11.2-3

Successful Search: $O(n)$ because traversing even a sorted linked list is $O(n)$

Unsuccessful Search: $O(n)$ because exhaustive search even through a sorted linked list will take worst-case $O(n)$

Insert: $O(n)$ because if linked lists are sorted descending and the new value is less than all existing values, it takes $O(n)$ time to insert it at the tail

Delete: $O(n)$ because in the worst case all n elements hash to the same slot and you may have to traverse all n of them to get to the list's tail

11.3-1

Because comparing long character strings can be an expensive operation, first calculate the hash (search key) and compare that value to the hash value contained in each node. Single integer-to-integer comparisons are faster than long string comparisons, so it will be faster to find a match this way. When an equivalent hash value is found, you can then use string comparison to verify that the strings are truly equal and not just a hash collision.