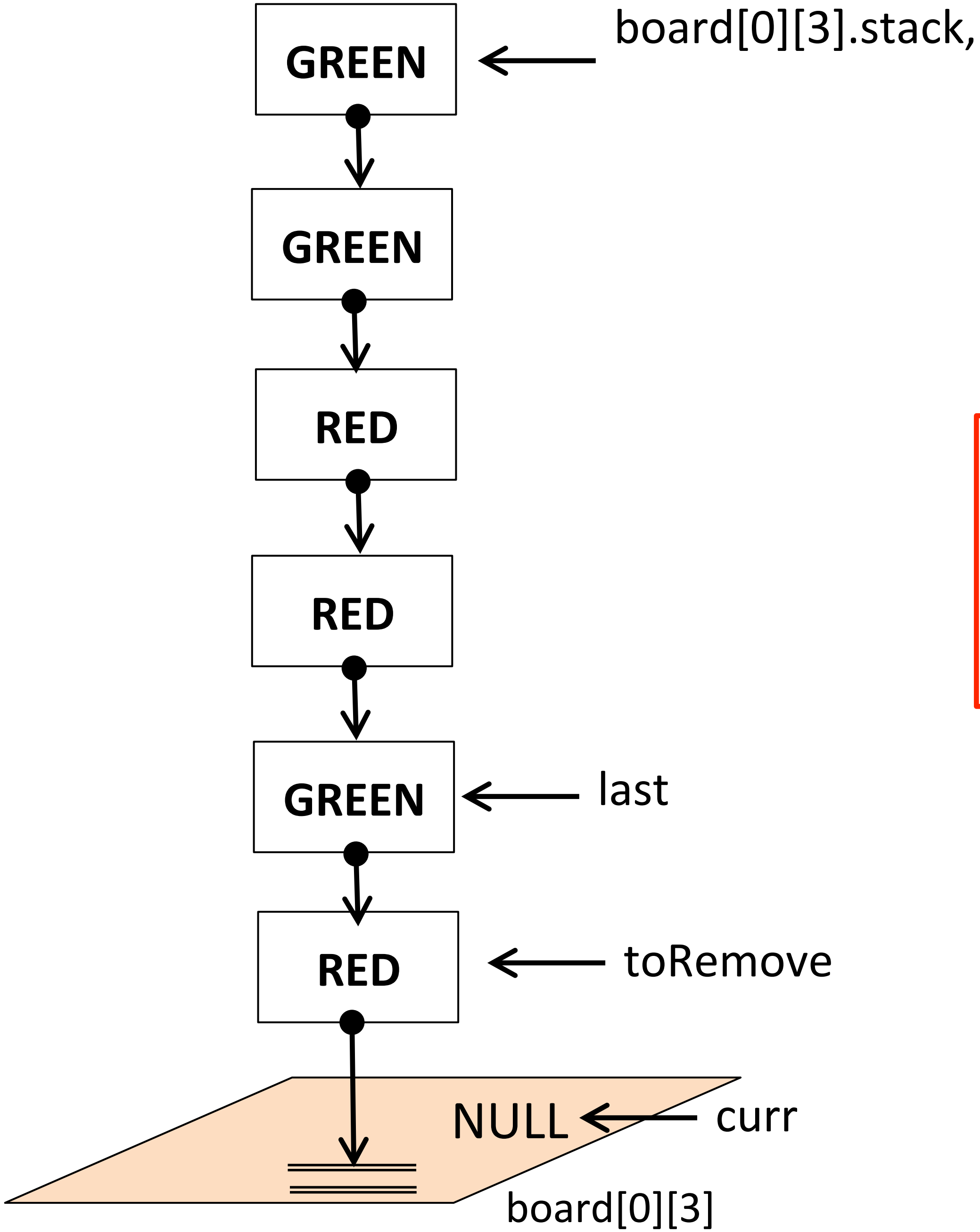
last

curr, toRemove

board[0][3]

