



**FIGURE 9.19B** The LazyList class: why validation is needed. In part (a), thread A is attempting to remove node *a*. After it reaches the point where  $\text{pred}_A$  refers to  $\text{curr}_A$ , and before it acquires locks on these nodes, the node  $\text{pred}_A$  is logically and physically removed. After A acquires the locks, validation will detect the problem. In part (b), A is attempting to remove node *a*. After it reaches the point where  $\text{pred}_A$  refers to  $\text{curr}_A$ , and before it acquires locks on these nodes, a new node is added between  $\text{pred}_A$  and  $\text{curr}_A$ . After A acquires the locks, even though neither  $\text{pred}_A$  nor  $\text{curr}_A$  is marked, validation detects that  $\text{pred}_A$  is not the same as  $\text{curr}_A$ , and A's call to `remove()` will be restarted.