Permutações Ambíguas

Contest Local, Universidade de Ulm Alemanha

Timelimit: 5

Alguns problemas de competições de programação são mesmo melindrosos: não apenas exigem um formato de saída diferente do que você podia esperar, mas também o exemplo de saída não mostra a diferença. Por exemplo, vejamos as permutações.

Uma **permutação** dos inteiros de *1* a *n* é uma ordenação desses inteiros. Então a maneira natural de se representar uma permutação é listar os inteiros nessa ordem. Para *n* = *5*, uma permutação seria 2, 3, 4, 5, 1.

Entretanto, há outra possibilidade de representar-se uma permutação: Cria-se uma lista de números onde o *i*-ésimo número é a posição do inteiro *i* na permutação. Chamemos essa segunda possibilidade de uma **permutação inversa**. A permutação inversa da sequência acima é 5, 1, 2, 3, 4.

Uma **permutação ambígua** é uma permutação que não pode distinguida de sua permutação inversa. A permutação 1, 4, 3, 2, por exemplo, é ambígua, porque sua permutação inversa é a mesma. Para se livrar desses irritantes exemplos de casos de teste, você deve escrever um programa que detecta se a permutação dada é ambígua ou não.

Entrada

A entrada consiste de vários casos de teste.

A primeira linha de cada caso de teste contém um inteiro **n** (1 ≤ **n** ≤ 100000). A linha seguinte contém uma permutação de inteiros 1 a **n**. Há exatamente um caractere de espaço entre inteiros consecutivos. Assuma que todo inteiro entre 1 e **n** aparece exatamente uma vez na permutação.

O último caso de teste é seguido por uma linha que contém um zero.

Saída

Para cada caso de teste imprima se a permutação é ambígua ou não, de acordo com o formato mostrado no exemplo de saída.

Exemplo de Entrada	Exemplo de Saída
4 1 4 3 2 5 2 3 4 5 1 1 1 0	ambiguous not ambiguous ambiguous

Univeristy of Ulm Local Contest 2005/2006