Praktische Informatik 1 WS 97/98

Prof. Bernd Krieg-Brückner
Heino Gärtner
Shi Hui
Kolyang
Christoph Lüth

6. Übungsblatt

Ausgabe: 29. Januar 1998
Bearbeitungszeit: zwei Wochen

Tip: Die Verwendung Funktionen höherer Ordnung erleichtert die Lösung der Aufgaben dieses Übungsblattes erheblich.

1 *Wortzählung* *Wichtung: 20%*

Implementieren Sie eine Funktion

```
countWords :: String-> [(String, Int)]
```

die die in einem Text enthaltenen Wörter zählt, und in Folge absteigender Häufigkeit sortiert ausgibt. Ein Wort ist hierbei ein durch Satz- oder Leerzeichen getrennte Folge von alphanumerischen Zeichen. Unterschiedliche Groß- und Kleinschreibung soll ignoriert werden, wie in:

```
countWords "Hi ho, hi ho, it's off to work we go." =
     [("ho",2), ("hi",2), ("go",1), ("we",1), ("work",1), ("to",1),
      ("off",1), ("s",1), ("it",1)]
```

Tip: Die vordefinierte Funktion `words:: String-> [String]` mag hierbei nützlich sein.

2 *Im Dienste Ihrer Majestät* *Wichtung: 40%*

Sie sind R (der Stellvertreter von Q) und müssen den Geheimagenten im Dienste Ihrer Majestät hilfreich zur Seite stehen. Heute geht es dabei um die Kodierung und Dekodierung von Geheimnachrichten.

Eines der einfachsten Verfahren zur Verschlüsselung von Nachrichten ist die *Cäsar-Rotation*, in dem jeder Buchstabe durch seinen n-ten Nachfolger ersetzt wird (wobei am Ende des Alphabets wieder von vorne angefangen wird). Implementieren sie eine Funktion

```
cesar :: Int-> Char-> Char
```

die für einen gegebene Verschiebung diese Ersetzung beschreibt. Mit Hilfe dieser Funktion implementieren Sie zwei Funktionen

```
rot_encode :: Int-> String-> String
rot_decode :: Int-> String-> String
```

die einen Text verschlüsselt und wieder entschlüsselt.

Eine Verallgemeinerung dieses Verfahrens ist es, nach einem festen Schlüssel jeden Buchstaben durch einen anderen zu ersetzen (z.B. "a" mit "x", "b" mit "r", "c" mit "u", "d" mit "a", etc.) Ein Schlüssel ist also eine Folge von 26 verschiedenen Buchstaben (eine Permutation des Alphabets), die diese Ersetzung beschreibt, also z.B. würde der Schlüssel für die oben beschriebene Ersetzung mit "xrud" anfangen.

Implementieren Sie eine Funktion

```
transpose :: [(Char, Char)] -> Char-> Char
```

die ausgehend von einer Liste von Paaren von Buchstaben die Ersetzung des jeweils ersten durch den zweiten Buchstaben beschreibt. Dabei kann davon ausgegangen werden, daß die ersten Buchstaben paarweise disjunkt sind, und daß jeder Buchstabe des Alphabets genau einmal vorkommt. Darauf aufbauend, implementieren Sie zwei Funktionen

```
encode :: String-> String-> String
decode :: String-> String-> String
```

deren erstes Argument ein Schlüssel ist, und die einen Text verschlüsselt, und wieder entschlüsselt.

$\boxed{3}$ *The Gold Bug*[1] *Wichtung: 40%*

Leider ist dieses Verfahren nicht sehr sicher, da kodierte Nachrichten recht einfach entschlüsselt werden können. Die Entschlüsselung beruht auf der Beobachtung, daß in einem durchschnittlichen Text nicht alle Buchstaben gleich oft vorkommen; so kommt z.B. in einem deutschen Text der Buchstabe “e” am häufigsten vor, und deshalb liegt es nahe, daß der am häufigsten im verschlüsselten Text auftretende Buchstabe die Verschlüsselung des “e” ist.

Implementieren Sie zuerst eine Funktion

```
char_frequency :: String-> String
```

die die Buchstaben des Alphabets in Reihenfolge absteigender Häufigkeit im Argumenttext sortiert zurückgibt. `char_frequency` angewandt auf einen repräsentativen Beispieltext, und auf den Verschlüsselungstext gibt jetzt zwei Zeichenketten, die genau die Entschlüsselung beschreiben. Implementieren Sie so eine Funktion

```
crack :: String-> String
```

die einen verschlüsselten Text entschlüsselt.

[1] Titel einer Kurzgeschichte (1843) des Computerpioniers Edgar Allen Poe, in der nach dem diesem Verfahren die verschlüsselte Lagebeschreibung eines verborgenen Schatzes entschlüsselt wird.