Praktikumstutorium – Blatt 4

Als Vorbereitung auf das Testat 4 sollten Sie unbedingt diese Aufgaben bearbeiten.
Sofern dabei Schwierigkeiten auftreten, sollten Sie unbedingt das **Tutorium des Programmierpraktikums** besuchen

Tutoriumszeiten im Raum OH 12/4.030

Montag	**Dienstag**	**Mittwoch**	**Donnerstag**	**Freitag**
12.00 – 14.00 Uhr	10.00 – 16.00 Uhr	10.00 – 14.00 Uhr	10.00 – 14.00 Uhr	*Testat*

Rekursion

Verzichten Sie allen Lösungen völlig auf die Benutzung von `for`- oder `while`-Schleifen!

1 - Quersumme

Entwickeln Sie eine rekursive Methode `int digitSum( int n )`, die für einen `int`-Parameter `n` die Quersumme des Wertes von `n` berechnet.

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

2 - Harmonische Zahl

Entwickeln Sie eine rekursive Methode `double harm( int n )`, die für einen positiven `int`-Parameter `n` die `n`-te harmonische Zahl, d.h. die Summe der ersten `n` Brüche *1/n*, als reelle Zahl zurückgibt. Es soll also die folgende Summe gebildet werden: *1/1 + 1/2 + 1/3 + ... 1/n*.

Beispiel: `harm(5)` ergibt *1/1+1/2+1/3+1/4+1/5=2,28*

3 - Potenz

Entwickeln Sie 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.

4 - optimierte Berechnung der Potenz

Entwickeln Sie eine weitere rekursive Methode `int powerPlus( int a, int n )`, die für zwei `int`-Parameter `a` und `n` die `n`-te Potenz des Wertes von `a` berechnet.

Dabei soll die Zahl der rekursiven Aufrufe und damit der Berechnungen klein gehalten werden, indem die Rechnung folgendermaßen ausgeführt wird:

$a^n = 1$ für $n = 0$;

$a^n = a^{n/2} * a^{n/2}$ für gerade n;

$a^n = a * a^{(n-1)/2} * a^{(n-1)/2}$ für ungerade n

Hinweis: Überlegen Sie zunächst, wie mit der oben stehenden Formel die Zahl der Berechnungen reduziert werden kann.

5 - Binärdarstellung

Entwickeln Sie eine rekursive Methode `String binaryCode( int i )`, die für einen positiven `int`-Wert dessen Binärdarstellung als String aus den Werten `0` und `1` bestimmt und diesen String zurückgibt.

Hinweis: Die Binärdarstellung einer Dezimalzahl ergibt sich aus den rückwärts zusammengefassten Resten einer wiederholt ausgeführten Division der Zahl durch *2*.

Beispiel: *6/2 = 3 Rest 0, 3/2 = 1 Rest 1, 1/2 = 0 Rest 1* ergibt also die Binärdarstellung `110` für den Dezimalwert *6*.

Rekursion mit Feldern

Verzichten Sie allen Lösungen völlig auf die Benutzung von `for`- oder `while`-Schleifen!

~~1 - Bestimmung des Maximums~~

Entwickeln Sie 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.

~~2 - Prüfen einer Sortierung~~

Entwickeln Sie 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, soll **`true`** zurückgegeben werden, sonst **`false`**.

3 - Inhaltsüberprüfung

Entwickeln Sie 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, soll **`true`** zurückgegeben werden, sonst **`false`**.

4 - Zählen von positiven Werten

Entwickeln Sie eine rekursive Methode **`int`** `countPositives(` **`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.

5 - Gleichheit von Feldinhalten

Entwickeln Sie 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 soll einen Wert des Typs **`boolean`** zurückgeben.

6 - Palindrome

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

Entwickeln Sie eine rekursive Methode **`boolean`** `palindromCheck(` **`char`**`[] arr,` **`int`** `i )`, die durch den Aufruf `palindromCheck( a, 0 )` für ein Feld `a` von Zeichen ermittelt, ob die Folge der Zeichen in `a` ein Palindrom bildet. Der Parameter `i` soll dazu benutzt werden, den betrachteten Bereich des Feldes `arr` einzuschränken. Die Methode soll einen Wert des Typs **`boolean`** zurückgeben.