Aufgaben zur Vorbereitung auf Testat 3

Als Vorbereitung auf das Testat 3 solltest Du unbedingt diese Aufgaben bearbeiten.
Ein Teil der Aufgaben sind bereits von den vorangehenden Praktikumsblättern bekannt.

Jetzt soll bei der Implementierung der Methoden auf Schleifen verzichtet und ausschließlich Rekursion genutzt werden.

1 - Quersumme

Entwickle eine *rekursive* Methode `int digitSum( int n )`, die für einen `int`-Parameter `n` die Quersumme des Wertes von `n` berechnet und zurückgibt.

Hinweis: Die Quersumme einer Zahl ergibt sich aus der Addition der letzten Ziffer mit der Summe der restlichen Ziffern.

2 - Potenz

Entwickle eine *rekursive* Methode `int power( int a, int n )`, die für zwei `int`-Parameter `a` und `n` die `n`-te Potenz des Wertes von `a` berechnet und zurückgibt.

3 - Werte aufsummieren

Entwickle eine *rekursive* Methode `int sumUpNegatives( int[] arr, int n )`. Die Methode `sumUpNegatives` bildet die Summe der negativenWete im Feld `arr` im Bereich von `arr[0]` bis `arr[n]` mit `0<=n<arr.length` und gibt die ermittelte Summe zurück.

4 - Zählen von positiven Werten

Entwickle eine *rekursive* Methode `int countPositives( int[] arr, int n )`, die für ein Feld `arr` die Anzahl der positiven Werte im Bereich von `arr[0]` bis `arr[n]` mit `0<=n<arr.length` bestimmt und zurückgibt.

5 - Zählen von positiven Werten in einem Teil des Feldes

Entwickle eine *rekursive* Methode `int countPositivesLimited( int[] 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 Werte bestimmt und zurückgibt.

6 - Bestimmung des Maximums

Entwickle eine *rekursive* Methode `int maximum( int[] 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.

7 - Prüfen einer Sortierung

Entwickle eine *rekursive* Methode `boolean isSorted( int[] 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, wird `true` zurückgegeben, sonst `false`.

8 - Inhaltsüberprüfung

Entwickle eine *rekursive* Methode `boolean contains( int[] arr, int i, int 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, wird `true` zurückgegeben, sonst `false`.

9 - Gleichheit von Feldinhalten

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

10 - Palindrome

Ein Palindrom ist eine Zeichenfolge, die vorwärts und rückwärts identisch gelesen werden kann.

Entwickle eine rekursive Methode `boolean palindromCheck( char[] arr, int i )`, die ermittelt, ob die Folge der Zeichen im Feld `arr` ein Palindrom bildet. Der Parameter `i` soll dazu benutzt werden, den betrachteten Bereich des Feldes `arr` einzuschränken. Die Methode gibt einen Wert des Typs `boolean` zurück.

11 - Ermitteln des Index

Entwickle eine *rekursive* Methode `int getIndex( int[] arr, int i, int x )`, die den *kleinsten* Index zurückgibt, an dem der Wert `x` in `arr` 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, wird `-1` zurückgegeben.

12 - Erzeugen eines Textes

Entwickle eine *rekursive* Methode `String toString( int[] arr, int i )`. Die Methode `toString` erzeugt einen Text, der alle Werte des Feldes im Bereich von `arr[0]` bis `arr[i]` mit `0<=i<arr.length` in der Reihenfolge ihres Auftretens durch Kommas getrennt enthält. Der erzeugte Text wird zurückgegeben.