Ambiguous permutations

1 to n is obtained by placing these numbers in a certain order. A logical way to represent permutations is thus lists. For $n = 5$ a permutation can, for example, be represented in the following way:

```
2, 3, 4, 5, 1
```

However, there is another way to represent permutations: as a list of integeres, where the i-th number indicates the position of the number i in the permutation. This second representation is called an inverse permutation. The **inverse permutation** of the above example is thus:

```
5, 1, 2, 3, 4
```

because number 1 is at position 5 in the permutation, number 2 is at position 1 in the permutation, …

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.

Assignment

- Write a function inversePermutation to which a list of integers must be passed as argument. This list must represent a permutation of the numbers 1 to n, where n is equal to the length of the list. The function must return a new list as a result, which represents the inverse permutation.
- Use the inversePermutation function to write a function ambiguousPermutation. To this function, a list of integers must be passed as an argument, which is a permutation of the numbers 1 to n (with n equalling the length of the list). The function must return a Boolean value as a result, which indicates whether the given list is ambiguous or not.

Example

```
>>> inversePermutation([1, 4, 3, 2])
[1, 4, 3, 2]
>>> inversePermutation([2, 3, 4, 5, 1])
[5, 1, 2, 3, 4]
>>> inversePermutation([1])
[1]

>>> ambiguousPermutation([1, 4, 3, 2])
True
>>> ambiguousPermutation([2, 3, 4, 5, 1])
False
>>> ambiguousPermutation([1])
True
```

Een **permutatie** van de natuurlijke getallen 1 tot en met n wordt bekomen door deze getallen

in een bepaalde volgorde te plaatsen. Een logische manier om permutaties voor te stellen, bestaat er dus in om de getallen in deze volgorde op te lijsten. Voor $n = 5$ kan een permutatie dus op de volgende manier voorgesteld worden:

2, 3, 4, 5, 1

Er bestaat echter nog een andere manier om permutaties voor te stellen: als een lijst van natuurlijke getallen, waarbij het i-de getal de positie van getal i in de permutatie aangeeft. Deze tweede voorstelling wordt een **omgekeerde permutatie** genoemd. De omgekeerde permutatie van bovenstaand voorbeeld wordt dus:

5, 1, 2, 3, 4

want getal 1 staat op positie 5 in de permutatie, getal 2 staat op positie 1 in de permutatie, …

Een **dubbelzinnige permutatie** is een permutatie die niet verschilt van zijn omgekeerde permutatie. De permutatie 1, 4, 3, 2 is bijvoorbeeld dubbelzinnig, omdat de omgekeerde permutatie precies hetzelfde is.

Opgave

- Schrijf een functie omgekeerdePermutatie waaraan een lijst van natuurlijke getallen als argument moet doorgegeven worden. Deze lijst moet een permutatie van de getallen 1 tot en met n voorstellen, waarbij n gelijk is aan de lengte van de lijst. De functie moet een nieuwe lijst als resultaat teruggeven, die de omgekeerde permutatie voorstelt.
- Gebruik de functie omgekeerdePermutatie om een functie dubbelzinnigePermutatie te schrijven. Aan deze functie moet een lijst van natuurlijke getallen als argument doorgegeven worden, die een permutatie van de getallen 1 tot en met n voorstelt (met n gelijk aan de lengte van de lijst). De functie moet een Booleaanse waarde als resultaat teruggeven, die aangeeft of de gegeven lijst dubbelzinnig is of niet.

Voorbeeld

```
>>> omgekeerdePermutatie([1, 4, 3, 2])
[1, 4, 3, 2]
>>> omgekeerdePermutatie([2, 3, 4, 5, 1])
[5, 1, 2, 3, 4]
>>> omgekeerdePermutatie([1])
[1]

>>> dubbelzinnigePermutatie([1, 4, 3, 2])
True
>>> dubbelzinnigePermutatie([2, 3, 4, 5, 1])
False
>>> dubbelzinnigePermutatie([1])
True
```