Aufgaben zur Vorbereitung auf Testat 4

Als Vorbereitung auf das Testat 4 solltest Du unbedingt diese Aufgaben bearbeiten.
Ein Teil der Aufgaben sind bereits von den vorangehenden Praktikumsblättern bekannt.

Bei der Implementierung der Methoden soll hier auf Schleifen verzichtet und ausschließlich Rekursion genutzt werden.

Rekursion mit Feldern von Objekten

1 - Bestimmung des Maximums

Entwickle eine *rekursive* Methode `Fraction maximum( Fraction[] arr, int i )`, die für ein Feld `arr` das Maximum im Bereich von `arr[0]` bis `arr[i]` mit `0<=i<arr.length` bestimmt und zurückgibt. Ist das Feld leer, soll `null` zurückgegeben werden.

2 - Bestimmung des letzten positiven Bruchs

Entwickle eine *rekursive* Methode `Fraction lastPositive( Fraction[] arr, int i )`, die für ein Feld `arr` im Bereich von `arr[0]` bis `arr[i]` mit `0<=i<arr.length` den Bruch bestimmt und zurückgibt, dessen Wert positiv ist und der am größten Index in `arr` abgelegt ist. Gibt es keinen solchen Bruch, soll `null` zurückgegeben werden.

3 - Bestimmung des ersten positiven Bruchs

Entwickle eine *rekursive* Methode `Fraction firstPositive( Fraction[] arr, int i )`, die für ein Feld `arr` im Bereich von `arr[0]` bis `arr[i]` mit `0<=i<arr.length` den Bruch bestimmt und zurückgibt, dessen Wert positiv ist und der am kleinsten Index in `arr` abgelegt ist. Gibt es keinen solchen Bruch, soll `null` zurückgegeben werden.

4 - Prüfen einer Sortierung

Entwickle eine *rekursive* Methode `boolean isSorted( Fraction[] arr, int i )`, die für ein Feld `arr` im Bereich von `arr[0]` bis `arr[i]` mit `0<=i<arr.length` feststellt, ob das Feld aufsteigend sortiert ist, also für alle `k` mit `0<=k<i` gilt: `arr[k]<=arr[k+1]`. Ist das der Fall, soll `true` zurückgegeben werden, sonst `false`.

5 - Inhaltsüberprüfung

Entwickle eine *rekursive* Methode `boolean contains( Fraction[] arr, int i, Fraction x )`, die für ein Feld `arr` im Bereich von `arr[0]` bis `arr[i]` mit `0<=i<arr.length` bestimmt, ob dort der Wert `x` vorkommt. Ist das der Fall, soll `true` zurückgegeben werden, sonst `false`.

6 - Zählen von positiven Brüchen

Entwickle eine *rekursive* Methode `int countPositives( Fraction[] arr, int d, int t )`, die für ein Feld `arr` im Bereich von jeweils einschließlich `arr[d]` bis `arr[t]` mit `d<=t<arr.length` die Anzahl der positiven Brüche bestimmt und zurückgibt.

7 - Gleichheit von Feldinhalten

Entwickle eine *rekursive* Methode `boolean contentCheck( Fraction[] arr1, Fraction[] arr2, int i )`, die für die beiden als Parameter übergebenen Felder feststellt, ob die Folgen der Brüche im Bereich der Indizes von `0` bis `i` mit `0<=i<arr.length` gleich sind. Die Methode soll einen Wert des Typs `boolean` zurückgeben.

8 - Palindrome

Ein Palindrom ist eine Folge von Brüchen, die vorwärts und rückwärts gelesen die identischen Werte liefert.

Entwickle eine *rekursive* Methode `boolean palindromCheck( Fraction[] arr, int i )`, die durch den Aufruf `palindromCheck( a, 0 )` für ein Feld `arr` von Brüchen ermittelt, ob die Folge der Brüche in `arr` ein Palindrom bildet. Der Parameter `i` soll dazu benutzt werden, den betrachteten Bereich des Feldes `arr` einzuschränken.

9 - Ermitteln des Index

Vervollständige die Methode `int getIndex( Fraction[] arr, int i, Fraction x )`, die den *kleinsten* Index zurückgibt, an dem der Bruch `x` im Bereich von `arr[0]` bis `arr[i]` mit `0<=i<arr.length` vorkommt. Kommt `x` nicht vor oder wird für `i` ein Wert außerhalb der Grenzen des Felds `arr` übergeben, soll `-1` zurückgegeben werden.