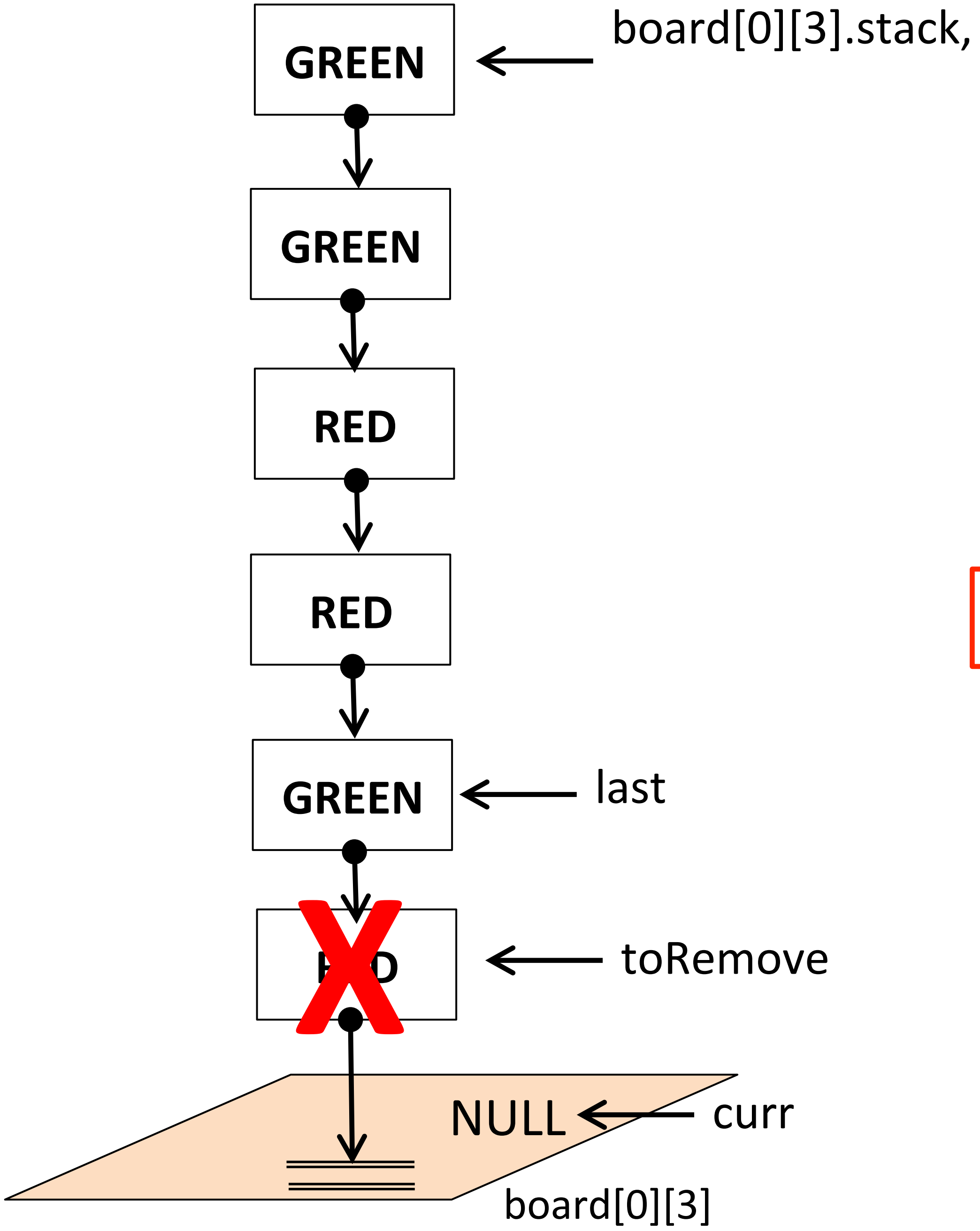
count = 5;

```
if(last != NULL){
    curr = curr->next;
    while(curr!=NULL){
        toRemove = curr;
        curr = curr->next;
        free(toRemove);
        }
    last->next = NULL;

    }
```



