```
71    boolean remove(T x) {
72      int bottomLevel = 0;
73      Node<T>[] preds = (Node<T>[]) new Node[MAX_LEVEL + 1];
74      Node<T>[] succs = (Node<T>[]) new Node[MAX_LEVEL + 1];
75      Node<T> succ;
76      while (true) {
77        boolean found = find(x, preds, succs);
78        if (!found) {
79          return false;
80        } else {
81          Node<T> nodeToRemove = succs[bottomLevel];
82          for (int level = nodeToRemove.topLevel;
83               level >= bottomLevel+1; level--) {
84            boolean[] marked = {false};
85            succ = nodeToRemove.next[level].get(marked);
86            while (!marked[0]) {
87              nodeToRemove.next[level].compareAndSet(succ, succ, false, true);
88            succ = nodeToRemove.next[level].get(marked);
89            }
90          }
91          boolean[] marked = {false};
92          succ = nodeToRemove.next[bottomLevel].get(marked);
93          while (true) {
94            boolean iMarkedIt =
95              nodeToRemove.next[bottomLevel].compareAndSet(succ, succ,
96                                                           false, true);
97            succ = succs[bottomLevel].next[bottomLevel].get(marked);
98            if (iMarkedIt) {
99               find(x, preds, succs);
100              return true;
101           }
102           else if (marked[0]) return false;
103         }
104       }
105     }
106   }
```

FIGURE 14.12 The LockFreeSkipList class: the remove() method.