```
59  bool add(int key) {
60    mm->op_begin();
61    Node<T> *prev, *curr, *next;
62    while (true) {
63      if (find(key, prev, curr, next, mm)) {
64        mm->op_end();
65        return false;
66      }
67      Node *new_node = new Node(key, curr);
68      if (prev->next.compare_exchange_strong(curr, new_node)) {
69        mm->op_end();
70        return true;
71      }
72      mm->unreserve(prev); mm->unreserve(curr);
73    }
74  }
75  bool contains(int key, MemManager* mm) {
76    mm->op_begin();
77    Node<T> *prev, *curr, *next;
78    bool ans = find(key, prev, curr, next, mm);
79    mm->op_end();
80    return ans;
81  }
82  bool remove(int key, MemManager* mm) {
83    mm->op_begin();
84    Node<T> *prev, *curr, *next;
85    while (true) {
86      if (!find(key, prev, curr, next, mm)) {
87        mm->op_end();
88        return false;
89      }
90      if (!curr->next.compare_exchange_strong(next, mark(next))) {
91        mm->unreserve(prev); mm->unreserve(curr);
92        continue;
93      }
94      if (prev->next.compare_exchange_strong(curr, next)) {
95        mm->sched_for_reclaim(curr);
96      } else {
97        mm->unreserve(prev); mm->unreserve(curr);
98        find(key, prev, curr, next, mm);
99      }
100     mm->op_end();
101     return true;
102   }
103 }
```

FIGURE 19.5 Public methods of the C++ nonblocking list-based set.