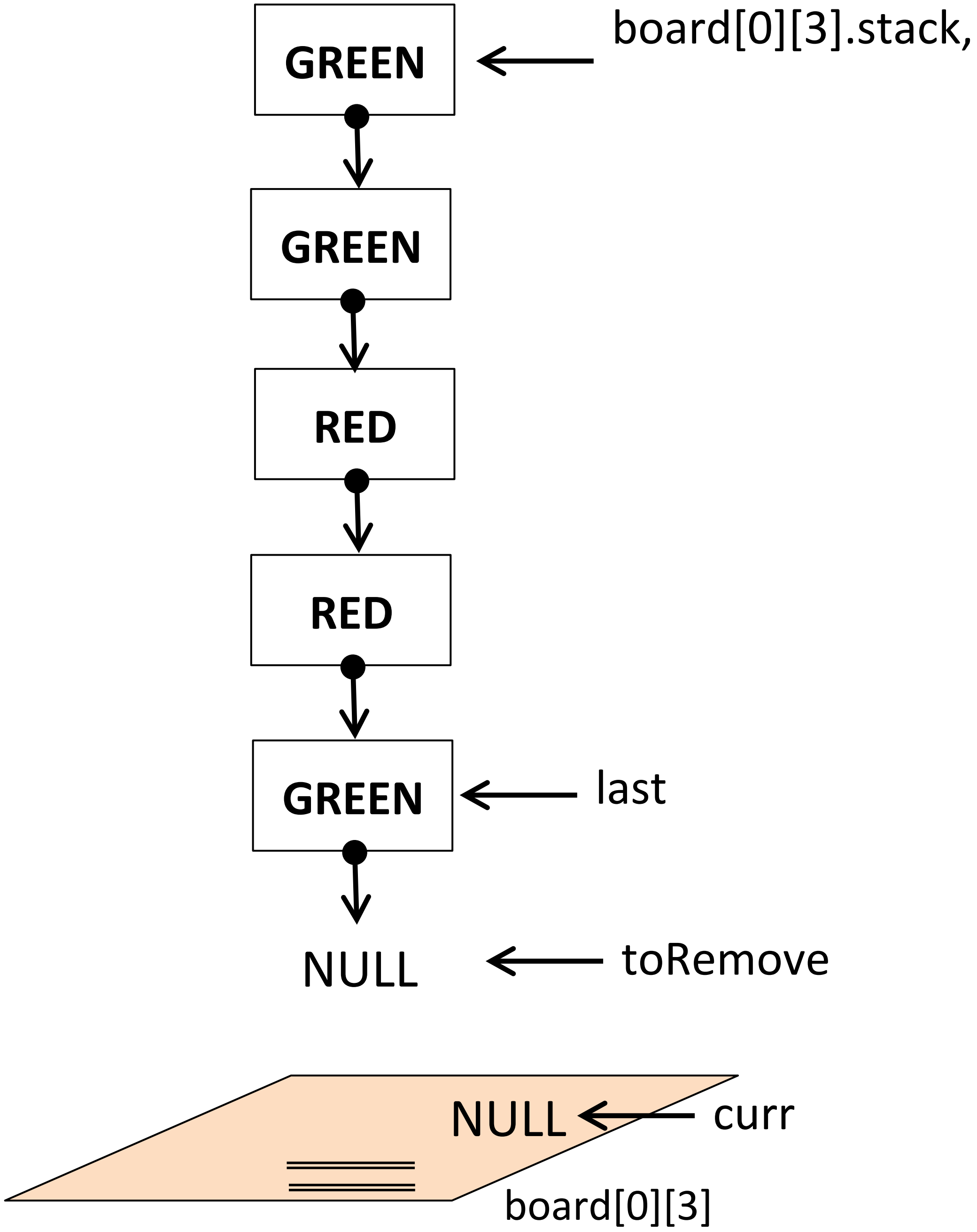
count = 5;

```
if(last != NULL){
    curr = curr->next;
    while(curr!=NULL){
        toRemove = curr;
        curr = curr->next;
        free(toRemove);
        }
    last->next = NULL;

    }
```



