

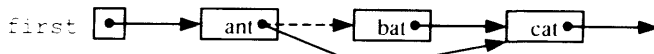


The variable `first` is simply updated to refer to the first node's successor (i.e., the second node). Now, if we follow the links from `first`, we see that the SLL's nodes contain 'bat', 'cat', etc. The node containing 'ant' is no longer part of the SLL. (If nothing else refers to that node, Java will automatically deallocate it.)

The above code fragment assumes that the SLL has at least one node initially. If the SLL has exactly one node, `first.succ` is null and the code fragment makes `first` null. This is correct behavior.

(b) The following code fragment deletes the SLL's second node:

```
SLLNode second = first.succ;
first.succ = second.succ;
```

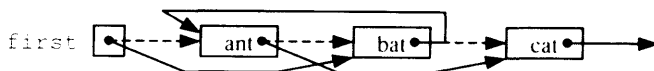


The local variable `second` is made to refer to the second node, then the first node's successor is updated to be the second node's successor (i.e., the third node). Now, if we follow the links from `first`, we see that the SLL's nodes contain 'ant', 'cat', etc. The node containing 'bat' is no longer part of the SLL.

The above code fragment assumes that the SLL has at least two nodes initially. If the SLL has exactly two nodes, `second.succ` is null; the code fragment still works correctly.

(c) The following code fragment swaps the SLL's first and second nodes:

```
SLLNode second = first.succ;
first.succ = second.succ;
second.succ = first;
first = second;
```



The declaration and first statement unlink the second node, as in (b), but the local variable `second` still refers to this node. The remaining two statements link this node back into the SLL, this time as the first node. Now, if we follow the links from `first`, we see that the SLL's nodes contain 'bat', 'ant', 'cat', etc.

The above code fragment assumes that the SLL has at least two nodes initially.

This example has demonstrated how easy it is to restructure an SLL, simply by manipulating links. There is no need to disturb the elements contained in the nodes.

When we restructure a linked list, however, we must take great care to update all the links correctly. Omitting to update even one link would fatally damage the linked list's structure. In practice this is a common programming error.