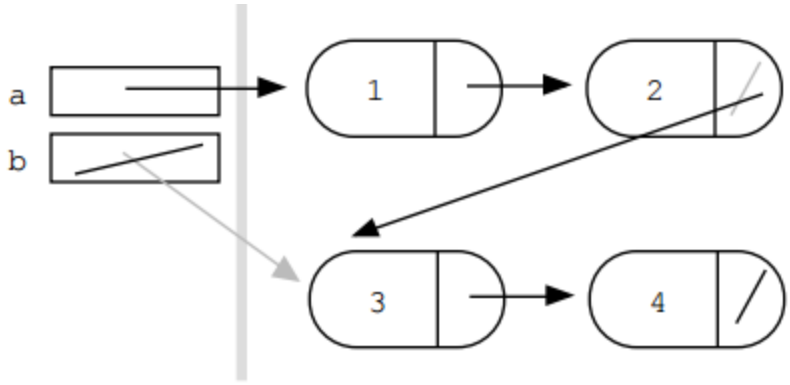
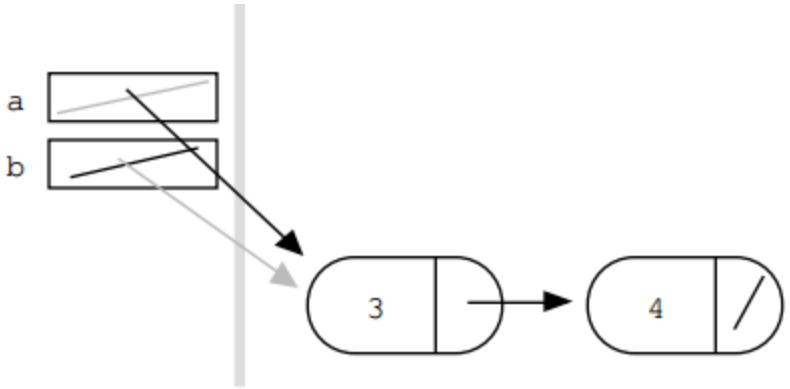
1. Write an Append() function that takes two lists, 'a' and 'b', appends 'b' onto the end of 'a', and then sets 'b' to NULL (since it is now trailing off the end of 'a'). Here is a drawing of a sample call to Append(a, b) with the start state in gray and the end state in black. At the end of the call, the 'a' list is {1, 2, 3, 4}, and 'b' list is empty.



It turns out that both of the head pointers passed to Append(a, b) need to be reference parameters since they both may need to be changed. The second 'b' parameter is always set to NULL. When is 'a' changed? That case occurs when the 'a' list starts out empty. In that case, the 'a' head must be changed from NULL to point to the 'b' list. Before the call 'b' is {3, 4}. After the call, 'a' is {3, 4}.




```
// Append 'b' onto the end of 'a', and then set 'b' to NULL.
void Append(struct node** aRef, struct node** bRef) {
        // Your code...
}
```

You will have to write the code for creating the initial lists as well! (add-first/last) and print function. You cannot use a ***tail*** variable.

2. What is the complexity of deleting an element from a singly linked list? Does having a doubly linked list make any difference? Why?