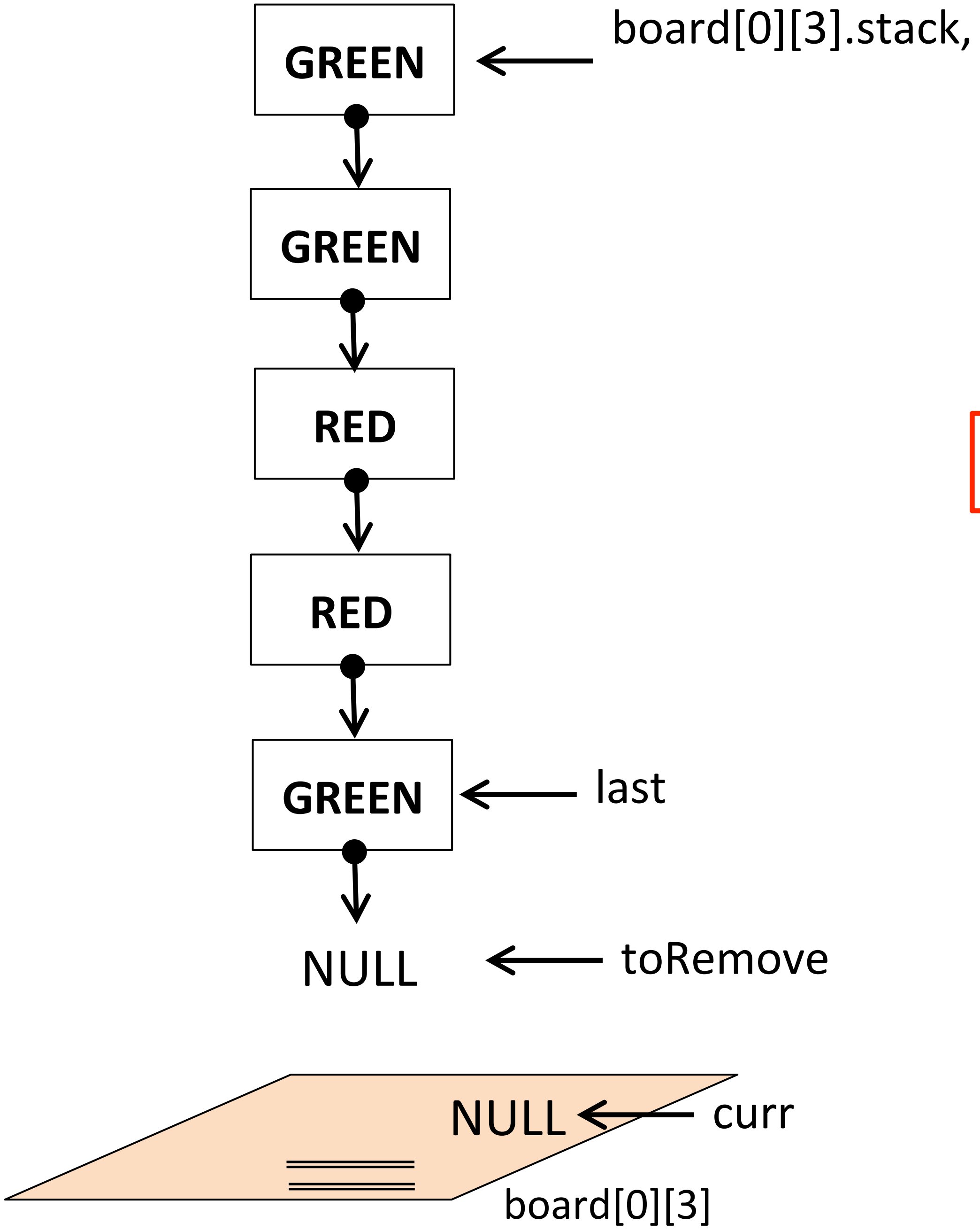
count = 5;

```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