Erweitern Sie Ihre Implementierung des **ADS_set** um die Methode

std::pair<size_type,key_type> y() const

Diese liefert den größten jemals in diesem **ADS_set** enthaltenen Wert („historisches Maximum“). Das historische Maximum kann auch ein Wert sein, der inzwischen wieder entfernt worden ist. Sollten in dem **ADS_set** zum Zeitpunkt des Aufrufs von **y()** noch nie Werte enthalten gewesen sein, so ist **y().first** gleich 0 und der Wert von **y().second** ist nicht definiert (d.h. es ist jeder beliebige Wert zulässig). Andernfalls ist **y().second** das historische Maximum und **y().first** ist die Anzahl der Funktionsaufrufe, die das historische Maximum bis dahin neu gesetzt haben.

Zu diesem Zweck sind im **ADS_set** zusätzlich das historische Maximum sowie die Anzahl der Funktionsaufrufe zu speichern, die das historische Maximum verändert haben („Historie“). Bitte beachten: es sind nur „externe“ Aufrufe der Funktionen zu berücksichtigen. Externe Aufrufe sind Aufrufe von außerhalb des **ADS_set** (also hier vom Unit-Test), nicht hingegen Aufrufe dieser Funktionen durch andere Funktionen des **ADS_set** („interne“ Aufrufe). Jeder Funktionsaufruf, der das historische Maximum neu setzt, zählt als ***ein*** Aufruf, auch wenn mit einem Aufruf mehrere Werte eingefügt werden, die größer sind als das bisherige historische Maximum.

Die einzelnen Funktionen verhalten sich wie folgt:

Wenn eine der **insert**-Funktionen[1] einen Wert einfügt, der größer ist als das bisherige historische Maximum, dann wird das historische Maximum neu gesetzt und der Aufruf zählt.

Defaultkonstruktor, Initializer-List-Konstruktor, Range-Konstruktor und Initializer-List-Zuweisungsoperator[2] setzen die Historie in den Anfangszustand (noch kein historisches Maximum vorhanden). Wenn durch diese Funktionen auch Werte eingefügt werden, wird das historische Maximum gesetzt und der Funktionsaufruf zählt.

Kopierkonstruktor und Kopierzuweisungsoperator[3] übernehmen die Historie aus **other**. Der Funktionsaufruf zählt nicht.

Die **swap**-Operationen tauschen die Historie der beiden **ADS_set**s. Der Funktionsaufruf zählt nicht.

Die übrigen Operationen ignorieren die Historie.

Die Zeit- und Speicherkomplexität von **y()** muss O(1) sein. Die Zeit- und Speicherkomplexität aller übrigen Methoden und Funktionen müssen unverändert (spezifikationskonform) bleiben.

Der Aufruf **std::less<key_type>{}(key1,key2)** für die beiden Werte **key1** und **key2** liefert **true**, falls **key1** kleiner als **key2** ist und **false** sonst (alternativ kann der alias **key_compare** verwendet werden, sofern vorhanden).

---

[1] `void insert(std::initializer_list<key_type> ilist)`
`std::pair<iterator,bool> insert(const key_type &key)`
`template<typename InputIt> void insert(InputIt first, InputIt last)`

[2] `ADS_set()`
`ADS_set(std::initializer_list<key_type> ilist)`
`template<typename InputIt> ADS_set(InputIt first, InputIt last)`
`ADS_set &operator=(std::initializer_list<key_type> ilist)`

[3] `ADS_set(const ADS_set &other)`
`ADS_set &operator=(const ADS_set &other)`