Find a Node in Linked List

Send Feedback

You have been given a singly linked list of integers. Write a function that returns the index/position of integer data denoted by 'N' (if it exists). Return -1 otherwise.

Note :

Assume that the Indexing for the singly linked list always starts from 0.

Input format :

The first line contains an Integer 'T' which denotes the number of test cases.

The first line of each test case or query contains the elements of the singly linked list separated by a single space.

The second line contains the integer value 'N'. It denotes the data to be searched in the given singly linked list.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence -1 would never be a list element.

Output format :

For each test case, return the index/position of 'N' in the singly linked list. Return -1, otherwise.

Output for every test case will be printed in a separate line.

Note:

You do not need to print anything; it has already been taken care of. Just implement the given function.

Constraints :

1 <= T <= 10^2
0 <= M <= 10^5

Where 'M' is the size of the singly linked list.

Time Limit: 1 sec

Sample Input 1 :

```
2
3 4 5 2 6 1 9 -1
5
10 20 30 40 50 60 70 -1
6
```

Sample Output 1 :

```
2
-1
```

```
public class Solution {

	public static int findNode(LinkedListNode<Integer> head, int n) {

		// Write your code here.

    if(head==null)

       return -1;

    LinkedListNode<Integer> temp=head;

    int count = 0;

    while(temp!=null && temp.data !=n){

       temp= temp.next;

       count++;

    }
```

```
        if(temp!=null)

            return count;

        else

            return-1;

              }

}
```

AppendLastNToFirst

Send Feedback

You have been given a singly linked list of integers along with an integer 'N'. Write a function to append the last 'N' nodes towards the front of the singly linked list and returns the new head to the list.

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

The first line of each test case or query contains the elements of the singly linked list separated by a single space.

The second line contains the integer value 'N'. It denotes the number of nodes to be moved from last to the front of the singly linked list.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence, would never be a list element.

Output format :

For each test case/query, print the resulting singly linked list of integers in a row, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^5
0 <= N < M
Time Limit: 1sec

Where 'M' is the size of the singly linked list.

Sample Input 1 :

```
2
1 2 3 4 5 -1
3
10 20 30 40 50 60 -1
5
```

Sample Output 1 :

```
3 4 5 1 2
20 30 40 50 60 10
```

```
public class Solution {



            public static LinkedListNode<Integer> appendLastNToFirst(LinkedListNode<Integer> head, int n)
{

                        //Your code goes here

        LinkedListNode<Integer> node=head,checkNode=null,newHead=null;
```

```
if (n==0)

{

    return head;

}


int count=0;

while(node!=null)

{

    node=node.next;

    count=count+1;

}

if (count<n)

{

    return head;

}


n=count-n;

node=head;

for (int i=0;i<n-1;i++)

{

    node=node.next;

}

checkNode=node.next;

node.next=null;

newHead=checkNode;

//System.out.println("Shifting from element: "+checkNode.data);

//System.out.println("Now last element is: "+node.data);

//System.out.println("Now first element is: "+newHead.data);

while(checkNode.next!=null)

{
```

```
          checkNode=checkNode.next;

        }

        checkNode.next=head;

        head=newHead;

        return head;



            }



}
```

Eliminate duplicates from LL

Send Feedback

You have been given a singly linked list of integers where the elements are sorted in ascending order. Write a function that removes the consecutive duplicate values such that the given list only contains unique elements and returns the head to the updated list.

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

The first and the only line of each test case or query contains the elements(in ascending order) of the singly linked list separated by a single space.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence, would never be a list element.

Output format :

For each test case/query, print the resulting singly linked list of integers in a row, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^5
Time Limit: 1sec

Where 'M' is the size of the singly linked list.

Sample Input 1 :

```
1
1 2 3 3 3 3 4 4 4 5 5 7 -1
```

Sample Output 1 :

```
1 2 3 4 5 7
```

```
public class Solution {



    public static LinkedListNode<Integer> removeDuplicates(LinkedListNode<Integer> head) {

        //Your code goes here
        if(head==null)

            return head;

        if(head.next==null)

            return head;

        LinkedListNode<Integer>t1 = head,t2=head.next;

        LinkedListNode<Integer>finalhead = head;

        while(t2 !=null){

            if(t1.data.equals(t2.data)){

                t2 = t2.next;

            }

            else{

                t1.next = t2;

                t1=t2;

            }

        }

        t1.next = null;

        return finalhead;



              }



}
```

Print Reverse LinkedList

Send Feedback

You have been given a singly linked list of integers. Write a function to print the list in a reverse order.