```



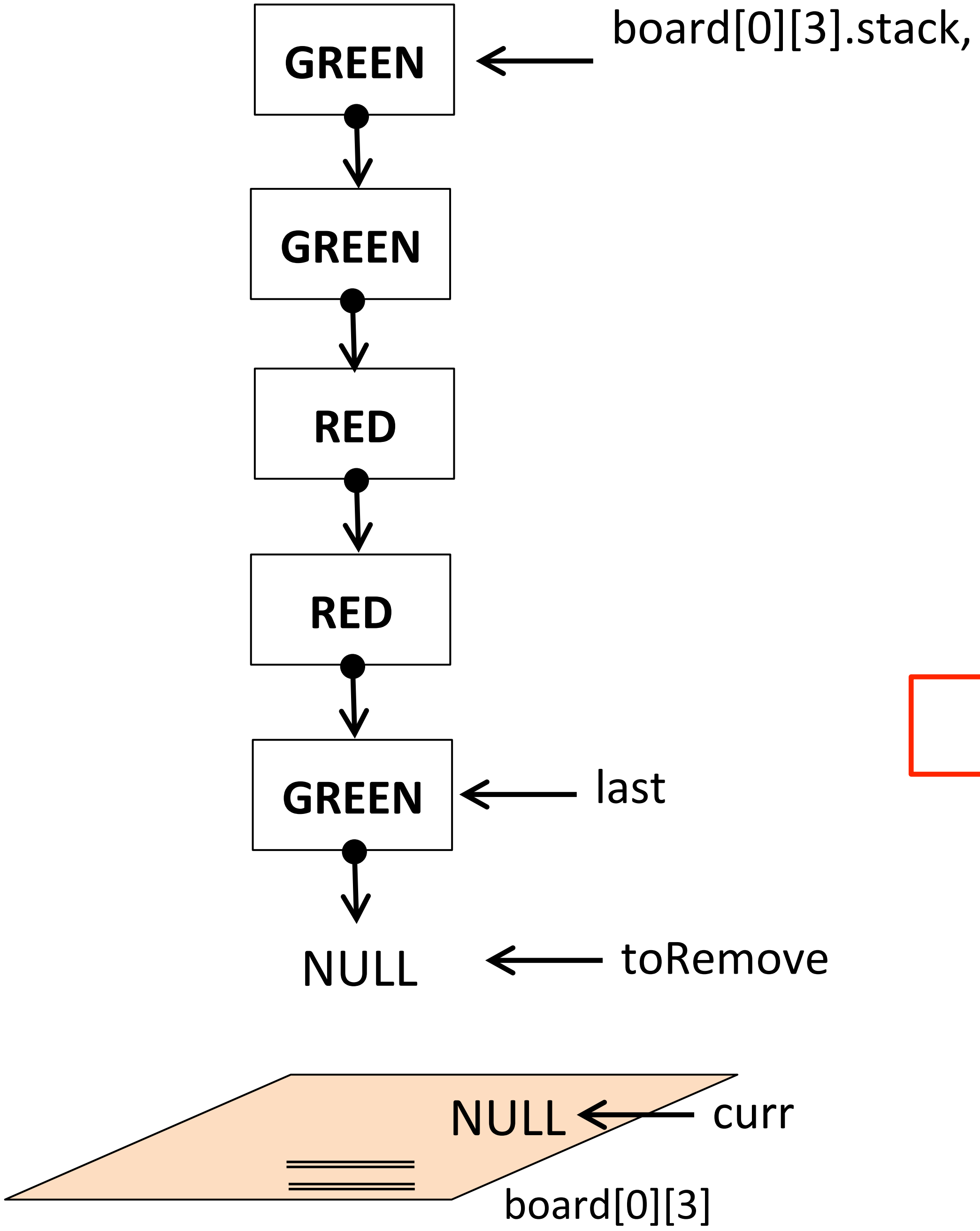
count = 5;

