Aufgabe 7: (Der Algorithmus von Euklid)

Der Euklidische Algorithmus berechnet den größten gemeinsamen Teiler von zwei natürlichen Zahlen. Der Algorithmus wurde bereits ca. 300 v.Chr. von Euklid beschrieben. Wir werden im Folgenden eine rekursive Variante des Algorithmus angeben.
Seien $a \in \mathbb{N}_0$, $b \in \mathbb{N}$. Dann gilt $\mathrm{ggT}(a, b) = \mathrm{ggT}(b, a \bmod b)$ und $\mathrm{ggT}(a,0)=a$.

Schreiben Sie ein rekursives Programm in einer beliebigen Programmiersprache auf, welches $\mathrm{ggT}(a, b)$ berechnet.

Aufgabe 8: (rekursive Decent Parser)

Schreiben Sie einen Algorithmus, der rekursiv Worte der Form $a^n b^n$ erkennt. Erklären Sie den Begriff Linksrekursion. Schreiben Sie einen Algorithmus, der arithmetische Ausdrücke auswertet. Dabei soll ‚Punkt vor Strich' zunächst außer Acht gelassen werden.

Aufgabe 0: Wiederholung Halteproblem

Aufgabe 1: (Rekursion) Die Türme von Hanoi

Die Legende, welche die Türme von Hanoi begleitet, besagt, dass unter der Herrschaft von Kaiser Fo Hi in Benares ein Tempel stand – mit einem Dom, der die Mitte der Welt markierte. Als die Welt erschaffen wurde, stellte Gott dort 3 diamantene Stäbe auf, die eine Elle lang und so dick wie 3 Körper einer Biene waren. Danach legte Gott 64 goldene Scheiben auf den ersten der Stäbe. Die Scheiben wurden immer kleiner, so dass sie zusammen eine Art Kegel bildeten.
Er trug den Mönchen auf, die goldenen Scheiben auf den dritten Stab zu transportieren. Der zweite Stab darf als "Zwischenlager" benutzt werden. Es darf aber niemals eine größere auf einer kleineren Scheibe zu liegen kommen und immer nur eine Scheibe auf einmal bewegt werden. In diesem Dom bewegen die Mönche seither goldene Scheiben zwischen diamantenen Stäben. Man sagt, dass das Ende der Welt kommt, wenn die Mönche mit ihrer Aufgabe fertig sind.

a) Geben Sie eine Lösung für $n = 3$ Scheiben an. Wie viele Züge benötigen Sie für die Lösung mindestens?

b) Überlegen Sie sich eine Formel, mit der Sie für jede beliebige Anzahl n von Scheiben bestimmen können, wie viele Züge mindestens notwendig sind, um den Stapel den Regeln entsprechend zu versetzen.

c) Überlegen Sie sich einen rekursiven Algorithmus, der das Problem löst. Es genügt dabei, wenn ihr Programm bzw. Ihr Pseudocode für jeden Schritt den Ausgangs- und den Zielstapel der zu bewegenden Platte ausgibt.

Aufgabe 2: (Rekursion) Summe für *expN(x,n)*

Gegeben sei die Exponentialreihe, die auf der Menge *IR* wie folgt definiert ist:

$$exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} \qquad \text{mit } x \in \mathbb{R}$$

Implementieren Sie eine rekursive function *expN(x,n)*, die die Exponentialreihe für die Eingabe *x* bis zum *n*-ten Glied berechnet.

Aufgabe 3 (Quersumme rekursiv)

Implementieren Sie eine rekursive function *Quer*, die die Quersumme einer natürlichen Zahl berechnet.

Aufgabe 4 (Ackermann-Funktion)

Die Funktion $A(m, n)$ ist durch folgende Rekursionen für natürliche Zahlen m und n definiert:

(i) $A(0, n) = n + 1$

(ii) $A(m, 0) = A(m-1, 1)$, *für* $m > 0$

(iii) $A(m, n) = A(m-1, A(m, n-1))$, *für* $m, n > 0$

Die Funktion A heißt Ackermann-Funktion. Implementieren Sie eine rekursive function A, die $A(m, n)$ berechnet.

Aufgabe 6 (Dezimal-/Dualkonvertierung)

Programmieren Sie eine **rekursive** Funktion, die als Parameter eine positive, ganze Zahl annimmt und einen String mit der Binärdarstellung dieser Zahl zurückliefert.

Beispiel: Parameter: 14 Ergebnis: “1110“
Parameter: 22 Ergebnis: “10110“

Aufgabe 5 (Buchstabenersetzung)

Es soll ein Wortspiel realisiert werden. Gegeben sind zwei Wörter gleicher Länge (frei wählbar durch den Nutzer). Gesucht werden weitere Wörter, die durch Austauschen eines Buchstabens das eine Wort in das andere überführen.

Beispiel: BEIN → BERN → BERG → WERG

Es soll ein neues Wort durch Austauschen des i-ten Buchstaben zusammengebaut werden. Beispiel:

Wort1=BEIN, Wort2=WERG 1 Bst. ersetzt: Zwischenergebnis = WEIN
Wort1=BEIN, Wort2=WERG 2 Bst. ersetzt: Zwischenergebnis = BEIN
Wort1=BEIN, Wort2=WERG 3 Bst. ersetzt: Zwischenergebnis = BERN
Wort1=BEIN, Wort2=WERG 4 Bst. ersetzt: Zwischenergebnis = BEIG (kein Wort)

Bereits implementiert sei eine Funktion IsWort(s:String) return boolean, die entscheidet, ob ein String s ein Wort ist oder nicht. Sie können das Prinzip der Rekursion verwenden.

Zusatzaufgabe: Es sollen auch Wörter gesucht werden, die einen „Umweg“ bedeuten, d.h. Buchstaben verwandt werden, die in keinem der beiden Wörter vorhanden sind.
Beispiel: BEIN → BERN → BERT → BELT (R ist nicht in beiden enthalten!)