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

```
std::pair<key_type,key_type> x() const;
```

Diese soll den kleinsten (**x().first**) und größten (**x().second**) jemals in dieses **ADS_set** erfolgreich mit **insert()** eingefügten Wert liefern. Das können auch Werte sein, die inzwischen wieder entfernt worden sind. Sollten in dem **ADS_set** zum Zeitpunkt des Aufrufs von **x()** noch keine mit **insert()** eingefügten 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 kleinste jemals mit **insert** erfolgreich eingefügte Wert zu verwalten („Historie“) und bei Einfügeoperationen gegebenenfalls entsprechend anzupassen. Bitte beachten: es sind nur „externe“ Aufrufe einer der drei **insert**-Funktionen[1] 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).

Beim Kopierkonstruktor und Kopierzuweisungsoperator[2] wird die Historie aus **other** übernommen.

Bei allen anderen Konstruktoren[3] sowie beim Initializer-List-Zuweisungsoperator[4] wird die Historie in den Anfangszustand gesetzt („noch keine mit **insert()** eingefügten Werte vorhanden“).

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

Bei **clear** und **erase** bleibt die Historie unverändert.

Bei den Vergleichsoperationen[5] wird die Historie ignoriert.

Die Zeit- und Speicherkomplexität von **x()** muss O(1) sein. Die Zeit- und Speicherkomplexität aller übrigen Methoden und Funktionen müssen unverändert (spezifikationskonform) bleiben. Für die Implementierung der Lösung und die Verwendung der STL gelten dieselben Regeln wie im übrigen Projekt. Unter anderem müssen alle Instanzvariablen **private** sein, globale Variablen sind nicht zulässig, etc.

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.

---

[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(const ADS_set &other);
ADS_set &operator=(const ADS_set &other);
```

[3]
```
ADS_set();
ADS_set(std::initializer_list<key_type> ilist);
template<typename InputIt> ADS_set(InputIt first, InputIt last);
```

[4]
```
ADS_set &operator=(std::initializer_list<key_type> ilist);
```

[5]
```
friend bool operator==(const ADS_set &lhs, const ADS_set &rhs);
friend bool operator!=(const ADS_set &lhs, const ADS_set &rhs);
```