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

```
size_t y() const;
```

Diese soll die Länge der längsten, zusammenhängenden, aufsteigend sortierten Teilfolge in der durch einen Iterator gelieferten Reihenfolge der Schlüsselwerte liefern.

Für den Vergleich von Werten ist **`std::less`** (bzw. der alias **`key_compare`**, sofern vorhanden) zu verwenden. Der Aufruf **`std::less<T>{}(x,y)`** für die beiden Werte **`x`** und **`y`** vom Typ **`T`** liefert **`true`**, falls **`y`** größer als **`x`** ist, und **`false`** sonst. Aufruf von anderen Methoden oder Funktionen, insbesondere die Verwendung von Iteratoren (und damit z. B. auch die Verwendung einer range based for loop), ist nicht erlaubt.

Hinweis: auch **`size()`** ist eine Methode und daher verboten. Es gibt aber sicher eine passende Instanzvariable.

Die Zeitkomplexität der Funktion **`y`** muss O(n) sein (n ist dabei die Anzahl der Elemente im Set), die Speicherkomplexität O(1). Es ist beispielsweise nicht erlaubt, zusätzliche Felder mit einer nicht konstanten Größe zu verwenden.

Beispiele:

Angenommen der Iterator liefert alle gespeicherten Schlüsselwerte in der Reihenfolge	dann liefert der Aufruf von y()	denn die längste(n) aufsteigend sortierte(n) Teilfolge(n) ist/sind
(4,7,1,5,3,6,0,8,10,2,9)	3	(0,8,10)
(9,1,3,5,6,7)	5	(1,3,5,6,7)
(7,8,9,4,1,5,10)	3	(7,8,9) und (1,5,10)
(7,8,9)	3	(7,8,9)
(7,4)	1	(7) und (4)
(9)	1	(9)
()	0	()

Tipp: Eine einfache Methode ist, die Werte in Iteratorreihenfolge zu durchlaufen (allerdings, ohne Iteratoren zu verwenden) und dabei die Längen der gefundenen, aufsteigend sortierten Teilfolgen zu ermitteln. Eine aufsteigend sortierte Teilfolge ist daran zu erkennen, dass jeder Wert in der Teilfolge größer als sein Vorgänger (so vorhanden) ist.