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

```
std::pair<size_t,size_t> y(size_t idx) const;
```

Diese soll Statistikdaten zu bestimmten Funktionen des **ADS_set** liefern. **first** enthält die Anzahl der Aufrufe der Funktion, **second** die Summe der Anzahl der Werte im **ADS_set** zum Zeitpunkt der Aufrufe. Abhängig vom Parameter **idx** sind Statistikdaten zu den folgenden Funktionen zu liefern (für **idx>2** ist das Ergebnis von **y()** undefiniert, keine Absicherung nötig):

idx==0: Methode **swap** **idx==1**: globale Funktion **swap** **idx==2**: Methode **clear**

Zu diesem Zweck sind die Statistikdaten im **ADS_set** zu verwalten und bei Aufrufen der drei Funktionen entsprechend anzupassen. Bei Aufrufen von **swap** sind die Statistikdaten für **beide** betroffenen **ADS_set**s zu aktualisieren. Achtung: es sind nur „externe" Aufrufe der Funktionen zu zählen. Externe Aufrufe sind Aufrufe von außerhalb des **ADS_set** (also hier vom Unit-Test), nicht hingegen Aufrufe der Funktionen durch andere Funktionen des **ADS_set** („interne" Aufrufe).

Beim Kopierkonstruktor und Kopierzuweisungsoperator

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

werden die Statistikdaten aus **other** übernommen.

Bei allen anderen Konstruktoren sowie beim Initializer-List-Zuweisungsoperator

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

werden die Statistikdaten in den Anfangszustand (alles 0) gesetzt.

Bei **swap**-Operationen werden die Statistikdaten der beiden **ADS_set**s getauscht (**nach** der Aktualisierung). Bei **insert** und **erase** bleiben die Statistikdaten unverändert. Bei den Vergleichsoperatoren (**==**, **!=**) werden die Statistikdaten ignoriert.

Die Zeitkomplexität von **y()** muss O(1) sein, die Speicherkomplexität O(1). Die Zeit- und Speicherkomplexität aller übrigen Funktionen (inklusive Methoden) müssen unverändert (spezifikationskonform) bleiben. Für die Verwendung der STL gelten dieselben Regeln wie im übrigen Projekt. Insbesondere müssen alle Instanzvariablen **private** sein. **std::pair** ist für diese Aufgabe als Bestandteil der Datenstruktur (Instanzvariable) erlaubt (falls benötigt).

Hinweis: Falls eine Funktion sowohl extern als auch intern aufgerufen wird, kann man die Unterscheidung zB durch Hinzufügen eines weiteren Parameters realisieren. Dieser Parameter erhält einen Vorgabewert (für externe Aufrufe) und einen anderen Wert bei internen Aufrufen.

Beispiel:

```
ADS_set s {1,2,3,4};        // s.size()==4
s.clear();
s.insert({1,2});            // s.size()==2
s.clear();
```

Dann liefert **s.y(2)** → **{2,6}** (2 Aufrufe von **clear** mit insgesamt 4+2=6 Elementen)

```
ADS_set t {5,6}, u{7,8,9}; // t.size()==2 u.size()==3
t.swap(u);
```

Dann liefert **t.y(0)** → **{1,3}** und **u.y(0)** → **{1,2}** (Statistikdaten wurden getauscht)