

# Algorithmen und Programmierung I

WS 2004 / 2005

## Übung 9 (25 Punkte)

Ausgabe: Mo. 3.1.2005

Abgabe: Mi. 12.1.2005, 10.00

### Verspätet abgegebene Lösungen werden nicht akzeptiert.

Die Abgabe erfolgt in Zweiergruppen. Die Gruppe darf nur mit ausdrücklicher Genehmigung des Tutors gewechselt werden.

Abgegeben werden müssen

- alle Lösungen einmal ausgedruckt. **Handschriftliche Lösungen werden nicht akzeptiert.**
- Programme, Testdaten, Testergebnisse in elektronischer Form in Absprache mit den Tutorinnen und Tutoren.

### Aufgabe 9.1 (10 Punkte)

Der folgende verschlüsselte Text wurde mit dem RSA-Verfahren und dem öffentlichen Schlüssel (11, 9788111) verschlüsselt:

```
[7321434, 3598567, 1716197, 4119550, 1716197, 1348875, 8435094, 8273804, 543720, 4505902, 4941202, 6726051, 8083299, 4896702, 3790340, 8176577, 3956251, 3855025, 9779898, 9626936, 7581573, 530038, 5596561, 1789054, 3956251, 3855025, 9779898, 1631514, 8435094, 9779898, 60307, 7379435]
```

Dabei wurden die Zeichen des Textes als Blöcke von je 2 Zeichen verschlüsselt. Bei einer ungeraden Anzahl von Zeichen wird ein Leerzeichen angefügt. Es wird nach der Entschlüsselung mit ausgegeben. Gesucht ist der ursprüngliche Text. Sie müssen dazu zunächst ein Programm schreiben, das den privaten Schlüssel ermittelt (Das geht nur, weil die verwendeten Zahlen viel zu klein sind. Sollten Sie einen Algorithmus finden, der auch für sehr große Zahlen mit einigen hundert Stellen in erträglicher Zeit schafft, erhalten Sie den Nobelpreis, mindestens aber den Turing Award – die höchste Auszeichnung, die in der Informatik vergeben wird ;).

Es wurde hier das Alphabet `chr 32` bis `char 127` zugrunde gelegt, also insgesamt 96 Zeichen (D.h. keine Umlaute etc.). Nach der Entschlüsselung müssen die Zahlen (zur Basis 96 ;) wieder in Blöcke mit zwei Zeichen zerlegt und in eine Zeichenkette umgewandelt werden.

Entwerfen Sie das Gesamtprogramm `rsaDecString` zur Dechiffrierungsprogramm von nach dem RSA-Verfahren verschlüsselten Texten. Gehen Sie nach dem Top-Down Verfahren vor und implementieren Sie die einzelnen Funktionen.

Einige nützliche Programmfragmente finden Sie auf der Alp1-Webseite.

### Aufgabe 9.2 (5 Punkte)

a) Ergänze die folgenden Definitionen:

```
instance (Ord x, Ord y)      => Ord (x,y) where...
instance (Ord a)            => Ord [a]  where ...
    -- lexikographische Ordnung, also wie im Telefonbuch!
```

b) Die folgenden Paare von Typen sollen unifiziert werden. Geben Sie den allgemeinsten Unifikator an, wenn das möglich ist, andernfalls begründen, warum es keine Unifikation gibt.

- (1) `(Int, a, a)` - `(a, a, [Bool])`
- (2) `(Int -> b)` - `(a -> Bool)`

(3)  $(a, [a]) - (b, c)$

Zusatzfrage zu (3) : können die Typen zu  $(\text{Bool}, [\text{Bool}])$  unifiziert werden? Zu  $([\text{Int}], [\text{Bool}])$  ? Zu  $([\text{Int}], [[\text{Int}]])$  ? Wenn ja / nein - warum (nicht) ?

### Aufgabe 9.3 (5 Punkte)

Was versteht man unter Top-Down, was unter Bottom-Up Entwurf?

Als Lösung wird mehr als ein Satz erwartet, vielmehr ein kurzer Text von ca. ½ Seite. Sie dürfen alle Ihnen zur Verfügung stehenden Quellen benutzen, müssen die aber zitieren.

*Diese Aufgabe dient nicht nur der fachlichen Aufarbeitung der Begriffe sondern dient auch der Entwicklung der für Informatiker unerlässlichen Fähigkeit, fachliche Sachverhalte auszudrücken.*

### Aufgabe 9.4 (5 Punkte)

Zeigen Sie, dass gilt:

$$\text{flip} \ . \ \text{flip} = \text{id}$$

wobei  $\text{flip} :: (a \rightarrow b \rightarrow c) \rightarrow (b \rightarrow a \rightarrow c)$

$$\text{flip} \ f \ x \ y = f \ y \ x$$

und  $\text{id}$  die Identität ist.