**HACKERRANK CHALLENGE:CYCLE DETECTION**

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

public class Solution {

    static class SinglyLinkedListNode {

        public int data;

        public SinglyLinkedListNode next;

        public SinglyLinkedListNode(int nodeData) {

            this.data = nodeData;

            this.next = null;

        }

    }

    static class SinglyLinkedList {

        public SinglyLinkedListNode head;

# public SinglyLinkedListNode tail;

        public SinglyLinkedList() {

            this.head = null;

            this.tail = null;

        }

        public void insertNode(int nodeData) {

            SinglyLinkedListNode node = new SinglyLinkedListNode(nodeData);

            if (this.head == null) {

                this.head = node;

            } else {

                this.tail.next = node;

            }

            this.tail = node;

        }

    }

    public static void printSinglyLinkedList(SinglyLinkedListNode node, String sep, BufferedWriter bufferedWriter) throws IOException {

        while (node != null) {

            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {

                bufferedWriter.write(sep);

            }

        }

    }

    // Complete the hasCycle function below.

    /\*

     \* For your reference:

     \*

     \* SinglyLinkedListNode {

     \*     int data;

     \*     SinglyLinkedListNode next;

     \* }

     \*

     \*/

    static boolean hasCycle(SinglyLinkedListNode head) {

        if(head==null)

        {

            return false;

        }

    SinglyLinkedListNode slow=head;

    SinglyLinkedListNode fast=head;

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

    {

        slow=slow.next;

        fast=fast.next.next;

        if(slow==fast)

        return true;

    }

    return false;

    }

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) throws IOException {

        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

        int tests = scanner.nextInt();

        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

        for (int testsItr = 0; testsItr < tests; testsItr++) {

            int index = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            SinglyLinkedList llist = new SinglyLinkedList();

            int llistCount = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            for (int i = 0; i < llistCount; i++) {

                int llistItem = scanner.nextInt();

                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist.insertNode(llistItem);

            }

            SinglyLinkedListNode extra = new SinglyLinkedListNode(-1);

            SinglyLinkedListNode temp = llist.head;

            for (int i = 0; i < llistCount; i++) {

                if (i == index) {

                    extra = temp;

                }

                if (i != llistCount-1) {

                    temp = temp.next;

                }

            }

            temp.next = extra;

            boolean result = hasCycle(llist.head);

            bufferedWriter.write(String.valueOf(result ? 1 : 0));

            bufferedWriter.newLine();

        }

        bufferedWriter.close();

        scanner.close();

    }

}