Problem 3

1. This solution achieves O(logN) time because it uses binary search to search the data structure. It divides it into halves and then keeps dividing it into halves until the middle element is the value we are searching for. This makes it O(logN).

Code for Linked List:

**int** binarySearch(Node a, **int** value){

getMiddle(a);

**if**(a.data == value){

**return** value;

}

**while**(a.data != value){

**if**(value < a.data){

getMiddle();

}

**if**(value > a[mid]){

low =mid +1;

}

mid = (low+high)/2;

}

**return** mid;

}

Node getMiddle(Node a){

Node one;

Node two;

one = first;

two = first.next;

**while**(two.next !=**null**){

two = two.next;

one = one.next;

}

**return** one;

}

Although I was not able to properly finish the algorithm code for this structure, I know how to achieve this in O(logn) time. The best way is to the the middle node of the data structure which uses a linked list, and then continuing to find the middle node until we land on the node that we are searching for. This is binary search, which is a divide algorithm.