RS-Übung 4

Arne Beer (MN 6489196),
Rafael Epplee (MN 6269560),
Julian Polatynski (MN 6424884)

November 28, 2012

4.1

a)

Obwohl es diese Methode erlaubt, periodische Binärzahlen nach einer Multiplikation miteinander zu vergleichen, ist sie ungeeignet für den Vergleich von Zahlen mit geringer Differenz (z.B. $x \in \mathbb{R} < 1,0E-12$). So könnten bei dieser Methode zwei gänzlich unterschiedliche, wenn auch kleine, Werte als der gleiche angesehen werden.

b)

Eine andere Möglichkeit zum Vergleich wäre die der Berechnung einer relativen Differenz x. Bei dieser Methode ist es auch möglich große normalisierte Zahlen, bei denen wegen Speicherbegrenzung Rundungsfehler auftreten, zu vergleichen. Die Differenz ist wie folgt definiert:

$$\frac{a}{b} = x$$

Man legt für x einen Toleranzbereich fest, für den a=b gilt. z.B.

$$0,9999999999 <= x < 1 + 1,0000000001$$

c)

Der Nachteil ist hier, dass der Wert nur allgemein verglichen wird. Bei großen Zahlen z.B. $1 \cdot 10^{40}$ und $1 \cdot 10^{40} - 1$ würden diese als gleich angesehen werden, obwohl sie offensichtlich unterschiedlich sind.

4.2

a)

Die CR LF SP Lösung CR LF SP SP der CR LF SP SP SP Übungsaufgabe CR LF SP SP SP SP liegt CR LF SP SP SP SP SP vor CR LF SP SP SP SP SP Ihnen!

```
Die
 Lösung
  der
   Übungsaufgabe
    liegt
     vor
      Ihnen!
```

b)

Da für Zeilenumbrüche die Kombination aus Carriage-Return (CR) und Line-Feed(LF) verwendet wurde, ist offensichtlich, dass der Text auf einem der folgenden Systeme erstellt wurde: Windows, Dos, OS/2, CP/M, Tos (Atari)

4.3

a)

Da sowohl normale Buchstaben als auch Umlaute mit einer ISO-8859-1 Tabelle dargestellt werden können, bei der mit 8-Bit kodiert wird, werden insgesamt $800000 \cdot 8 \text{ bit } = 800000$ Bytes verwendet.

In Unicode wird mit 16 bit codiert, folglich ergibt sich:

$$800000 \cdot 16 \text{ bit } = 1600000 \text{ Byte}$$

In UTF-8 wird der Lateinische Zeichensatz mit 8 bit und die deutschen Umlaute mit 16 bit codiert. Der Anteil der Umlaute im Text beträgt:

$$(0,59\% + 0,29\% + 0,62\% + 0,31\%) \cdot 800000 = 14240$$

Folglich ist die Größe der Textdatei:

$$14240 \cdot 16 \text{ bit } + 785760 \cdot 8 \text{ bit } = 6513920 \text{ bit } = 814240 \text{ Byte}$$

b)

$$\begin{aligned} n &= (4DBF16 - 340016 + 1) + (9FCF16 - 4E0016 + 1) \\ &= (19903 - 13312 + 1) + (40911 - 19968 + 1) \\ &= 6592 + 20944 \\ &= 27536 \end{aligned}$$

c)

Bei direkter Kodierung mit UTF-16 werden 16 Bit pro Zeichen verwendet, also werden wie im deutschsprachigen Text 1,6 MB benötigt. In UTF-8 werden 24-Bit je Zeichen benötigt, daher werden 2,4 MB verwendet.

4.4

a) $y = 10 \cdot x \Leftrightarrow y = x \ll 3 + x \ll 1$
b) $y = 30 \cdot x \Leftrightarrow y = x \ll 5 - x \ll 1$
c) $y = -48 \cdot x \Leftrightarrow y = x \ll 4 - x \ll 7$
d) $y = 60 \cdot (x + 6) \Leftrightarrow y = 360 + x \ll 6 - x \ll 2$

4.5

a)

```
int bitNor(int x, int y) {
 return (~ x)&(~ y);
}
```

b)

```
int bitXor(int x, int y){
 return ~ (x & y)& ~ ((~ x)&(~ y))
}
```

c)

Bei der Ausführung von Rotate Right mit n Verschiebungen werden die Bits im Bereich von 2^0 bis 2^{n-1} nach rechts geschoben und an den Stellen 2^{32} bis 2^{32-n+1} eingesetzt.
Man kann die Operation in Java darstellen, indem man zuerst die zu rotierende

Zahl X um n Stellen logisch nach rechts verschiebt, und zu dieser Zahl durch die logische Operation "OR" die um $(32+ \sim n + 1)$ logisch nach links veschobene Zahl X addiert.

int bitXor(int x, int y){
 return $(x \ggg n) \mid (x \ll (32+ \sim n + 1)$
}

d)

Zunächst shiftet man die ursprüngliche Zahl x arithmetisch um 31 Stellen nach rechts, was dazu führt, dass bei einer negativen Zahl eine Einser- und bei einer positiven Zahl eine Nullermaske entsteht. Wenn man nun XOR auf das ursprüngliche x und die Maske anwendet, entsteht bei einer negativen Zahl ihre Negation und bei einer positiven Zahl die Zahl selber.

Nun muss man noch im Falle einer negativen Zahl 1 addieren. Allerdings nur, wenn die ursprüngliche Zahl auch negativ ist. Darum addiert man das Ergebnis eines logischen shifts nach rechts um 31 Bits. Bei einer negativen Zahl ist dieses Ergebnis eine simple 1, bei einer positiven Zahl eine 0.

$abs(x) = x \hat{} (x \gg 31) + (x \ggg 31)$