PERMUT2 - Ambiguous Permutations

[#ad-hoc-1](http://www.spoj.com/problems/tag/ad-hoc-1)

Some programming contest problems are really tricky: not only do they require a different output format from what you might have expected, but also the sample output does not show the difference. For an example, let us look at permutations.  
A **permutation** of the integers *1* to *n* is an ordering of these integers. So the natural way to represent a permutation is to list the integers in this order. With *n = 5*, a permutation might look like 2, 3, 4, 5, 1.   
However, there is another possibility of representing a permutation: You create a list of numbers where the *i*-th number is the position of the integer *i* in the permutation. Let us call this second possibility an **inverse permutation**. The inverse permutation for the sequence above is 5, 1, 2, 3, 4.   
An **ambiguous permutation** is a permutation which cannot be distinguished from its inverse permutation. The permutation 1, 4, 3, 2 for example is ambiguous, because its inverse permutation is the same. To get rid of such annoying sample test cases, you have to write a program which detects if a given permutation is ambiguous or not.

Input Specification

The input contains several test cases.  
The first line of each test case contains an integer *n* (*1 ≤ n ≤ 100000*). Then a permutation of the integers *1* to *n* follows in the next line. There is exactly one space character between consecutive integers. You can assume that every integer between *1* and *n*appears exactly once in the permutation.   
The last test case is followed by a zero.

Output Specification

For each test case output whether the permutation is ambiguous or not. Adhere to the format shown in the sample output.

Sample Input

4

1 4 3 2

5

2 3 4 5 1

1

1

0

Sample Output

ambiguous

not ambiguous

ambiguous

<http://www.spoj.com/problems/PERMUT2/>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

//int n = 5; // int.Parse(Console.ReadLine());

//int[] permut = { 2 ,3, 4, 5, 1 };

while (true)

{

int n = int.Parse(Console.ReadLine());

if (n == 0)

{

break;

}

int[] permut = Array.ConvertAll(Console.ReadLine().Split(' '), e => int.Parse(e));

int[] inverse = new int[n];

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

{

inverse[permut[i] - 1] = i + 1;

}

//foreach (int elem in inverse)

//{

// Console.Write(elem + " ");

//}

string res = "ambiguous";

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

{

if (inverse[i] != permut[i])

{

res = "not ambiguous";

break;

}

}

Console.WriteLine(res);

}

// Console.ReadLine();

}

}

}

----------OTRA SOLUCION ACEPTADA-------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

while (true)

{

int n = int.Parse(Console.ReadLine());

if (n == 0) break;

int[] arr = Array.ConvertAll(Console.ReadLine().Split(' '), e => int.Parse(e));

string ans = "ambiguous";

for (int i = 0; i +1 < arr.Length; i++)

{

if (arr[arr[i] - 1] + 1 != arr[arr[i+1] - 1])

{

ans = "not ambiguous";

break;

}

}

Console.WriteLine(ans);

}

Console.ReadLine();

}

}

}