Prof. Dr. A. Weber
Dr. Hassan Errami

Universität Bonn
Institut für Informatik II
24.10.2018

Wintersemester 2018/2019

Übungen zur Vorlesung
Algorithmisches Denken und imperative Programmierung (BA-INF-014)
Aufgabenblatt 2

Zu bearbeiten bis: 02.11.2018

Aufgabe 1 *(Addition von kleinen Zahlen - 4 Punkte)*

Kompilieren und starten Sie das folgende C-Programm.

```
#include <stdio.h>

int main(){

    char x1,x2,result;

    // Beispiel 1:
    x1 =  35;
    x2 =  85;
    result =  x1 + x2;
    printf("Beispiel 1: %hi + %hi = %hi\n",x1 ,x2, result);

    // Beispiel 2:
    x1 =  85;
    x2 =  85;
    result = x1 + x2;
    printf("Beispiel 2: %hi + %hi = %hi\n",x1 ,x2, result);

    return 0;
}
```

Erklären Sie die Ergebnisse des Programms!

Aufgabe 2 *(Fibonacci-Zahlen - 2+2=4 Punkte)*

Die Fibonacci-Folge f_n ist eine unendliche Folge von Zahlen (den Fibonacci-Zahlen), bei der sich die jeweils folgende Zahl durch Addition ihrer beiden vorherigen Zahlen ergibt: $0, 1, 1, 2, 3, 5, 8, 13, \ldots$. Das rekursive Bildungsgesetz ist wie folgt definiert:

$$f_n = \begin{cases} 0 & \text{falls } n = 0 \\ 1 & \text{falls } n = 1 \\ f_{n-1} + f_{n-2} & \text{falls } n \geq 2 \end{cases}$$

- Implementieren Sie in C eine Funktion **long fib_rec(long n)**, welche f_n berechnet.
- Implementieren Sie in C eine Funktion, die die ersten 50 geraden Fibonacci-Zahlen ausgibt.

Aufgabe 3 *(Perfekte Zahlen und Defiziente Zahlen - 4 Punkte)*

Eine natürliche Zahl heißt,

- vollkommen (auch perfekt), wenn sie gleich der Summe aller ihrer (positiven) echte Teiler ist (die Summe aller Teiler ohne die Zahl selbst).
- defizient, wenn ihre echte Teilersumme kleiner ist als die Zahl selbst.

Z.B.: 6 ist eine vollkommene Zahl, weil $6 = 3 + 2 + 1$ und 10 ist eine defiziente Zahl, weil $1 + 2 + 5 < 10$.

Schreiben jeweils ein Programm für folgende Aufgaben:

- Testen, ob eine natürliche Zahl n vollkommen ist.
- Testen, ob eine natürliche Zahl n defizient ist.

- Ausgabe aller vollkommenen Zahlen, die kleiner als eine natürliche Zahl r sind.
- Berechnung der Anzahl von defizienten Zahlen, die kleiner als eine natürliche Zahl r sind.

Testen Sie Ihre Programme für $n = 14, 18, 25, 28$ und 51 und für $r = 499$.

Aufgabe 4 *(Verschachtelte Schleife - 4 Punkte)*

Schreiben Sie ein C-Programm, welches das gesamte kleine Einmaleins ausgibt. Hier müssen zwei ineinander verschachtelte Schleifen verwendet werden. Die Ausgabe soll dabei etwa folgende Form haben:
1 x 1 = 1
2 x 1 = 2
3 x 1 = 3
...
9 x 1 = 9
10 x 1 = 10
1 x 2 = 2
2 x 2 = 4
...
9 x 10 = 90
10 x 10 = 100

Aufgabe 5 *(Potenzierung - 2+2=4 Punkte)*

Es seien $a \in \mathbf{R}$ und $n \in \mathbf{N}$. Schreiben Sie jeweils ein iteratives Programm zur Berechnung von a^n, das die nachfolgenden Verfahren verwendet. Geben Sie auch an, wieviele Schritte jedes Verfahren braucht, um a^{17} zu berechnen.

a) $a^n = a \cdot a^{n-1}$

b)

$$a^n = \begin{cases} 1 & \text{falls } n = 0 \\ a^{\frac{n}{2}} \cdot a^{\frac{n}{2}} & \text{falls } n \mod 2 = 0 \\ a \cdot a^{\frac{n-1}{2}} \cdot a^{\frac{n-1}{2}} & \text{falls } n \mod 2 \neq 0 \end{cases}$$

Tipp: Beachten Sie, wie die mathematische Vorschrift auf Folie 37 im Foliensatz zu Algorithmen auf Folie 44 unten implementiert wird. Die Implementierung der o.g. Vorschrift erfolgt sehr ähnlich. Der Modulo-Operator wird in C mit dem Zeichen % verwendet.