```java
public class main {

    public static void main(String[] args) {
        Data object = new Data();
        object.add(1);
        object.add(5);
        object.add(8);
        object.add(10);
        object.add(12);
        object.add(14);
        object.add(18);
        object.add(20);
        object.add(33);
        object.add(41);


    }
    public static class ProblemThree{
        public int key;
        public int root;
        public class Node{
            Node data;
        }
        public int search(int root, int key){
            if(root == key){
                return 0;
            }
            else if(root < key){
                return key;
            }
            else if(root > key)
                return root;
            return key;
    }

    }
    public static class Data{
        private Node first;
        public int size;
        public int index;
        public class Node{
```

```
    private int data;
    private Node next;
}

public Data(){
    first = null;
}

public boolean isEmpty(){
    return first == null;
}

public void add(int data){
    Node head = first;
    first = new Node();
    first.data = data;
    first.next = head;
    size++;
}

public int peek(){
    return first.data;
}

public int remove(){
    int data = first.data;
    first = first.next;
    size--;
    return data;
}

public int getValue(int index){
    int data = 0;

    if (index > size){
        return -1;
    }
    else{
        for(int i = 0;i<=index;i++){
            data = first.data;
            first = first.next;
```

```
                }
            add(data);
            return data;
        }
        }
    }
}
```

The search algorithm is trying to compare the left and right side of the values that were added like a BST. Since the left is always smaller than the right. We have a root value which is the index that we a searching for. Then it will compare it with the left and right nodes and return the value. Because of the left of right operations the tree will run in O(logN) time. So the entire search is O(logN)