```
int32_t remove_subtree (node_t** rootp, int N);
{
  int32_t count;
  if (NULL == *rootp) { return 0; }
  count = remove_subtree (&(*rootp)->left, N);
  count += remove_subtree (&(*rootp)->right, N);
  if (N == (*rootp)->value) {
    free (*rootp);
    *rootp = NULL;
    count++;
  }
  return count;
}
```

WARNING: This version doesn't work!

(Freeing our node requires freeing both subtrees. We could add calls to remove_all inside the if (N==…) statement body, but the other version of the code on the next slide is fine.)