5a) Pseudo - Code :

```
class LinkedList {
    Node top
    Node prevNode
    Node curr-Node

    boolean search (int k) {
        curr-Node = top
        prevNode = null
        boolean found = false

        while curr-Node is not null and found is false {
            if curr-Node.data == k
                found = true

            else if curr-Node data < k and curr-Node.link[1] != null
                prevNode = curr-Node
                curr-Node = curr-Node.link[1]

            else
                if prevNode == null
                    curr-Node = null
                else
                    curr-Node = prevNode.link[0]

        return found
    }
```

5b) $\frac{n}{2} - 1$ (How many nodes are strictly)

Justification:

The total number of nodes is represented by 'n'. In the fast lane there are a total of $n/2$ nodes.

The nodes strictly between 2 adjacent nodes in the fast lanes are the nodes that are only in the slow lane.

Therefore, the total number of nodes strictly in the slow lane, would be one less than those in the fast lane; hence $\frac{n}{2} - 1$

$n/2 + 1$ (worst case)

Justification: The worst case in scenario is if the int 'k' is in the last node in the slow lane. Then we would have to traverse the entire fast lane which is $n/2$ and then an extra node in the slow lane to give $\frac{n}{2} + 1$.

5c) $n - \sqrt{n}$

Justification:
The number of slow nodes between 2 adjacent nodes is $\sqrt{n}$.
The total number of slow nodes between all adjacent nodes is $(\sqrt{n} - 1)(\sqrt{n})$. This will give $n - \sqrt{n}$

$2\sqrt{n}$ → worst case scenario
Justification: The worst case is when the int k is in the slow node that comes immediately after the second to the last node in the fast lane.

In this case, to find int k, we would iterate through all the nodes in the fast lane which gives $\sqrt{n}$. Then we would have to check all the slow nodes between the second to the last fast lane node and the last fast lane node and in terms of n that is $\sqrt{n}$. Adding both together gives $2\sqrt{n}$.