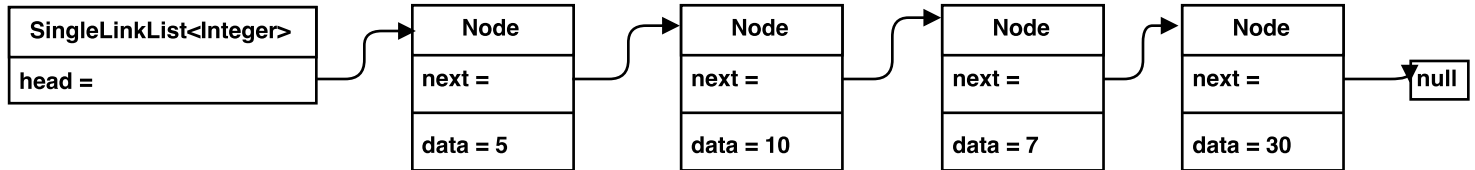


1) The big-O for the Single-Link get operation is ==> O(n)

2) The big-O for the set operation is ==> O(1)

3) The big-O for the search operation is ==> O(n)

4) Single-linked list containing the integers 5, 10, 7, and 30 and referenced by the head:



```

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

5) Explain the effect in the fragments:

a) `head = new Node<>("Shakira", head.next);`
Adds a new node to the beginning of the list.

b) `Node<String> nodeRef = head.next;`
`nodeRef.next = nodeRef.next.next;`
Adds a new node to the beginning of the list.
Then, removes the next two nodes.

c) `Node<String> nodeRef = head;`
`while (nodeRef.next != null)`
`nodeRef = nodeRef.next;`
`nodeRef.next = new Node<>("Tamika");`
Traverses the list and adds a new node to the end of the list.

d) `Node<String> nodeRef = head;`
`while (nodeRef != null && !nodeRef.data.equals("Harry"))`
`nodeRef = nodeRef.next;`
`if (nodeRef != null) {`
`nodeRef.data = "Sally";`
`nodeRef.next = new Node<>("Harry", nodeRef.next.next);`
`}`
Searches for **Harry**.
If **Harry** is in the list, **Sally** will replace him, **Harry** will be moved to the next node and the node after **Harry** will be removed.