Prof. Andreas Goerdt
M. Pippig, M. Riedel, U. Schönke

TU Chemnitz
Wintersemester 2003/2004

Algorithmen und Programmierung

4. Übung

1. Aufgabe:

   a) Wir betrachten die 2er-Komplementdarstellung auf n und m, $m > n$, Stellen. Wie gewinnt man aus der 2er-Komplementdarstellung bezüglich n Stellen die 2er-Komplementdarstellung bezüglich m Stellen?

   b) Wir betrachten die 2er-Komplementdarstellung auf 4 Stellen. Geben Sie die Darstellung von -8, -4, -1, 3 und 5 in diesem 2er-Komplement an. Addieren Sie $3+(-8)$, $(-4)+(-1)$, $(-8)+(-1)$.

   c) Wie kann man an den Überträgen erkennen, ob die Addition von 2er-Komplementzahlen auf n Stellen aus dem Bereich zwischen -2^{n-1} und $2^{n-1}-1$ hinausführt?

   d) Notieren Sie die Reste der Division einer negativen Zahl durch 2, etwa -7, wie bei der Entstehung ihrer Binärdarstellung. Sehen Sie einen Zusammenhang zur 2er-Komplementdarstellung? Erläutern Sie diesen Zusammenhang am Beispiel der -7!

2. Aufgabe:
   Wir betrachten die Zahlendarstellungen bezüglich 2 und 2^a, $a \geq 2$. Wie kann man die binäre Darstellung $(z_{n-1}...z_0)_2$ in $(x_{m-1}...x_0)_{2^a}$ leichter als mit dem üblichen Verfahren transformieren?

3. Aufgabe:

   a) Probieren Sie die Java-Operation / für ganzzahlige Division und % für ganzzahligen Rest auf positiven und negativen Zahlen aus. Vergleichen Sie das Ergebnis mit unserer Definition

   $$a = DIV(a,b) \cdot b + MOD(a,b)$$

   wobei $b \neq 0$ und $0 \leq MOD(a,b) < b$ festgelegt ist.

   b) Versuchen Sie herauszufinden, ob es in Java eine Möglichkeit gibt zu erkennen, ob bei der Addition ein Überlauf aufgetreten ist.

4. Aufgabe:
   Beweisen Sie die wichtige Formel der geometrischen Reihe:

   $$\sum_{i=0}^{n} a^i = \frac{1-a^{n+1}}{1-a}$$

5. Aufgabe:
   Folgendes Programm zerlegt eine eingegebene Zahl in ihre Primfaktoren:

```
import Prog1Tools.IOTools;

public class Faktor {

  public static void main(String[] args) {

    long a ,d;
    a =  IOTools.readLong("Einlesen␣eines␣langen␣a␣>=␣2:␣");
    System.out.println(a +"␣soll␣zerlegt␣werden.");
    System.out.println();
    d = 2;

    while (d * d <= a) {
      while (a % d == 0){
        System.out.println( d + "␣ist␣Primfaktor");
        a = a/ d;
      }
      d++;
    }
    System.out.println();
    if (a>1) {
      System.out.println(a + "␣ist␣Primfaktor");
    }
  }
}
```

   Beschleunigen Sie den Faktorisierungsalgorithmus, indem Sie die Folge der Versuchsteiler d ausdünnen. Verhindern Sie dazu zunächst die Erzeugung aller Vielfachen von 2 oder 3. Überlegen Sie sich danach eine Methode, wie die noch vorhandenen Vielfachen von 5 leicht eliminiert werden können.