Übungen zu Computergrundlagen WS 2018/2019

Übungsblatt 9: Grundlagen der Programmierung

20. Dezember 2017

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 11.01.2019, 11:00 Uhr**
- Schickt die Lösungen bitte per Email an Euren Tutor:
  - Montag 14:00–15:30: Grant Cates (`gcates@icp.uni-stuttgart.de`)
  - Dienstag 9:45–11:15: Kai Szuttor (`kai@icp.uni-stuttgart.de`)
  - Dienstag 15:45–17:15: Julian Michalowsky (`jmichalowsky@icp.uni-stuttgart.de`)
  - Mittwoch 15:45–17:15: Michael Kuron (`mkuron@icp.uni-stuttgart.de`)
  - Donnerstag 9:45–11:15: Johannes Zeman (`zeman@icp.uni-stuttgart.de`)
- Die Übungen sollen in Gruppen von jeweils *zwei bis drei* Leuten bearbeitet werden. Abgaben von Einzelpersonen werden nicht akzeptiert. Bitte gebt *nur eine Lösung pro Gruppe* ab und nennt in eurer Abgabe alle Mitglieder eurer Gruppe!

Aufgabe 9.1: Zahlensysteme (5 Punkte)

- **9.1.1** Berechne die folgenden Zahlen a bis k, indem Du zwischen verschiedenen Zahlensystemen umrechnest. Dabei steht 1234_7 für die Zahl 1234 im Zahlensystem zur Basis 7. $a_{10} = 1234_7$ bedeutet also, daß die Zahl 1234 im Zahlensystem zur Basis 7 ins Zahlensystem zur Basis 10 (Dezimalsystem) umgerechnet werden soll. (3 Punkte)
- Du kannst Dir entweder ein `bash`-Skript schreiben, dass die Umrechnung für euch vornimmt, oder aber die Umrechnung per Hand durchführen.

  Hinweis: In einigen Fällen dürfte die nebenstehende Tabelle nützlich sein.

  - $a_{10} = 1234_7$
  - $b_{16} = 1234_7$
  - $c_{16} = 1234_{10}$
  - $d_8 = 1234_{10}$
  - $e_7 = 1234_{10}$
  - $f_2 = CD_{16}$
  - $g_2 = 27_8$
  - $h_8 = 10000001_2$
  - $i_8 = 10100101_2$
  - $j_{16} = 10000001_2$
  - $k_{16} = 10100101_2$

2	**7**	**8**	**10**	**16**
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	3	3	3	3
100	4	4	4	4
101	5	5	5	5
110	6	6	6	6
111	10	7	7	7
1000	11	10	8	8
1001	12	11	9	9
1010	13	12	10	A
1011	14	13	11	B
1100	15	14	12	C
1101	16	15	13	D
1110	20	16	14	E
1111	21	17	15	F
10000	22	20	16	10

- **9.1.2** Im Computerumfeld wird häufig das Hexadezimalsystem ($B = 16$) verwendet.
  - Welchen Vorteil bietet das System gegenüber dem Dezimalsystem ($B = 10$) im Computerumfeld? (1 Punkt)
  - Welchen Vorteil bietet es gegenüber dem Oktalsystem ($B = 8$) im Computerumfeld? (1 Punkt)

Hinweise:

- In `bash` kann ein String, der z. B. in der Variable `STRING` steht, mittels `echo $STRING | grep -o .` in ein Array von Zeichen umgewandelt werden.
- Mit `rev` kann die Reihenfolge umgekehrt werden.
- `bash` unterstützt nativ nur Integerarithmetik, besonders hilfreich sind dabei die *modulo* und die Divisions Operation `%` und `/`.
- Auch der Kommandozeilen-Taschenrechner `bc` kann verwendet werden:
  `result=$(echo ‘‘1+2’’ | bc -l)`
- Es kann hilfreich sein, schrittweise vorzugehen, also die Zahl zunächst beispielsweise im Dezimalsystem darzustellen um von dort aus in eine andere Basis zu wechseln.

Aufgabe 9.2: Interpretierte vs. kompilierte Programmiersprache (1 Punkt)

Beschreibe *kurz*:
Was ist der Unterschied zwischen interpretierten und kompilierten Programmiersprachen? Nenne jeweils ein Beispiel für eine solche Sprache. Gib je einen Vor- und Nachteil von interpretierten und kompilierten Sprachen an.

Aufgabe 9.3: Numerische Berechnung einer Fakultät (4 Punkte)

In dieser Aufgabe sollst Du Dir überlegen, wie man in einem Computerprogramm die Fakultät (englisch: “factorial”) einer natürlichen Zahl berechnen kann. Die Fakultät einer Zahl $n \in \mathbb{N}$ ist definiert als

$$n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n \,. \tag{1}$$

Einen Spezialfall stellt die Fakultät der Zahl Null dar, sie ist definiert als $0! = 1$.

- **9.3.1** (2 Punkte) Schreibe eine Funktion `recursive_factorial(n)`, die die Fakultät einer an sie übergebenen Zahl `n` *rekursiv* berechnet. Die Funktion soll also ausnutzen, dass $n! = n \cdot (n-1)!$ gilt. Dabei muss keine tatsächlich existierende Programmiersprache verwendet werden; eine Art von Pseudocode, der den Programmfluss eindeutig darstellt, ist vollkommen ausreichend.

  Hinweis: Achte darauf, dass die Funktion auch mit den Parametern `n=0` und `n=1` zurechtkommt!

- **9.3.2** (2 Punkte) Schreibe eine Funktion `loop_factorial(n)`, welche die Fakultät von `n` mit Hilfe einer Schleife (also *nicht* rekursiv!) berechnet. Begründe, welche Art Schleife du verwendest!