Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

TU Kaiserslautern
Fachbereich Informatik

Lösungshinweise zum Übungsblatt 3: Programmieren in C (WS 2018/19)

1. Zur Beantwortung von Fragen und Hilfe bei Problemen stehen wir Ihnen immer in der Praktische Übung, Mittwoch 15:30, Raum 32-410-PC zur Verfügung sowie per Email an `pinc-support@cs.uni-kl.de`!
2. Sie können sich im Exclaim System unter `https://softech.cs.uni-kl.de/exclaim` mit Ihrem persönlichen STATS-Account einloggen und Dateien zu den einzelnen Übungen hochladen.
3. Die Aufgaben auf diesem Übungsblatt sind alle freiwillig zu bearbeiten.

Aufgabe 1: Kleinste Zahl

Abgabe: Exclaim als `min3.c`

Erstellen Sie ein C-Programm, das drei ganze Zahlen einliest und die kleinste dieser Zahlen ausgibt.

1. Geben Sie einen Algorithmus zur Berechnung der kleinsten Zahl zunächst als Flussdiagramm an.
2. Implementieren Sie Ihren Algorithmus als C-Programm `min3.c`.

Die Ausgabe des Programms soll folgendermaßen gestaltet sein:

```
Das Minimum von 5, -4, 7 ist -4.
```

```c
#include<stdio.h>

int main(void)
{
    int x,y,z, min;
    scanf("%d", &x);
    scanf("%d", &y);
    scanf("%d", &z);

    if (x < y)
        if (x < z) min = x;
        else min = z;
    else if (y < z) min = y;
        else min = z;

    printf("Das Minimum von %d, %d, %d ist %d.\n", x, y, z, min);
    return 0;
}
```

Aufgabe 2: Summe berechnen

Abgabe: Exclaim als `summe.c`

Erstellen Sie ein C-Programm, das eine positive ganze Zahl n einliest und die Zahlen 1, 2,..., n aufaddiert.

1. Geben Sie einen Algorithmus zur Berechnung der Summe zunächst als Flussdiagramm an.

2. Implementieren Sie Ihren Algorithmus als C-Programm `summe.c`.

Die Ausgabe des Programms soll bei Eingabe `5` folgendermaßen gestaltet sein:

```
Die Summe von 1 bis 5 ist 15.
```

```c
// Summe aller Zahlen von 1 bis n (Eingabe)

#include <stdio.h>

int main(void)
{
    int n, summe;
    scanf("%d", &n);

    summe = 0;
    for (int i = 1; i <= n; i++) {
        summe += i;
    }
    printf("Die Summe von 1 bis %d ist %d.\n", n, summe);
    return 0;
}
```

Aufgabe 3: Binäre Zahlendarstellung (keine Abgabe)

Abgabe: –

1. Ergänzen Sie folgende Tabelle, sodass jede Zeile den gleichen Zahlenwert in dem jeweiligen Zahlenformat hat.

Bitmuster(8B)	Int	Unsign.Int.	Hex	Okt	ASCII
10010110					
	127				
		179			
			FB		
				64	
					U

Bitmuster(8B)	Int	Unsign.Int.	Hex	Okt	ASCII
1001 0110	-106	105	96	226	-
0111 1111	127	127	7F	177	DEL
1011 0011	-77	179	B3	263	3
1111 1011	-5	251	FB	373	$\hat{U}$
0011 0100	52	52	34	64	4
0101 0101	85	85	55	125	U

2. Rechnen Sie die folgende Zahlen direkt - ohne über Dezimalzahlen zu gehen dafür aber über Binärzahlen - in die rechts des Zuweisungspfeils angegebenen Zahlenformate (Basen) um.

