Aufgabe 1. Die folgende Funktion `array_sum` soll zu einem gegeben `int`-Array die Summe aller Einträge bestimmen. Dabei hat sich eine unbekannte Anzahl von Fehlern eingeschlichen. Man finde sie alle.

```
#include <iostream>

int array_sum(int          array,
              unsigned     length)
{
   int sum = 0;

   for (unsigned i = 1; i <= length; ++i)
   {
      sum += array[i];
   }
   return sum;
}

int main()
{
   int a[6] = { 3, 12, 1, 3, 7, 21 };
   std::cout << array_sum(a,6) << std::endl;
}
```

Aufgabe 2. 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:
$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$
- Implementieren Sie eine Logarithmus-Funktion `logarithm(x)`, die $\ln(x)$ mithilfe folgender Reihedarstellung 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 3. Die Folge $(f_n)_{n\in\mathbb{N}}$ der Fibonacci-Zahlen ist wie folgt rekursiv definiert: Es gelten $f_0 = 1$ und $f_1 = 1$, und für $n > 2$ gilt $f_n = f_{n-1} + f_{n-2}$. Betrachten Sie die folgende Funktion, die zu einer gegebenen Zahl n die Fibonacci-Zahl f_n zu 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 4. 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, indem Sie die Werte a_i mit $i \in \{0, \ldots, n-1\}$ abspeichern.

Aufgabe 5. Wir betrachten wieder die Bell-Zahlen $(B_n)_{n\in\mathbb{N}}$ aus Aufgabe 5 von Zettel 3, die definiert sind über die Gleichungen $B_0 = 1$ und $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$ für $n \in \mathbb{N}$. 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 6. Schreiben Sie eine Funktion, die zu einer gegeben positiven ganzen Zahl n die Menge aller Teiler von n in einem Vektor speichert.