64-041 Übung Rechnerstrukturen und Betriebssysteme

Aufgabenblatt 5 Ausgabe: 10.11., Abgabe: 17.11. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 5.1 (Punkte 5+5+5)

Gleitkommazahlen: Wandeln Sie folgende Dezimalzahlen in Gleitkommazahlen einfacher Genauigkeit gemäß IEEE 754 um. Es genügt dabei, wenn Sie die acht höchstwertigen Bit der Mantisse angeben:

(a) 3

(b) $-4,75$

(c) $10,125$

Aufgabe 5.2 (Punkte 5+7+8)

Größenvergleich von Gleitkommazahlen: Für den Vergleich von Gleitkommazahlen bietet Java alle sechs Vergleichsoperatoren:

```
a == b    a != b    a > b    a >= b    a < b    a <= b
```

Aufgrund der unvermeidlichen Rundungsfehler bei Gleitkommarechnung ist jedoch Vorsicht bei Verwendung dieser Operatoren geboten. Zum Beispiel liefert

```
double a = 0.1;
double b = 0.3;
System.out.println( (3*a) == b );
```

den Wert `false`.

(a) Ein naheliegender Ansatz ist daher, zwei Zahlen als „gleich“ anzusehen, wenn der Absolutwert ihrer Differenz kleiner als eine (vom Benutzer) vorgegebene Konstante ist:

```
final double eps = 1.0E-12;

if (Math.abs( a - b ) <= eps) { // Zahlen fast gleich
  ...
```

Welchen offensichtlichen Nachteil hat dieses Verfahren?

(b) Überlegen Sie sich ein Verfahren zum Vergleich von zwei Gleitkommazahlen, das nicht die absolute (wie oben), sondern eine relative Abweichung berücksichtigt.

(c) Hat auch dieses Verfahren Nachteile? ... und wenn ja, wie könnte man sie beheben?

Aufgabe 5.3 (Punkte 10+5)

UTF-8 Zeichen: Entschlüsseln Sie mit Hilfe der Vorlesungsunterlagen den folgenden hexadezimal codierten Text.

```
44 69 65 73 65 72 20 54 65 78 74 20 69 73 74 20 55 54 46 2D 38 20 63 6F
64 69 65 72 74 21 0D 0A 44 6F 72 74 20 67 69 62 74 20 65 73 0D 0A 09 55
6D 6C 61 75 74 65 20 28 C3 A4 29 0D 0A 09 5A 65 69 63 68 65 6E 20 28 C2
AE 29 0D 0A 09 53 79 6D 62 6F 6C 65 20 28 F0 9D 84 9E 29 0D 0A
```

(a) Wie sieht der Text aus? Notieren Sie dazu die Textdarstellung (mit Steuerzeichen).

(b) Was verrät Ihnen der Text über den Rechner mit dem er erstellt worden ist?

Aufgabe 5.4 (Punkte 5+5+5)

Shift-Operationen statt Multiplikation: Ersetzen Sie die folgenden Berechnungen *möglichst effizient* durch eine Folge von Operationen: <<, +, -. Nehmen Sie für die Variablen x und y den Datentyp `int` (32-bit Zweierkomplementzahl) an.

(a) $y = 20 \cdot x$

(b) $y = -56 \cdot x$

(c) $y = 124 \cdot (x + 2)$

Aufgabe 5.5 (Punkte 5+5+10+15)

Logische- und Shift-Operationen: Realisieren Sie, die folgenden Funktionen als *straightline*-Code in Java, das heißt ohne Schleifen oder If-Else Abfragen, bzw. ternärer Operator `.. ? .. : ...` Außerdem dürfen nur einige der logischen und arithmetischen Operatoren benutzt werden:

```
! ~ & ^ | + << >> >>>
```

Alle Eingabeparameter und Rückgabewerte sind jeweils (32-bit) Integerwerte.

(a) `bitNand(x,y)` Diese Funktion soll das bitweise NAND liefern: $\overline{x_i \wedge y_i}$. Als Operatoren dürfen nur | und ~ (OR, Negation) benutzt werden.

(b) `bitXnor(x,y)` Diese Funktion soll die XNOR-Verknüpfung (Äquivalenz) realisieren: $x_i \equiv y_i$. Als Operatoren dürfen nur | und ~ (OR, Negation) benutzt werden.

(c) `rotateLeft(x,n)` Die Funktion soll den in Java nicht vorhandenen Rotate-Left Operator für `x` nachbilden. Für das zweite Argument `n` gilt: $0 \leq n \leq 31$.

(d) `abs(x)` Der Absolutwert (Betrag) von `x`. Welchen Wert liefert ihre Funktion für den Eingabewert -2^{31}? Beschreiben Sie, wie Ihre Lösung funktioniert.