Westfälische Hochschule

Prof. Dr. Manfred Meyer
Fachbereich 6 / Bocholt
Professur BWL / Wirtschaftsinformatik

Algorithmen und Datenstrukturen
(Studiengang Wirtschaftsinformatik 2. Semester)

Aufgabenstellung für das Praktikum am 26.10.2023

Aufgabe 5:
Erstellen Sie ein Java-Programm zur Ausgabe aller vollkommenen Zahlen (auch *perfekte Zahlen* genannt) zwischen 1 und einer einzugebenden Obergrenze **n**.
Testen Sie Ihr Programm und analysieren Sie es im Hinblick auf Terminierung, Korrektheit und Komplexität (Speicher- und Laufzeitverhalten in Abhängigkeit von **n**).

Aufgabe 6:
Erstellen Sie ein Java-Programm zur Berechnung aller Primzahlen zwischen 1 und einer einzugebenden Obergrenze **n**, das (nur ausnahmsweise und nur zu Übungszwecken!) nur partiell korrekt (also nicht total korrekt) ist.

Aufgabe 7:
Untersuchen Sie die folgenden in Java formulierten Algorithmen hinsichtlich Terminierung und Komplexität (Laufzeit und Speicherplatz). Was kann man zu deren Korrektheit sagen?

Um Ihre Überlegungen oder Vermutungen zu bestätigen, messen Sie mit Hilfe der Methode `System.currentTimeMillis()` die Laufzeit der Algorithmen bei unterschiedlichen ansteigenden Werten für n – diese müssen u.U. schon einigermaßen groß werden; für zu kleine Werte sind die Laufzeiten möglicherweise zu klein, so dass Sie keine hilfreichen Ergebnisse erhalten! Ermitteln Sie außerdem die jeweilige Anzahl der Durchläufe insgesamt (Anzahl der Aufrufe der Operation `System.out.println()`) für die untersuchten unterschiedlichen Werte für n.

a)
```
static void f1(int n)
{
      for (int i=0; i<n; i++)
      {
            int[] a=new int[n];
            while (i<n)
            {
                  a[i]=i;
                  System.out.println(i++);
            }
      }
}
```

b)
```
static void f2(int n)
{
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < n; j++)
		{
			int[] a = new int[n];
			a[j] = j;
			System.out.println(j);
		}
	}
}
```

c)
```
static void f3(int n)
{
	int i = 0;
	int j = 0;
	int k = 0;
	while (i < n)
	{
		while (j < n)
		 {
			while (k < n)
			{
				System.out.println(k++);
			}
			j++;
		}
		i++;
	}
}
```

d)
```
static void f4(int n)
{
	int i = 0;
	int j = 0;
	while (i < n)
	{
		System.out.println(i++);
		i = i * j;
	}
}
```

e)
```
static void f5(int n)
{
	int i = 1;
	int j = 1;
	int k = 1;
	while (i < n)
	{
		while (j < n)
		{
			int[] a = new int[i * j * k];
			k = 0;
			while (k < n) {
				System.out.println(k++);
				a[i - 1] = j;
			}
			j++;
		}
		i++;
	}
}
```

Aufgabe 8:
Untersuchen Sie durch Messungen das Laufzeitverhalten des folgenden Algorithmus für immer größer werdende Parameter n. Sie müssen den Algorithmus selbst in seiner Funktionsweise noch nicht verstehen (das kommt später noch):

```
static int f(int n)
{
	if (n < 2)
		return 1;
	else
		return f(n-1) + f(n-2);
}
```

Haben Sie eine Vermutung zur Laufzeitkomplexität dieses Algorithmus? Können Sie diese vielleicht sogar begründen?

Viel Spaß und Erfolg bei der Bearbeitung der Aufgaben!