To explain it further, you need to start printing the data from the tail and move towards the head of the list, printing the head data at the end.

Note :

You can’t change any of the pointers in the linked list, just print it in the reverse order.

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

The first and the only line of each test case or query contains the elements of the singly linked list separated by a single space.

Remember/Constraints :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence, would never be a list element.

Output format :

For each test case, print the singly linked list of integers in a reverse fashion, in a row, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^3
Time Limit: 1sec

Where 'M' is the size of the singly linked list.

Sample Input 1 :

```
1
1 2 3 4 5 -1
```

Sample Output 1 :

```
5 4 3 2 1
```

```
public class Solution {



            public static void printReverse(LinkedListNode<Integer> root) {

                        //Your code goes here

        if(root==null)

            return;

        printReverse(root.next);

        System.out.print(root.data+" ");

            }



}
```

Palindrome LinkedList

Send Feedback

You have been given a head to a singly linked list of integers. Write a function check to whether the list given is a 'Palindrome' or not.

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First and the only line of each test case or query contains the the elements of the singly linked list separated by a single space.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence, would never be a list element.

Output format :

For each test case, the only line of output that print 'true' if the list is Palindrome or 'false' otherwise.

Constraints :

```
1 <= t <= 10^2
0 <= M <= 10^5
Time Limit: 1sec

Where 'M' is the size of the singly linked list.
```

Sample Input 1 :

```
1
9 2 3 3 2 9 -1
```

Sample Output 1 :

```
true
```

Sample Input 2 :

```
Import
java.util.*;
               public class Solution {
                      public static boolean isPalindrome(LinkedListNode<Integer> head) {
                             //Your code goes here
                       LinkedListNode<Integer> node=head;
                       ArrayList<Integer> arr = new ArrayList<>();
                       while (node!=null)
                       {
                           arr.add(node.data);
                           node=node.next;
                       }
                       int start=0, end=arr.size()-1;
                       while(start<end)
                       {
                           if (arr.get(start)!=arr.get(end))
                           {
                               return false;
                           }
                           else
                           {
                               start=start+1;
                               end=end-1;
                           }
                       }

                       return true;
                      }
               }
```

Delete node Recursively

Send Feedback

Given a singly linked list of integers and position 'i', delete the node present at the 'i-th' position in the linked list recursively.

Note :

Assume that the Indexing for the linked list always starts from 0.

No need to print the list, it has already been taken care. Only return the new head to the list.

input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

The first line of each test case or query contains the elements of the singly linked list separated by a single space.

The second line of input contains a single integer depicting the value of 'i'.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence, would never be a list element

Output format :

For each test case/query, print the elements of the updated singly linked list.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^5
Where M is the size of the singly linked list.
0 <= i < M

Time Limit: 1sec

Sample Input 1 :

```
1
3 4 5 2 6 1 9 -1
3
```

Sample Output 1 :

```
3 4 5 6 1 9
```

```
public class Solution {



        public static LinkedListNode<Integer> deleteNodeRec(LinkedListNode<Integer> head, int pos) {

        //Your code goes here

    if(head==null){

        return head;

    }

    if(pos==0){

        head = head.next;

        return head;

    }

    else{

        LinkedListNode<Integer>smallerHead = deleteNodeRec(head.next,pos-1);

        head.next = smallerHead;
        return head;

      }

           }



}
```

Reverse LL (Recursive)

Send Feedback

Given a singly linked list of integers, reverse it using recursion and return the head to the modified list. You have to do this in O(N) time complexity where N is the size of the linked list.

Note :

No need to print the list, it has already been taken care. Only return the new head to the list.

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

The first and the only line of each test case or query contains the elements of the singly linked list separated by a single space.

Remember/Consider :

While specifying the list elements for input, -1 indicates the end of the singly linked list and hence, would never be a list element

Output format :

For each test case/query, print the elements of the updated singly linked list.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= M <= 10^4
Where M is the size of the singly linked list.

Time Limit: 1sec

Sample Input 1 :

```
1
1 2 3 4 5 6 7 8 -1
```

Sample Output 1 :

```
8 7 6 5 4 3 2 1
```

```
public class Solution {



           public static LinkedListNode<Integer> reverseLinkedListRec(LinkedListNode<Integer> head) {

                      //Your code goes here

      if(head==null || head.next==null){

        return head;

      }

      LinkedListNode<Integer>smallerHead = reverseLinkedListRec(head.next);
```

```
        LinkedListNode<Integer> node=smallerHead;

        while(node.next !=null){

           node=node.next;

        }

        node.next=head;

        head.next=null;

        return smallerHead;

              }


}
```