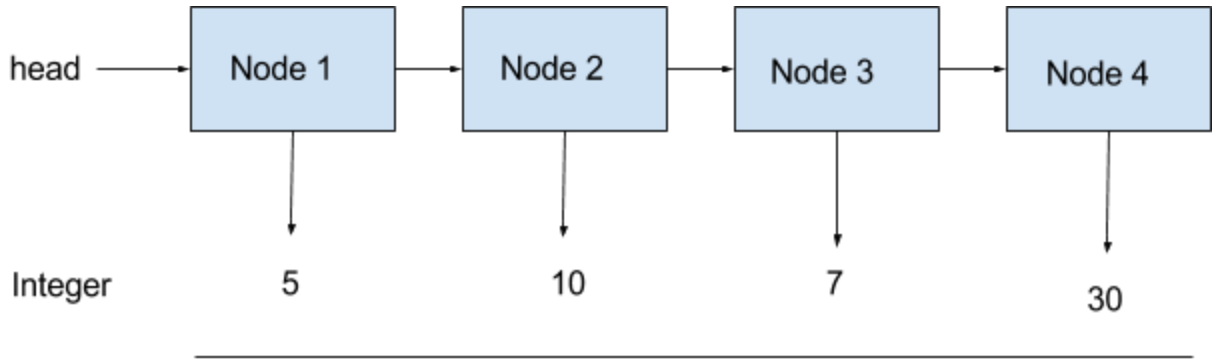
1) O(n)
2) O(n)
3) O(n)
4)



a)

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

5)

a) Replaces the first node in the list
b) Deletes the 3rd node in the list
c) Adds a new node Tamika at the end of the list (NullPointerException if list is empty)
d) Replaces the data in a node with the String “Harry” with the String “Sally”. Then it replaces the node after sally with one holding the String “Harry”

Programming

2)

a)
```
Node<String> currentNode = head;
Node<String> previousNode = null;
Boolean finished = false;
while(currentNode != tail && !finished)
{
        previousNode = currentNode;
        currentNode = currentNode.next;
        if(currentNode.data.equals("Tom"))
        {
                if(previousNode != null)
                {
                        previousNode.next = new Node<String>("Bill",currentNode);
                }
                Else
                {
                        head.next = new Node<String>("Bill",currentNode);
                }
        }
}
```

b)
```
Node<String> currentNode = head;
Node<String> previousNode = null;
Boolean finished = false;
while(currentNode != tail && !finished)
{
        previousNode = currentNode;
        currentNode = currentNode.next;
        if(currentNode.data.equals("Sam"))
        {
                if(previousNode != null)
                {
                        previousNode.next = new Node<String>("Sue",currentNode);
                }
                Else
                {
                        head.next = new Node<String>("Sue",currentNode);
                }
        }
}
```

c)
```
Node<String> currentNode = head;
Node<String> previousNode = null;
Boolean finished = false;
while(currentNode != tail && !finished)
{
        previousNode = currentNode;
        currentNode = currentNode.next;
        if(currentNode.data.equals("Bill"))
        {
                if(previousNode != null)
                {
                        previousNode.next = currentNode.next
                }
                Else
                {
                        head = head.next
                }
        }
}
```

d)
```
Node<String> currentNode = head;
Node<String> previousNode = null;
Boolean finished = false;
while(currentNode != tail && !finished)
{
        previousNode = currentNode;
        currentNode = currentNode.next;
        if(currentNode.data.equals("Sam"))
        {
                if(previousNode != null)
                {
                        previousNode.next = currentNode.next
                }
                Else
                {
                        head = head.next
                }
        }
}
```