   - $1001001010111111_2 \Rightarrow$ _____$_8$

     $\Rightarrow 111277_8$

   - $1001001010111111_2 \Rightarrow$ _____$_{16}$

     $\Rightarrow 92BF_{16}$

   - $CBAF_{16} \Rightarrow$ _____$_8$

     $\Rightarrow 1100101110101111_2 \Rightarrow 145657_8$

   - $726_8 \Rightarrow$ _____$_{16}$

$\Rightarrow 111010100110_2 \Rightarrow EA6_{16}$

3. Welche Zahlen sind hier als single-precision float-Zahlen dargestellt?
   - 01000000111000000000000000000000
     - Vorzeichen: +
     - Exponent: $10000001_2 = 129 - 127 = 2$
     - Mantisse: $1 + 2^{-1} + 2^{-2} = 1,75$
     - Ergebnis: $+1,75 \times 2^2 = 7$
   - 11000011010110000000000000000000
     - Vorzeichen: -
     - Exponent: $10000110_2 = 134 - 127 = 7$
     - Mantisse: $1 + 2^{-1} + 2^{-3} + 2^{-4}$
     - Ergebnis: $-(1 + 2^{-1} + 2^{-3} + 2^{-4}) \times 2^7 = -(1^7 + 2^6 + 2^4 + 2^3) = -216$
4. Stellen Sie die folgenden Zahlen als single-precision float-Zahlen dar!
   - $178,625_{10}$

     $178,625_{10} = 10110010,101_2 = 1,0110010101_2 \times 2^7 = 0\ 10000110\ 01100101010000000000000$
   - 10010110_2

     $10010110_2 = 1,0010110_2 \times 2^7 = 0\ 10000110\ 00101100000000000000000$
   - $-152,8_{10}$

     $-152,8_{10} = -10011000,110011001100\ldots_2 = -1,0011000110011001100\ldots_2 \times 2^7 =$
     1 10000110 00110001100110011001100110011001
5. Lösen Sie die folgenden Rechenaufgaben in Binärarithmetik (Binär-Addition, d.h. keine Subtraktion)! Falls nötig, berechnen Sie hierzu zunächst das Zweierkomplement der gegebenen Dezimalzahlen.
   - 17 + 19

     ```
     17: 00010001
     19: 00010011
         --------
         00100100 : 36
     ```
   - 16 – 10

     ```
     16: 00010000
    -10: 11110110
         --------
       1 00000110 : 6
     ```
   - 18 – 23

     ```
     18: 00010010
    -23: 11101001
         --------
         11111011 : -5
     ```
   - (-8) – (-11)

```
-8: 11111000
11: 00001011
    --------
  1 00000011 : 3
```

Aufgabe 4: Wie teuer ist das Laden eines Smartphone-Akkus?

Abgabe: Exclaim als `smartphone.c`
Die meisten Smartphone-Akkus halten gerade mal einen Tag durch. Deshalb ist der Griff zum Ladegerät am Abend für viele Smartphone-Nutzer schon Routine. Aber was kostet es, wenn wir unser Smartphone jeden Tag aufladen?

Nehmen Sie hierzu an, dass Ihr Smartphone-Akku eine Kapazität von 3000 mAh bei einer Spannung von 3.85 V aufweist und täglich zu 80% geladen werden muss. Die Verluste des Ladegeräts können vernachlässigt werden. Eine kW-Stunde Strom kostet im Durchschnitt 30 Cent.

1. Geben Sie die mathematische Formel an, um die Kosten für das Laden des Smartphones pro Tag zu berechnen.
2. Implementieren Sie Ihre Formel als C-Programm `smartphone.c`. Das Programm nimmt als Eingabe die Anzahl an Tagen, für die die Kosten berechnet werden sollen.

Die Ausgabe des Programms soll bei Eingabe `365` folgendermaßen gestaltet sein:

```
Kosten: 1.0118 EUR
```

Hinweis: Die Ausgabe des Geldbetrags muss mit 4 Nachkommastellen erfolgen.

```c
// Kosten fuer Akkuaufladung

#include <stdio.h>

int main(void)
{
    int n;
    scanf("%d", &n);

    double kapazitaet = 3.;     //Ah
    double spannung = 3.85;    //V
    double faktor = 0.8;
    double kosten = 0.0003;    //Wh

    double pro_tag = kapazitaet * faktor * spannung * kosten;
    double gesamt = pro_tag * n;

    printf("Kosten: %.4f EUR\n", gesamt);

    return 0;
}
```