https://course.acciojob.com/idle?question=ea750944-265e-44cf-bd63-cf2bbc274594

MEDIUM

Max Score: 40 Points

Remove loop in Linked List

You are given the `head` node of the linked list. If any cycle is present in the linked list, you have to remove it without losing any nodes.

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

Just remove the loop without loss of nodes, you dont have to print anything. Driver will check if the loop has been removed correctly or not.

The driver prints -1 if your solution is incorrect

Example 1

Input

```
3
1 3 4
2
```

Output

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

Explanation

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

This is the list given in question. A loop is present in this linked list. We remove that loop.

Example 2

Input

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

Output

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

Explanation

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

This is list you are given in the question. Since, There is no cycle, we do not change anything.

Constraints

1 <= N <= 1000

1 <= value of node <= 1000

Topic Tags

Linked lists

My code

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

class Node{
    int data;
```

```
    Node next;
}

class Main{
    public static Node newNode(int data){
        Node temp = new Node();
        temp.data = data;
        temp.next = null;
        return temp;
    }

    public static void makeLoop(Node head, int x){
        if (x == 0)
            return;
        Node curr = head;
        Node last = head;

        int currentPosition = 1;
        while (currentPosition < x)
        {
            curr = curr.next;
            currentPosition++;
        }

        while (last.next != null)
            last = last.next;
        last.next = curr;
    }

    public static boolean detectLoop(Node head){
```

```
    Node hare = head.next;
    Node tortoise = head;
    while( hare != tortoise )
    {
        if(hare==null || hare.next==null) return false;
        hare = hare.next.next;
        tortoise = tortoise.next;
    }
    return true;
}

public static int length(Node head){
    int ret=0;
    while(head!=null){
        ret += 1;
        head = head.next;
    }
    return ret;
}
public static void printList(Node node){
    while (node != null) {
        System.out.print(node.data + " ");
        node = node.next;
    }
}

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

```
            Node head = newNode(num);
            Node tail = head;
            for(int i=0; i<n-1; i++){
                num = sc.nextInt();
                tail.next = newNode(num);
                tail = tail.next;
            }
            int pos = sc.nextInt();
            makeLoop(head, pos);
            Solution x = new Solution();
            x.removeLoop(head);
            if( detectLoop(head) || length(head)!=n )
                System.out.println("-1");
            else
                printList(head);

        }
    }

    class Solution{
        public static void removeLoop(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=node.next;
        }
    }

}
```