- 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
key_type y() const;
Diese soll das Minimum der Werte liefern, die größer sind, als der Wert, für den **erase** zuletzt ohne Erfolg aufgerufen wurde. Erfolgte zum Zeitpunkt des Aufrufs der Methode **y** noch nie ein erfolgloser Aufruf von **erase** bzw. gibt es in der Datenstruktur keine größeren Werte als den beim letzten erfolglosen Aufruf von **erase** übergebenen Parameterwert, so ist eine Exception vom Typ **std::runtime_error** (notwendig: **#include <stdexcept>**) zu werfen. Es sind nur externe Aufrufe von **erase** zu berücksichtigen. Eventuelle interne Aufrufe durch andere Methoden von **ADS_set** (z.B. **clear**) sind zu ignorieren.
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.

Die Verwendung von Iteratoren (und damit auch die Verwendung einer range based for loop) zum Iterieren über die in der Datenstruktur gespeicherten Werte ist nicht erlaubt. Die Zeitkomplexität der Funktion **y** 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} der letzte, nicht erfolgreiche, externe Aufruf von **erase** sei mit dem Wert 3 erfolgt (3 wurde offenbar danach eingefügt). Ein Aufruf von z muss dann 5 liefern (Minimum der Werte 7 und 5, die größer als 3 sind).

Eine mathematische Formulierung der Aufgabenstellung für jene, die diese Art der Beschreibung bevorzugen: Sei C die Menge der im Container gespeicherten Werte und g der Parameterwert des letzten erfolglosen **erase** Aufrufs ($g = null$, falls **erase** noch nie erfolglos aufgerufen wurde). Für $g \neq null$ sei $A = \{x | x \in C \ \wedge std :: less < key\_type > \{\}(g, x) = true\}$. Dann liefert

$$y(): \begin{cases} \min(A) \; falls \; g \neq null \ \wedge \ A \neq \emptyset \\ eine \; Exception \; vom \; Typ \; std :: runtime\_error \; sonst \end{cases}$$

Tipp:
Sie benötigen *zumindest* eine weitere Instanzvariable in Ihrer Klasse, um sich zu merken, ob **erase** erfolglos aufgerufen wurde und wenn ja, mit welchem Wert $\boldsymbol{g}$ der letzte erfolglose Aufruf erfolgte. Diese Information ist bei jedem externen Aufruf von **erase** entsprechend zu pflegen und kann dann in der Methode **y** verwendet werden. **y** wirft eine Exception vom Typ **std::runtime_error** , falls **erase** noch nie erfolglos (von extern) aufgerufen wurde. Andernfalls ermittelt **y** das Minimum aller Werte $\boldsymbol{x}$ in der Datenstruktur, für die $\boldsymbol{std :: less < key\_type > \{\}(g, x)}$ den Wert **true** liefert. Wird kein solcher Wert gefunden, so ist eine Exception vom Typ **std::runtime_error** zu werfen. Andernfalls retourniert **y** das ermittelte Minimum. (Achtung: Die Verwendung von Iteratoren oder einer range based for loop zum Durchlaufen der gespeicherten Werte ist nicht erlaubt!)
Die Informationen bezüglich des zuletzt fehlgeschlagenen Aufrufs von **erase** werden beim Kopieren von Objekten (also im Kopierkonstruktur und Kopierzuweisungsoperator) oder bei swap-Operationen nicht übernommen. Die Informationen werden im neu erstellten Objekt auf den Anfangszustand gesetzt (Kopierkonstruktor) bzw. bleiben unverändert (Kopierzuweisungsoperator und swap-Operation).