- Studierendenausweis auf dem Platz hinterlegen
- Elektronische Geräte aller Art **ausschalten** und **unerreichbar verstauen**.
- Einloggen
- Lesen und Beachten der Regeln, die auf der Webseite angeführt sind.
- Änderungen in Ihrem Programm durchführen (Edit Button)
- Hochladen und Testen (Abschicken Button)
- Sichern ohne Test ist ebenfalls möglich (Speichern Button rechts oben)
- Um mit Ctrl-F im Programmtext zu suchen, müssen Sie zuerst in das Editorfenster klicken, da sonst eventuell die Browsersuche aktiviert wird.
- Hilfreiche Tastenkombinationen: Alt-Pos1 (Startseite), Strg-W (Fenster schließen) – Achtung: nicht gespeicherte Änderungen gehen bei beiden Tastenkombinationen verloren
- Sobald Sie SUCCESS erreichen, melden Sie sich bitte **unbedingt** bei der Prüfungsaufsicht, bevor Sie Ihren Arbeitsplatz verlassen. Andernfalls ist Ihr Ergebnis ungültig.
- Wenn kein SUCCESS erreicht wird, kann die Klausur zum nächsten Termin wiederholt werden.
- Programme, die SUCCESS erreichen, werden noch auf Korrektheit (z.B. Ordnungsverhalten) geprüft. Ihr Projekt wird ebenfalls noch einem Korrektheits- (richtige Datenstruktur, Einhaltung der Spezifikation) und einem Plagiatscheck unterzogen. Falls all diese Prüfungen auch erfolgreich durchlaufen werden, werden die Punkte für das Projekt vergeben.

Verwendung zusätzlicher Hilfsmittel bzw. Kommunikation mit NachbarInnen wird als Versuch gewertet, die Leistung zu erschleichen.

Erweitern Sie Ihre Klasse **ADS_set** um die Methode
const_iterator x(const key_type& k) const;
Diese soll einen Iterator auf das Maximum aller in der Datenstruktur gespeicherten Werte, die kleiner als der Parameterwert **k** sind, retournieren („nächstkleinerer Wert zu k“). Gibt es zum Zeitpunkt des Aufrufs von **x** keinen in der Datenstruktur gespeicherten Wert, der kleiner als **k** ist, so ist der end()-Iterator zu retournieren.

Zum Vergleich zweier Werte vom Typ **key_type** ist **std::less<key_type>** zu verwenden. 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.

Für die Suche nach dem Wert in der Datenstruktur ist die Verwendung von Iteratoren (und damit auch die Verwendung einer range based for loop) zum Iterieren über die in der Datenstruktur gespeicherten Werte nicht erlaubt. Die Zeitkomplexität der Funktion **x** muss O(n) sein, die Speicherkomplexität O(1). Es ist also beispielsweise nicht erlaubt, die Werte zu sortieren, auf ein eventuell vorhandenes Werte-Cache zuzugreifen, oder zusätzliche Felder mit einer nicht konstanten Größe zu verwenden.

Beispiel:
Gespeicherte Werte seien: C={7,1,5,3,0}. **x** wird mit dem Parameterwert 4 aufgerufen. Der zu retournierende Wert ist ein Iterator, der auf den Wert 3 verweist. (da 0, 1 und 3 kleiner als 4 sind und 3 von diesen Werten der maximale ist).

Eine mathematische Formulierung der Aufgabenstellung für jene, die diese Art der Beschreibung bevorzugen: Sei C die Menge der im Container gespeicherten Werte, k der Wert des Parameters und $M = \{w | w \in C \ \wedge std :: less < key\_type > \{\}(w,k) = true\}$

Dann liefert

$$x(k): \begin{cases} \text{Iterator, der auf } \max(M)\ verweist, falls\ M \neq \emptyset, \\ end() - Iterator\ sonst \end{cases}$$

Tipp:
Führen Sie eine übliche Maximumsuche durch, wobei Sie nur jene Werte berücksichtigen, die kleiner als der erhaltene Parameterwert sind. Verwenden Sie hierbei keine Iteratoren (oder range based for loops). Wird kein Maximum gefunden, retourniert **x** den end()-Iterator, andernfalls wird ein Iterator retourniert, der auf den gefundenen Maximalwert verweist.