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

```
key_type w() const;
```

Diese soll den größten aktuell im ADS_set gespeicherten Wert liefern, der kleiner ist als der größte jemals in diesem **ADS_set** vorhandene Wert („historisches Maximum“). Das historische Maximum kann auch ein Wert sein, der inzwischen wieder entfernt worden ist. Wenn in dem **ADS_set** zum Zeitpunkt des Aufrufs von **w()** kein entsprechender Werte enthalten ist, dann ist eine **exception** vom Typ **std::runtime_error** zu werfen (dafür ist **#include <stdexcept>** erforderlich). Andernfalls ist in **w()** eine Maximumsuche im **ADS_set** durchzuführen, bei der das historische Maximum zu ignorieren ist.

Anders ausgedrückt: wenn das historische Maximum aktuell im **ADS_set** gespeichert ist, dann liefert **w()** den zweitgrößten Wert im **ADS_set**, andernfalls liefert **w()** den größten Wert im **ADS_set**.

Zu diesem Zweck sind im **ADS_set** zusätzlich der größte jemals vorhandene Wert zu verwalten („historisches Maximum“) und bei allen entsprechenden Operationen gegebenenfalls anzupassen.

Beim Kopierkonstruktor **ADS_set(const ADS_set &other)** wird das historische Maximum aus **other** übernommen.

Bei allen anderen Konstruktoren

```
ADS_set();
ADS_set(std::initializer_list<key_type> ilist);
template<typename InputIt> ADS_set(InputIt first, InputIt last);
```

ist das historische Maximum nach der Erzeugung des **ADS_set** „nicht vorhanden“ und wird gegebenenfalls auf Basis der eingefügten Werte gesetzt.

Bei den Zuweisungsoperatoren

```
ADS_set &operator=(std::initializer_list<key_type> ilist);
ADS_set &operator=(const ADS_set &other);
```

bleibt das historische Maximum erhalten und wird gegebenenfalls auf Basis der Werte in **ilist** bzw. **other** angepasst.

Bei **swap**-Operationen werden die historischen Maxima der beiden **ADS_set**s getauscht. Bei **clear** und **erase** bleibt das historische Maximum unverändert.

Bei den Vergleichsoperationen

```
friend bool operator==(const ADS_set &lhs, const ADS_set &rhs);
friend bool operator!=(const ADS_set &lhs, const ADS_set &rhs);
```

wird das historische Maximum ignoriert.

Die Zeitkomplexität von **w()** muss O(n) sein, die Speicherkomplexität O(1). Die Zeit- und Speicherkomplexität aller übrigen Methoden und Funktionen müssen unverändert (spezifikationskonform) bleiben. Für die Verwendung der STL gelten dieselben Regeln wie im übrigen Projekt.

Zum Vergleich zweier Werte vom Typ **key_type** ist **std::less<key_type>** zu verwenden (oder der alias **key_compare** falls vorhanden). **std::less<key_type>{}(key1,key2)** liefert für die beiden Werte **key1** und **key2** **true**, falls **key1** kleiner als **key2** ist und **false** sonst.