### Intersection Point in Y Shaped Linked Lists

Given two singly linked lists of size **N** and **M,**write a program to get the point where two linked lists intersect each other.

**Example 1:**

**Input:**

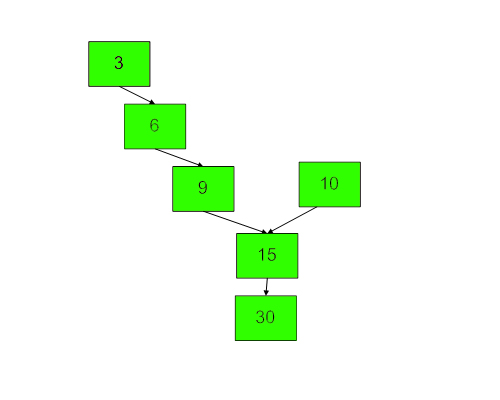
LinkList1 = 3->6->9->common

LinkList2 = 10->common

common = 15->30->NULL

**Output: 1**5

**Explanation:**



**Example 2:**

**Input:**

Linked List 1 = 4->1->common

Linked List 2 = 5->6->1->common

common = 8->4->5->NULL

**Output:** 8

**Explanation:**

**4 5**

**| |**

**1 6**

**\ /**

**8 ----- 1**

**|**

**4**

**|**

**5**

**|**

**NULL**

**Expected Time Complexity:** O(N+M)  
**Expected Auxiliary Space:** O(1)

**Constraints:**  
1 ≤ N + M ≤ 2\*105  
-1000 ≤ value ≤ 1000

**Company Tags**

[**VMWare**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=VMWare) [**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Snapdeal**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Snapdeal) [**D-E-Shaw**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=D-E-Shaw) [**FactSet**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=FactSet) [**MakeMyTrip**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MakeMyTrip) [**Visa**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Visa) [**Goldman Sachs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Goldman%20Sachs)[**MAQ Software**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MAQ%20Software) [**Adobe**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Adobe) [**Qualcomm**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Qualcomm)

//{ Driver Code Starts

import java.util.\*;

class Node

{

int data;

Node next;

Node(int d) {

data = d;

next = null;

}

}

class LinkedList\_Intersection

{

Node head = null;

Node tail = null;

public void addToTheLast(Node node)

{

if (head == null) {

head = node;

tail = head;

} else {

//Node temp = head;

//while (temp.next != null)

//temp = temp.next;

//temp.next = node;

tail.next=node;

tail = node;

}

}

/\* Function to print linked list \*/

void printList()

{

Node temp = head;

while (temp != null)

{

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

temp = temp.next;

}

System.out.println();

}

/\* Driver program to test above functions \*/

public static void main(String args[])

{

/\* Constructed Linked List is 1->2->3->4->5->6->

7->8->8->9->null \*/

Scanner sc = new Scanner(System.in);

int t=sc.nextInt();

while(t>0)

{

int n1 = sc.nextInt();

int n2 = sc.nextInt();

int n3 = sc.nextInt();

LinkedList\_Intersection llist1 = new LinkedList\_Intersection();

LinkedList\_Intersection llist2 = new LinkedList\_Intersection();

LinkedList\_Intersection llist3 = new LinkedList\_Intersection();

int a1=sc.nextInt();

Node head1= new Node(a1);

Node tail1= head1;

for (int i = 1; i < n1; i++)

{

int a = sc.nextInt();

tail1.next = (new Node(a));

tail1= tail1.next;

}

int b1=sc.nextInt();

Node head2 = new Node(b1);

Node tail2 = head2;

for (int i = 1; i < n2; i++)

{

int b = sc.nextInt();

tail2.next = (new Node(b));

tail2= tail2.next;

}

if(n3>0){

int c1=sc.nextInt();

Node head3= new Node(c1);

tail1.next = head3;

tail2.next = head3;

Node tail3=head3;

for (int i = 1; i < n3; i++)

{

int c = sc.nextInt();

tail3.next = (new Node(c));

tail3= tail3.next;

}

}

Intersect obj = new Intersect();

System.out.println(obj.intersectPoint(head1, head2));

t--;

}

}

}

// } Driver Code Ends

class Intersect

{

//Function to find intersection point in Y shaped Linked Lists.

public static int Length( Node head){

int c=1;

while(head!=null){

head=head.next;

c++;

}

return c;

}

int intersectPoint(Node head1, Node head2)

{

int n=Length(head1);

int m=Length(head2);

while(n>m){

head1=head1.next;

n--;

}

while(m>n){

head2=head2.next;

m--;

}

while(head1!=null && head2!=null){

if(head1==head2){

return head1.data;

}

head1=head1.next;

head2=head2.next;

}

return -1;

}

}