Hermit crabs

Hermit crabs adopt other creatures' castoff shells for protection. But as they grow, crabs must move into successively larger shells. This produces a curious phenomenon: when a crab finds a shell that's too big for it, it waits nearby. Other crabs may accumulate, forming a little conga line of dissatisfied shell seekers. Finally a "Goldilocks" crab arrives — a crab large enough to claim the new shell — and now each waiting crab can move into the shell abandoned by its larger neighbor. By cooperating to share a scarce resource, the whole species benefits.

The same thing happens in human societies — when one person finds a new apartment, car, or job, she leaves behind her old one, and the vacancy passes down through society until the final unit is cast away or destroyed. It's called a *vacancy chain*.

Assignment

In this assignment we work with sequences of integers that represent shell sizes. A **sequence of consecutive integers** is a sequence of n integers $a_1, a_2, \ldots, a_n$ such that $a_i + 1 = a_{i + 1} (i = 1, \ldots, n - 1)$. Your task:

- Write a function `consecutive` that takes a **sequence** (a list or a tuple) of integers. The function must return a Boolean value that indicates whether or not the given sequence forms a sequence of consecutive integers if the integers are sorted in increasing order.
- Write a function `goldilocks` that takes a **sequence** (a list or a tuple) of integers. In case the given sequence cannot be used to form a sequence of consecutive integers, but if a single integer can be added to the list so that a sequence of consecutive integers can be formed, the function must return this "missing" integer. Otherwise, the function must return the value `None`.
- Write a function `move1` that takes a **sequence** (a list or a tuple) of integers. The function must return a list that contains a copy of the given sequence of numbers, in the given order. In case the given sequence cannot be used to form a sequence of consecutive integers, but if a single integer can be added to the list so that a sequence of consecutive integers can be formed, the function must append this "missing" integer to the end of the list that is returned.
- Write a function `move2` that takes a **list** of integers. The function may not return a value (read:

the function must return the value None), but in case the given integers cannot be used to form a sequence of consecutive integers, and a single integer can be added to the list so that a sequence of consecutive integers can be formed, this "missing" integer must be appened to the end of the given list.

Example

```
>>> consecutive([7, 5, 4, 9, 6, 3, 8])
True
>>> consecutive((16, 13, 18, 17, 15, 14, 20))
False
>>> consecutive((3, 4, 1, 6, 8, 7))
False

>>> goldilocks([7, 5, 4, 9, 6, 3, 8])
>>> goldilocks((16, 13, 18, 17, 15, 14, 20))
19
>>> goldilocks((3, 4, 1, 6, 8, 7))

>>> move1([7, 5, 4, 9, 6, 3, 8])
[7, 5, 4, 9, 6, 3, 8]
>>> move1((16, 13, 18, 17, 15, 14, 20))
[16, 13, 18, 17, 15, 14, 20, 19]
>>> move1((3, 4, 1, 6, 8, 7))
[3, 4, 1, 6, 8, 7]

>>> shells = [7, 5, 4, 9, 6, 3, 8]
>>> move2(shells)
>>> shells
[7, 5, 4, 9, 6, 3, 8]

>>> shells = [16, 13, 18, 17, 15, 14, 20]
>>> move2(shells)
>>> shells
[16, 13, 18, 17, 15, 14, 20, 19]

>>> shells = [3, 4, 1, 6, 8, 7]
>>> move2(shells)
>>> shells
[3, 4, 1, 6, 8, 7]
```

Heremietkreeften gebruiken voor hun eigen bescherming schelpen die andere dieren hebben weggegooid. Naarmate ze groeien, moeten de kreeften steeds op zoek naar grotere schelpen. Dit levert een merkwaardig verschijnsel op: als een kreeft een verlaten schelp vindt die te groot voor hem is, dan blijft hij gewoon naast die schelp wachten tot er zich andere kreeften rond de schelp beginnen te verzamelen. Hierdoor vormt zich een polonaise van ontevreden schelpbewoners. Als er uiteindelijk een "Goudlokjes"-kreeft bij de groep aansluit — een kreeft die groot genoeg is om in de verlaten schelp in te trekken — dan kan elke wachtende kreeft verhuizen naar de schelp die zijn grotere buur net heeft verlaten. Zo haalt de hele soort haar voordeel uit de samenwerking om schaarse goederen te delen.

