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

```
key_type z() const;
```

Diese soll den zweitgrößten jemals in diesem **ADS_set** enthaltenen Wert liefern. Das kann auch ein Wert sein, der inzwischen wieder entfernt worden ist. Sollten in dem **ADS_set** zum Zeitpunkt des Aufrufs von **z()** noch keine zwei unterschiedlichen Werte enthalten gewesen sein, so ist eine exception vom Typ **std::runtime_error** zu werfen (dafür ist **#include <stdexcept>** erforderlich).

Zu diesem Zweck sind im **ADS_set** zusätzlich der größte und zweitgrößte jemals enthaltene Wert zu speichern („Historie“) und bei Einfügeoperationen gegebenenfalls entsprechend anzupassen.

Beim Kopierkonstruktor und Kopierzuweisungsoperator
```
ADS_set(const ADS_set &other);
ADS_set &operator=(const ADS_set &other);
```
wird die Historie aus **other** übernommen.

Beim Initializer-List-Zuweisungsoperator
```
ADS_set &operator=(std::initializer_list<key_type> ilist);
```
wird die Historie „gelöscht“ und aus den Werten in **ilist** neu ermittelt.

Bei **swap**-Operationen wird die Historie der beiden **ADS_set**s getauscht.

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 die Historie ignoriert.

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

Zum Vergleich zweier Werte vom Typ **key_type** ist **std::less<key_type>** zu verwenden (oder der alias **key_compare** falls vorhanden).

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.