https://course.acciojob.com/idle?question=6db52d7a-4879-482e-b702-e7f8fcd06331

- MEDIUM
- Max Score: 40 Points

Linked list cycle 2

You are given the `head` node of the linked list. If any cycle is present in the linked list, you have to return the node where the cycle begins. If there is no cycle return null.

A cycle means that the last node is connected to some other position of the list.

Note You need to only complete the function. Dont worry about input, it is for internal referece.

Input Format:

The first line of input contains `N` representing the number of nodes in linked list.

The second line of input contains `N` space seperated integers, representing elements in linked list.

The third line of input contains a number `x` representing which node is connected to other.

Output Format:

Return the node where the cycle begins. If the cycle is not present, return null.

Example 1

Input

```
3
1 3 4
2
```

Output

```
3
```

Explanation

```
1->3->4
  ^   |
  |___|
```

This is the list given in question. A loop is present in this linked list. The loop starts with node '3'.

Example 2

Input

```
4
1 2 3 4
0
```

Output

```
0
```

Explanation

`1->2->3->4`

This is list you are given in the question. Since, There is no cycle, we return null.

Constraints

1 <= N <= 1000

1 <= value of node <= 1000

Topic Tags

- **Linked lists**

My code

```
// n java
import java.util.*;
import java.io.*;
import java.lang.*;

class Node
{
    int data;
    Node next;

    Node(int x)
    {
        data = x;
```

```
        next = null;
    }
}

class Main
{
    public static void makeLoop(Node head, Node tail, int x){
        if (x == 0) return;

        Node curr = head;
        for(int i=1; i<x; i++)
            curr = curr.next;

        tail.next = curr;
    }

    public static void main (String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int num = sc.nextInt();
        Node head = new Node(num);
        Node tail = head;

        for(int i=0; i<n-1; i++)
        {
            num = sc.nextInt();
            tail.next = new Node(num);
            tail = tail.next;
        }
```

```
        int pos = sc.nextInt();
        makeLoop(head, tail, pos);

        Solution x = new Solution();
        Node y = x.detectLoop(head);
        if( y == null )
            System.out.println(0);
        else
            System.out.println(y.data);
    }
}

class Solution {
    public static Node detectLoop(Node node){
            HashMap<Node,Integer>hm=new HashMap<>();

while(node!=null)
        {
                hm.put(node,1);
                if(hm.containsKey(node.next))
                {
                      //node.next=null;
                        return node.next;
                }
                node=node.next;
        }
       return null;
    }
}
```