Bemerkung: Die ersten beiden Aufgaben sind eher aus mathematischer Sicht interessant. Wenn Sie nur an der Einübung des Programmierens interessiert sind, sind die letzten drei Aufgaben besser geeignet.

Aufgabe 1. Diese Aufgabe wird auf eine `power(x, y)`-Funktion führen, die für beliebige $x \in \mathbb{R}^+$ und $y \in \mathbb{R}$ den Wert von x^y berechnet.

- Implementieren Sie die Exponential-Funktion `expo(x)`, die e^x mithilfe folgender Reihendarstellung berechnet:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

- Implementieren Sie eine Logarithmus-Funktion `logarithm(x)`, die $\ln(x)$ mithilfe folgender Reihendarstellung berechnet:

$$\ln(x) = 2 \cdot \sum_{k=0}^{\infty} \left(\frac{x-1}{x+1} \right)^{2k+1} \frac{1}{2k+1}$$

- Verwenden Sie die Formel

$$x^y = e^{y \cdot \ln(x)}$$

  um `power(x, y)` zu bestimmen. Vergleichen Sie für ganzzahliges `y` Ihre Ergebnisse mit den Ergebnissen Ihrer Implementierung für Aufgabe 4 von Zettel 3.

Aufgabe 2. Betrachten Sie die folgende Funktion, die, analog zur Vorlesung, zu einer gegebenen Zahl n die Fibonacci-Zahl F_n berechnet:

```
unsigned fibonacci(unsigned n)
{
   if (n < 2)
   {
      return 1;
   }
   else
   {
      return fibonacci(n-1) + fibonacci(n-2);
   }
}
```

Zeigen Sie, dass dieses Verfahren wenigstens Φ^{n-1} Aufrufe der Funktion `fibonacci` benötigt, um zu einer gegebenen Zahl n den Wert F_n zu berechen, wobei $\Phi := \frac{1+\sqrt{5}}{2} \approx 1,618$ sei.
Hinweis: Sie können benutzen, dass $\Phi^2 = \Phi + 1$ gilt.

Aufgabe 3. Die Folge $(a_n)_{n \in \mathbb{N}}$ sei wie folgt auf rekursive Art gegeben. Es gelte $a_0 = a_1 = a_2 = 1$, und für $n > 2$ sei $a_n = 2a_{n-1} + a_{n-2} + 2a_{n-3}$. Implementieren Sie eine Funktion, die zu gegebenem n den Wert a_n berechnet. Programmieren Sie zwei Versionen, einmal mit rekursiven Aufrufen und einmal ohne.

Aufgabe 4. Die Bell-Zahlen $(B_n)_{n\in\mathbb{N}}$ (die natürlichen Zahlen enthalten bei uns stets die 0) sind wie folgt rekursiv definiert: $B_0 = 1$ und

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

für $n \in \mathbb{N}$.

a) Implementieren Sie eine rekursive Funktion, die zu gegebenem n den Wert B_n ausrechnet.

b) Schreiben Sie eine Funktion, die ohne Rekursion zu einem gegebenen n die Zahl B_n, ausrechnet, indem Sie geeignete Teilergebnisse abspeichern. Vergleichen Sie die Laufzeit mir Ihrer rekursiven Implementierung.

Aufgabe 5. Schreiben Sie eine Funktion, die zu einer gegeben positiven ganzen Zahl n die Menge aller Teiler von n in einem Vektor speichert.