Übung 5
Funktionen & Listen

A 5.1: Liste von "Adressen" (6)

(a) Definieren Sie eine Liste von mindestens 3 Elementen vom Typ `Adresse` (siehe Aufg. 4.1). Schreiben Sie eine Prozedur zur alphabetischen Sortierung dieser Liste nach den `Name-Ein`trägen. Sortierung sollte mittels Listen-Pointer realisiert werden

(b) Geben Sie dem Benutzer die Möglichkeit, eine neue Adresse einzugeben, und ordnen Sie diese in die Liste alphabetisch ein.

A 5.2: Funktion `Binomialkoeffizient` (6)

Schreiben Sie ein C++ - Programm mit einer Funktion

`int Binomialkoeffizient(int n, int k)`, die den Binomialkoeffizienten

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} \qquad n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

zweier Zahlen n und k berechnet und das Ergebnis zurückgibt. Geben Sie für n = 1 bis 6 und k = 1 bis n die Binomialkoeffizienten in der folgenden Form am Bildschirm aus:

```
1
2    1
3    3    1
4    6    4    1
5   10   10    5    1
6   15   20   15    6    1
```

A 5.3: Rekursion und explizite Deklaration von Funktionen (6)

Die mathematische Funktion **ulam(n)** ist für na

$$\mathbf{ulam(n)} = \begin{cases} n = 1 & \mathbf{0} \\ n > 1 \wedge \text{n gerade} & \mathbf{ulam}\left(\frac{\mathbf{n}}{\mathbf{2}}\right) \\ n > 1 \wedge \text{n ungerade} & \mathbf{ulam(3n+1)} \end{cases}$$

türliche Zahlen n > 0 wie folgt definiert:

Schreiben Sie ein C++ - Programm, in dem Sie zwei Funktionen `boolean ulam_even(int n)` und `boolean ulam_odd(int n)` definieren, die sich gegenseitig rekursiv aufrufen.
Beispiel: `ulam_odd(5)→ ulam_even(16)→ ulam_even(8)→ ulam_even(4) → ulam_even(2)→ ulam_odd(1)→ TRUE`
Erweitern Sie das Programm um die Möglichkeit, eine Zahl n einzugeben und die Anzahl der Funktionsaufrufe `ulam_even/odd` bis zum Abbruch der Rekursion auszugeben.
Beschreiben Sie zwei Möglichkeiten, diese Funktionsaufrufe zu zählen und implementieren Sie eine davon.
Bemerkung: Es soll die gesamte Zahl der Funktionsaufrufe gezählt werden, d.h. nur ein Zähler für `even` und `odd` !