Hetzelfde fenomeen doet zich ook bij mensen voor — wanneer één persoon een nieuw appartement, een nieuwe wagen of een nieuwe job vindt, dan laat hij een oude achter die aan een opvolger in de maatschappij wordt doorgegeven, totdat wat laatst overblijft uiteindelijk wordt weggegooid of vernietigd. In de sociologie wordt dit aangeduid met de Engelse term *vacancy chain*.

Opgave

In deze opgave werken we met reeksen getallen die de groottes van schelpen voorstellen. Een **reeks opeenvolgende gehele getallen** is een reeks van n gehele getallen $a_1, a_2, \ldots, a_n$ waarvoor geldt dat $a_i + 1 = a_{i + 1} (i = 1, \ldots, n - 1)$. Gevraagd wordt:

- Schrijf een functie `opeenvolgend` waaraan een **reeks** (een lijst of een tuple) van gehele getallen moet doorgegeven worden. De functie moet een Booleaanse waarde teruggeven, die aangeeft of de gegeven reeks een reeks opeenvolgende gehele getallen vormt als ze in oplopende volgorde gesorteerd wordt.
- Schrijf een functie `goudlokje` waaraan een **reeks** (een lijst of een tuple) van gehele getallen moet doorgegeven worden. Indien met de gegeven reeks geen opeenvolgende reeks gehele getallen kan gevormd worden, maar indien dit wel kan als er juist één geheel getal aan de reeks wordt toegevoegd, dan moet dit "ontbrekend" getal door de functie als resultaat teruggegeven worden. Anders moet de functie de waarde `None` als resultaat teruggeven.
- Schrijf een functie `verhuizen1` waaraan een **reeks** (een lijst of een tuple) van gehele getallen moet doorgegeven worden. De functie moet een lijst teruggeven die een kopie van de gegeven reeks getallen bevat, in de gegeven volgorde. Indien met de gegeven reeks geen reeks opeenvolgende gehele getallen kan gevormd worden, maar indien dit wel kan als er juist één geheel getal aan de reeks wordt toegevoegd, dan moet achteraan de lijst die door de functie wordt teruggegeven nog dit extra "ontbrekend" getal toegevoegd worden.
- Schrijf een functie `verhuizen2` waaraan een **lijst** van gehele getallen moet doorgegeven worden. De functie mag geen waarde teruggeven (lees: de functie moet de waarde `None` teruggeven), maar indien met de gegeven getallen geen reeks opeenvolgende gehele getallen kan gevormd worden, maar dit wel kan als er juist één geheel getal aan de lijst wordt toegevoegd, dan moet achteraan de gegeven lijst dit extra "ontbrekend" getal toegevoegd worden.

Voorbeeld

```
>>> opeenvolgend([7, 5, 4, 9, 6, 3, 8])
True
>>> opeenvolgend((16, 13, 18, 17, 15, 14, 20))
False
>>> opeenvolgend((3, 4, 1, 6, 8, 7))
False

>>> goudlokje([7, 5, 4, 9, 6, 3, 8])
>>> goudlokje((16, 13, 18, 17, 15, 14, 20))
19
>>> goudlokje((3, 4, 1, 6, 8, 7))

>>> verhuizen1([7, 5, 4, 9, 6, 3, 8])
[7, 5, 4, 9, 6, 3, 8]
>>> verhuizen1((16, 13, 18, 17, 15, 14, 20))
[16, 13, 18, 17, 15, 14, 20, 19]
>>> verhuizen1((3, 4, 1, 6, 8, 7))
[3, 4, 1, 6, 8, 7]

>>> schelpen = [7, 5, 4, 9, 6, 3, 8]
>>> verhuizen2(schelpen)
>>> schelpen
[7, 5, 4, 9, 6, 3, 8]

>>> schelpen = [16, 13, 18, 17, 15, 14, 20]
>>> verhuizen2(schelpen)
>>> schelpen
[16, 13, 18, 17, 15, 14, 20, 19]

>>> schelpen = [3, 4, 1, 6, 8, 7]
>>> verhuizen2(schelpen)
>>> schelpen
[3, 4, 1, 6, 8, 7]
```