Übung zu Funktionen

**Tannenbaum**

Schreibe eine Funktion, die einen Tannenbaum mit ASCII-Art „gezeichnet“ als Text zurückliefert. Eingabe ist die Höhe des Tannenbaums. Als Beispiel ein Tannenbaum der Höhe 5:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | X        XXX       XXXXX      XXXXXXX     XXXXXXXXX         I |

**Variation #1**

Auf Wunsch soll der Tannenbaum auch einen Stern als Spitze haben können:

Tannenbaum.ZeichnenMitSpitze(5) oder Tannenbaum.Zeichnen(5, true)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | \*         X        XXX       XXXXX      XXXXXXX     XXXXXXXXX         I |

**Fröhliche Zahlen**

Entwickle eine Funktion, die erkennt, ob eine Zahl „fröhlich“ ist oder nicht.

Eine [Fröhliche Zahl](http://de.wikipedia.org/wiki/Fr%C3%B6hliche_Zahl) ist eine Zahl, bei der die Summe der Quadrate ihrer Ziffern „auf die Dauer“ 1 ergibt. Beispiel:

19 -> 1^2 + 9^2 = 82 -> 8^2 + 2^2 = 68 -> 6^2 + 8^2 = 100 -> 1^2 + 0^2 + 0^2 = 1

# ROT-13

Schreibe eine Funktion, die einen Text mit dem Verfahren [ROT-13](http://www.code-knacker.de/verschluesselung.htm) verschlüsselt.

Bei ROT-13 wird jeder Buchstabe des zu verschlüsselnden Textes durch den ersetzt, der 13 Stellen weiter hinten im Alphabet liegt. Sollte dabei über das Alphabetende hinausgegriffen werden, geht es am Anfang weiter.

Beispiel: Hello, World -> URYYB, JBEYQ

Aus „e“ wird „R“ und aus „W“ wird „J“. Kleinbuchstaben werden in Großbuchstaben verwandelt.

Umlaute müssen vor Verschlüsselung durch Buchstabenkombinationen ersetzt werden: „Ö“ = „OE“, „Ä“ = „AE“, „Ü“ = „UE“, „ß“ = „SS“.

Zeichen die keine Buchstaben sind, werden nicht verschlüsselt.

### Variation #1

Mache den Versatz (hier: 13) variabel.

### Variation #2

Verschlüssele auch Ziffern. Nimm dafür an, dass Ziffern und Großbuchstaben eine Einheit bilden: 0123456789ABCDEFGH…XYZ.

Aus „0“ wird dann mit Versatz 13 „D“ und aus „Z“ wird dann nicht mehr „M“, sondern „C“.

# Textumbruch

Implementiere eine Funktion zur Umformatierung eines Textes. Funktionssignatur:

|  |  |  |
| --- | --- | --- |
|  |  | string Umbrechen(string text, int maxZeilenlänge) |

Ein potenziell mehrzeiliger Text soll an Wortgrenzen so umgebrochen werden, dass eine maximale Zeilenlänge nicht überschritten wird, z.B.

Ausgangstext (aus „Advent“ von [Loriot](http://www.personal.uni-jena.de/~dps/advent.html)):

|  |  |
| --- | --- |
| 1  2  3 | Es blaut die Nacht,  die Sternlein blinken,  Schneeflöcklein leis hernieder sinken. |

Mit einer max. Zeilenlänge von 9 würde die Funktion diesen Text produzieren:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | Es blaut  die  Nacht,  die  Sternlein  blinken,  Schneeflö  cklein  leis  hernieder  sinken. |

Und mit einer max. Zeilenlänge von 14 diesen:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Es blaut die  Nacht, die  Sternlein  blinken,  Schneeflöcklei  n leis  hernieder  sinken. |

Worte des Ausgangstextes werden so lange mit einem Leerzeichen zwischen ihnen zu Zeilen zusammengesetzt, bis die max. Zeilenlänge erreicht ist. Passt ein Wort nicht mehr in eine Zeile, fließt es in die nächste. Worte, die länger als die max. Zeilenlänge sind, werden einfach abgeschnitten; der Rest fließt dann in die nächste Zeile (s.o. z.B. „Schneeflöcklein“).

Auf Wunsch kann der umgebrochene Text auch im Blocksatz geliefert werden. Leerzeichen sollten dann möglichst gleichmäßig in den Zeilen verteilt sein, z.B. mit max. Zeilenlänge 22:

|  |  |
| --- | --- |
| 1  2  3  4 | Es  blaubt die  Nacht,  die Sternlein blinken,  Schneeflöcklein   leis  hernieder      sinken. |