Task 1

|  | Unsorted, singly linked | Sorted, singly linked | Unsorted, doubly linked | Sorted, doubly linked |
| --- | --- | --- | --- | --- |
| Search(L, k) |  |  |  |  |
| Insert(L, x) |  |  |  |  |
| Delete(L, x) |  |  |  |  |
| Successor(L, x) |  |  |  |  |
| Predecessor(L, x) |  |  |  |  |
| Minimum(L) |  |  |  |  |
| Maximum(L) |  |  |  |  |

Task 2  
  
We need to update the line 5 of TREE-DELETE, such that *y* is equal to TREE-MAXIMUM(z.left), and lines 6-12 so that every y.left and z.left is replaced with y.right and z.right and vice versa.  
To implement the fair strategy, each time calling TREE-DELETE, we would randomly decide whether or not to use the predecessor or successor.  
  
Task 3  
  
c) could not be the sequence of nodes examined because we take the left child from the 911 node, and yet somehow manage to get to the 912 node which cannot belong to the left subtree because it is greater.  
e) is also impossible because we take the right subtree on the 347 node and yet later come across the 299 node.