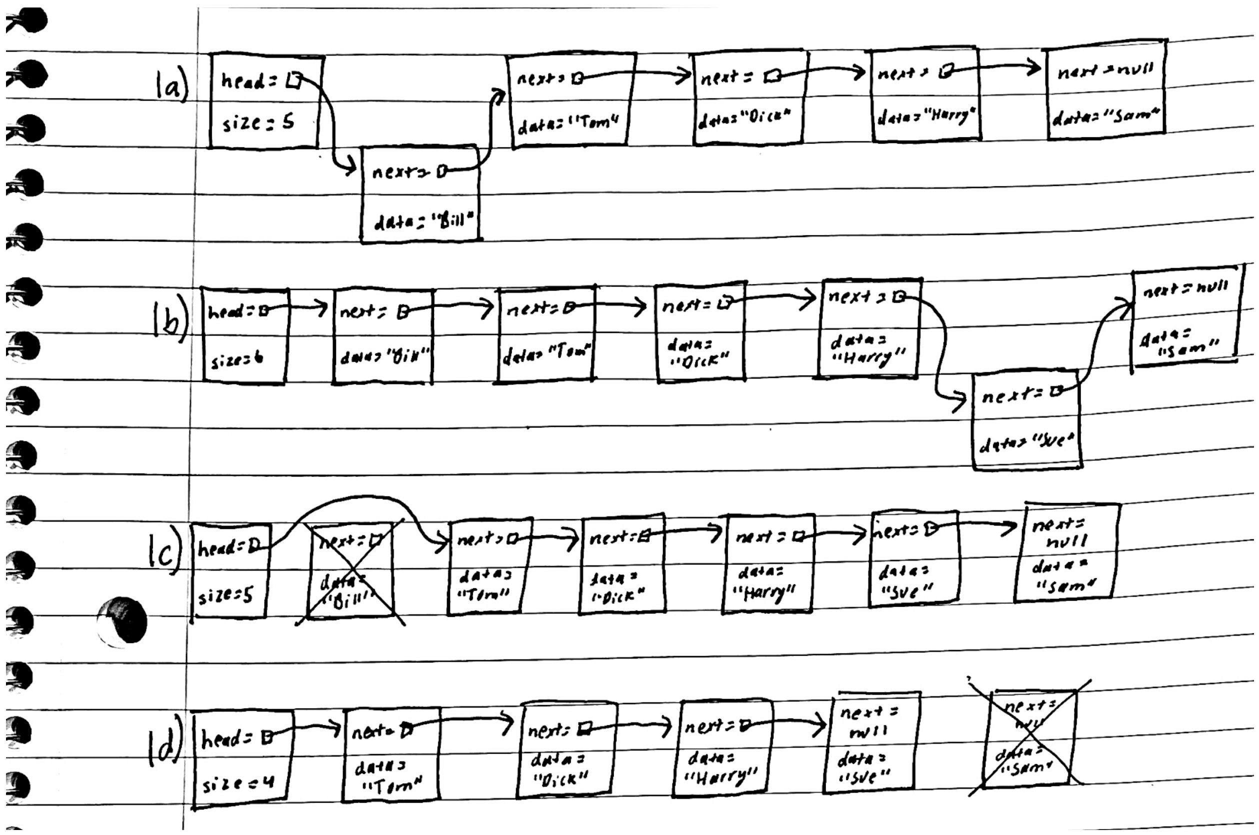
Lab 5

Draw Insertion/Removal

a. Insert “Bill” before “Tom”
b. Insert “Sue” before “Sam”
c. Remove “Bill”
d. Remove “Sam”

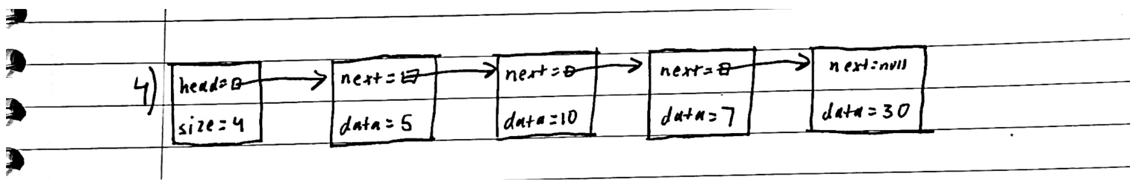


Self-Check 1
Big-O of get: O(n)

Self-Check 2
Big-O of set: O(n)

Self-Check 3
Big-O of add: O(1)

Self-Check 4



```
int sum = 0;
Node<Integer> nodeRef = head;
while(nodeRef != null) {
        int next = nodeRef.data;
        sum += next;
        nodeRef = nodeRef.next;
}
```

Self-Check 5

a. “Shakira” is now the first element in the list and points to “Tom.” Head now points to “Shakira.” This code inserted “Shakira” at the beginning of the list.
b. nodeRef is set to the element that the next of head points to which is “Tom.” The next of “Tom” is set to point to “Harry” now instead of “Dick.” This code removed “Dick” from the list.
c. nodeRef is set to the head which is “Shakira.” While the next of nodeRef is not null, set nodeRef to the next node. This traverses the linked list until the node whose next is null is found. This node is “Sam.” Make “Sam” point to a new node, “Tamika,” instead. This code inserted “Tamika” at the end of the list.
d. nodeRef is set to the head which is “Shakira.” While nodeRef is not equal to null and the data in nodeRef is not equal to “Harry,” traverse through the list. This will stop traversing when either “Harry” is found or the nodeRef is null. Once the list has stopped being traversed, check to see if nodeRef is null. If nodeRef is not null, meaning nodeRef is “Harry”, then the data in nodeRef is set to “Sally” instead. The next of “Sally” is set to point to a new node that contains “Harry”. This new node points to “Sam.” This code inserted “Sally” between “Tom” and “Harry.”