Aufgabe 1

Spezifizieren Sie die rationalen Zahlen als abstrakten Datentyp ohne Axiome. Eine rationale Zahl besteht aus einem Zähler und einem Nenner, die beide ganze Zahlen sind. Den Typ INT der ganzen Zahlen können Sie (mit + , - , *, /, ggt) als gegeben voraussetzen.

Definieren Sie (in dem im Buch benutzten Format)

- einen Konstruktor mk_rat, der aus zwei ganzen Zahlen eine rationale Zahl macht,
- die Selektoren nominator und denominator,
- ein Prädikat is_zero, das überprüft, ob eine rationale Zahl gleich 0 ist,
- ein Prädikat equal, das prüft, ob zwei rationale Zahlen gleich sind, d.h. den gleichen Wert haben,
- die Addition add, Subtraktion sub, Multiplikation mult sowie Division div auf rationalen Zahlen und
- eine Funktion normalize, die eine rationale Zahl so transformiert, daß ihr Wert gleich bleibt, aber der Betrag des Nenners möglichst klein ist.

Aufgabe 2

Der Datentyp *D*atum soll die Funktionen

```
mk_date:  erzeugt ein Datum
tag:      ermittelt Tag aus Datum
monat:    ermittelt Monat aus Datum
jahr:     ermittelt Jahr aus Datum
next:     bestimmt Datum des Folgetages
previous: bestimmt Datum des vorhergehenden Tages
plus:     addiert eine Anzahl Tage zu vorgegebenem Datum
between:  ermittelt Differenz in Tagen zwischen zwei Datumsangaben
is_before:Prädikat zum Vergleich zweier Datumsangaben
equal:    Prädikat zur Feststellung der Gleichheit zweier Datumsangaben
```

beinhalten.

a) Spezifieren Sie *D*atum als ADT in dem im Buch benutzten Format ohne Axiome.

b) Implementieren Sie die Klasse *D*atum in Java ohne Verwendung von existierenden Klassen. Zur Vereinfachung kann angenommen werden, daß alle Monate 30 Tage haben.

Aufgabe 3

Spezifizieren Sie den den abstrakten Datentyp SortSeq[X] mit den folgenden Funktionen sowie passenden Axiomen:

```
mt_SortSeq:        leere Sequenz
cons:              fügt Element vorn ein (hidden function)
insert:            sortiertes Einfügen eines neuen Elementes, doppelte Elemente sind erlaubt
del:               entfernt das angegebene Element (auch mehrmals möglich)
contains:          prüft, ob gegebenes Element enthalten ist
isEmpty:           prüft, ob die Sequenz leer ist
size:              Anzahl der Einträge
min:               kleinstes Element
max:               größtes Element
clear:             löscht Sequenz
```

SortSeq ist eine Sequenz mit Einträgen vom Typ X. Für den Typ X muss eine lineare Ordnung erklärt sein, wie sie beispielsweise für die Datentypen Integer, Float, Character und String existiert. Doppelte Einträge sind erlaubt.

Aufgabe 4

Implementieren Sie die Datenstruktur Deque (double-ended Queue), die das Anfügen von Objekten an beiden Enden sowie das Navigieren über alle Elemente mit Hilfe eines Iterators erlaubt. Verwenden Sie dabei keine der existierenden Java-Klassen !