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 mit Ausnahme das kleinsten[1] Elements dieselben Elemente in derselben Reihenfolge, das kleinste[1] im **`ADS_set`** enthaltene Element wird nicht geliefert (also „übersprungen“). In beiden Modi erreicht der Iterator nach dem letzten gelieferten Element **`end()`**.

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

```
const_iterator x() const;
```

`x()` erzeugt einen Iterator im Modus „speziell“. Wenn kein Element im **`ADS_set`** vorhanden ist, dann gilt **`x()`** == **`end()`**. Die Zeitkomplexität von **`x()`** muss O(n) sein, die Speicherkomplexität O(1). Die Zeitkomplexität und Speicherkomplexität der Operatorfunktionen müssen unverändert bleiben. So sind z. B. zusätzliche Felder mit nicht konstanter Größe unzulässig. Für die Verwendung der STL gelten dieselben Regeln wie für das gesamte Projekt.

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

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. Sie benötigen in der Regel zusätzliche Konstruktorparameter und/oder Instanzvariablen.
- Passen Sie die Iterator-Funktionen für den Modus „speziell“ an: wenn das kleinste Element an erster Stelle in der Iteratorreihenfolge steht, erfolgt das Überspringen bereits im Konstruktor, andernfalls in den Inkrement-Operationen. Nach dem letzten zu liefernden Element wird der Iterator (wie immer) auf **`end()`** gesetzt.
- Führen Sie in **`x()`** eine Minimumsuche durch, um das kleinste Element zu ermitteln, und erzeugen Sie einen entsprechenden Iterator im Modus speziell.

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).