```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```



count = 5;

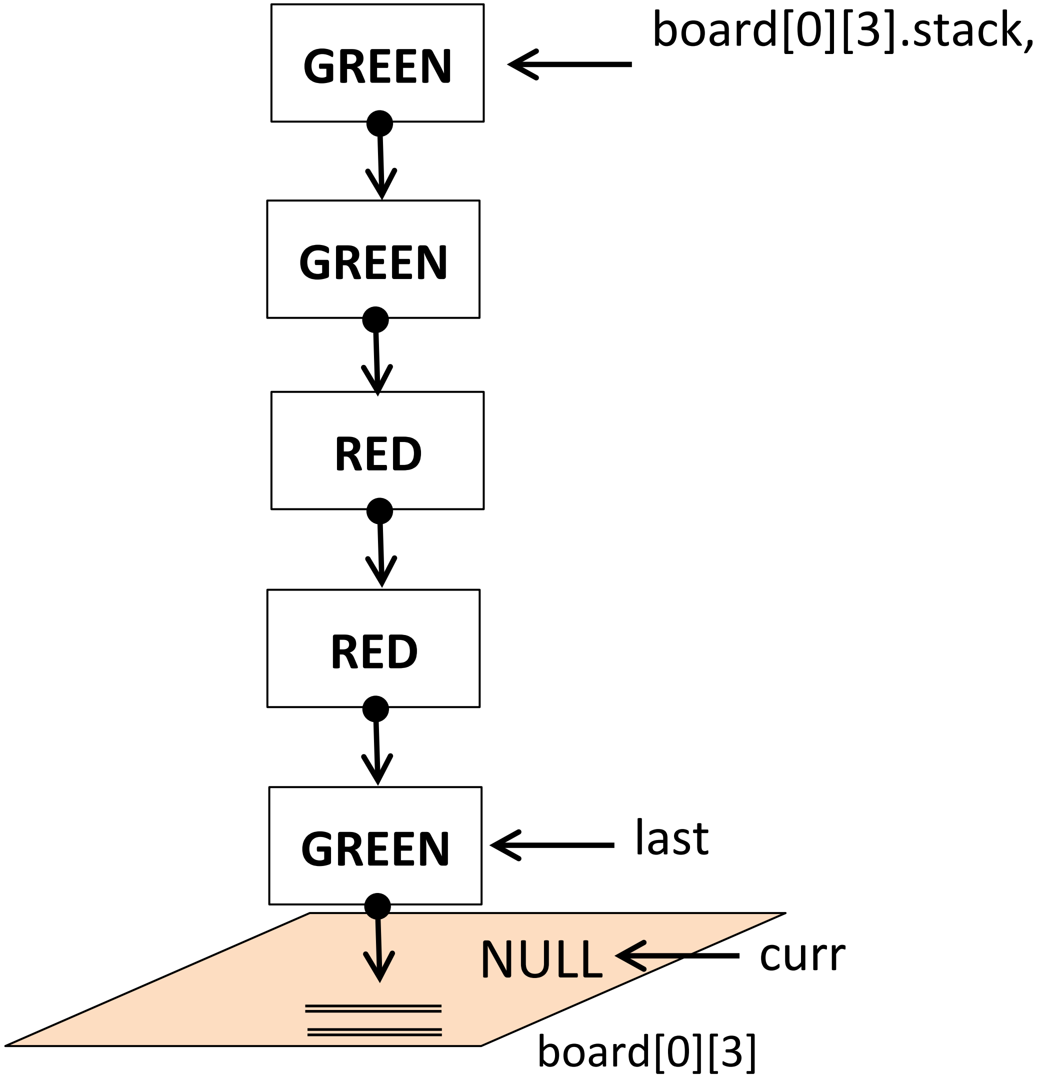
```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```



count = 5;

```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```

count = 5;



```
if(last != NULL){
        curr = curr->next;
        while(curr!=NULL){
                toRemove = curr;
                curr = curr->next;
                free(toRemove);
                }
        last->next = NULL;

        }
```

Recap

- String manipulation in C
- How to merge one stack on top of another in the Focus game
- How to remove pieces in excess from the stack.