**Check If Circular Linked List**

Given **head**, the head of a singly linked list, find if the linked list is circular or not. A linked list is called circular if it not NULL terminated and all nodes are connected in the form of a cycle. An empty linked list is considered as circular.

**Note:** The linked list does not contains any inner loop.

**Example 1:**

**Input:**

LinkedList: 1->2->3->4->5

(the first and last node is connected,

i.e. 5 --> 1)

**Output:** 1

**Example 2:**

**Input:**

LinkedList: 2->4->6->7->5->1

**Output:** 0

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

**Constraints:**  
1 <=Number of nodes<= 100

**Company Tags**

[**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**MAQ Software**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MAQ%20Software) [**SAP Labs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=SAP%20Labs)

//{ Driver Code Starts

import java.util.\*;

class Node

{

int data;

Node next;

Node(int d)

{

data = d;

next = null;

}

}

class Is\_LinklIst\_Circular

{

Node head;

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while(t>0)

{

int n = sc.nextInt();

int l = sc.nextInt();

sc.nextLine().trim().split(" ");

String str[]= sc.nextLine().trim().split(" ");

Node head = null, tail = null;

head = new Node(Integer.parseInt(str[0]));

tail = head;

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

{

tail.next = new Node(Integer.parseInt( str[i]));

tail = tail.next;

}

if(l==1 && n>=1)

{

Node temp = head;

while(temp.next != null)

{

temp = temp.next;

}

temp.next = head;

}

CodingMaxima g = new CodingMaxima ();

boolean b = g.isCircular(head);

if(b==true)

System.out.println(1);

else

System.out.println(0);

t--;

}

}

}

// } Driver Code Ends

/\* Structure of LinkedList

class Node

{

int data;

Node next;

Node(int d)

{

data = d;

next = null;

}

}

\*/

class CodingMaxima

{

boolean isCircular(Node head)

{

// Your code here

Node slow=head;

Node fast=head;

while(fast!=null && fast.next!=null){

fast=fast.next.next;

slow=slow.next;

if(slow==fast)

return true;

}

return false;

}

}