Der bestehende Iterator Ihres **`ADS_set`** ist um einen zusätzlichen „Modus“ zu erweitern. In der bisherigen Implementierung liefert der Iterator alle *n* Elemente des **`ADS_set`** in einer beliebigen Reihenfolge, wobei die Reihenfolge immer dieselbe sein muss, solange das **`ADS_set`** nicht geändert wird (Modus „normal“). Im neuen Modus „speziell“ liefert der Iterator die Elemente grundsätzlich in derselben Reihenfolge, aber mit den folgenden Einschränkungen: Wenn $n < 3$ dann werden alle *n* Elemente geliefert, andernfalls:

- Wenn das 2. Element größer ist als das 1. Element, dann werden nach dem 1. Element nur mehr genau jene Elemente geliefert, die größer[1] als das unmittelbar zuvor gelieferte Element sind, alle anderen Elemente werden übersprungen („aufsteigend sortierte Teilfolge“).
- Wenn das 2. Element kleiner ist als das 1. Element, dann werden nach dem 1. Element nur mehr genau jene Elemente geliefert, die kleiner[1] als das unmittelbar zuvor gelieferte Element sind, alle anderen Elemente werden übersprungen („absteigend sortierte Teilfolge“).

In beiden Modi erreicht der Iterator nach dem letzten gelieferten Element **`end()`**.

Details: Erweitern Sie Ihre Implementierung **`ADS_set`** um die Methode

```
const_iterator y() const;
```

`y()` erzeugt einen Iterator im Modus „speziell“. Wenn kein Element im **`ADS_set`** vorhanden ist, dann gilt **`y()`** == **`end()`**. Die Zeitkomplexität und Speicherkomplexität der Operatorfunktionen müssen unverändert bleiben. So sind z. B. zusätzliche Felder mit nicht konstanter Größe unzulässig.

Angenommen der von **`begin()`** retournierte Iterator liefert alle n gespeicherten Elemente in der Reihenfolge	Dann liefert der von y**`()`** retournierte Iterator die folgenden Elemente in der folgenden Reihenfolge
()	()
(1)	(1)
(1,4)	(1,4)
(7,4)	(7,4)
(1,4,2,3,5)	(1,4,5)
(7,4,6,5,3,8,1,2)	(7,4,3,1)

Anleitung: Schreiben Sie **keine** neue Iteratorklasse! Erweitern Sie die bestehende Iterator-Klasse wie folgt (dies ist nur einer der möglichen Lösungsansätze, abweichende korrekte Lösungen sind natürlich zulässig): Es muss ein Iterator im Modus „speziell“ erzeugt werden können. Dazu ist ein neuer Konstruktor zu schreiben und/oder bestehende zu erweitern. Eventuell benötigen Sie zusätzliche Konstruktorparameter und/oder Instanzvariablen. Die ersten beiden Elemente (falls vorhanden) legen die Sortierung (aufsteigend oder absteigend) fest. Passen Sie die Inkrement-Operationen für den Modus „speziell“ an: Alle Elemente, die kleiner (falls Sortierung aufsteigend) oder größer (falls Sortierung absteigend) sind als das zuvor gelieferte Element, werden übersprungen. Nach dem letzten zu liefernden Element wird der Iterator (wie immer) auf **`end()`** gesetzt.

Die Methode **`ADS_set::begin()`** liefert wie bisher einen Iterator im Modus „normal“.

---

[1] Der Aufruf **`std::less<key_type>{}(key1,key2)`** für die beiden Elemente **`key1`** und **`key2`** liefert **`true`**, falls **`key1`** kleiner als **`key2`** ist und **`false`** sonst (alternativ kann der alias **`key_compare`** verwendet werden, sofern dieser entsprechend definiert ist).