Aufgabe 1. Das folgende Programm sollte die Fakultätsfunktion implementieren und $5! + 16!$ auf der Konsole ausgeben. Dabei haben wir 6 Fehler dabei gemacht. Schnappen Sie sich alle!

```
#include <iostream>

int fakultaet (n)
{
   int ergebnis = 0;     /* speichert die Fakultaet */

   while (n > 0)         /* verkleinere n, bis es */
      ergebnis *= n;     /* null ist und multi- */
      n--;               /* pliziere mit ergebnis */
   return ergebnis;
}

int main ()
{
   int add2fak;
   const int a = 5;
   const int b = 16;

   add2fak = fakultaet (a) + fakulatet (b);
   std::cout << a << "! + " << b "! = " << add2fak << std::endl;
   return 0,
}
```

Was passiert, wenn man nach der Korrektur `b` statt mit 16 mit 17 initialisiert?

Aufgabe 2. a) Implementieren Sie die Wurzelfunktion `wurzel(x)` (mit dem Heron-Verfahren, Algorithmus 2 vom ersten Zettel) als Funktion.

b) Lagern Sie diese Funktion in eine eigene Datei aus.

Aufgabe 3. Schreiben Sie eine Funktion, die zu drei gegebenen `int`-Werten das Maximum und das Minimum dieser Werte bestimmt. Die Funktion soll die beiden berechneten Werte in zwei zusätzlichen Variablen, die per Referenz übergeben werden, zurückgeben.

Aufgabe 4. a) Implementieren Sie für $x \in \mathbb{R}$ und $n \in \mathbb{N}$ eine Potenzfunktion $x^n =$ `power(x, n)` mit der Double-and-Add-Methode:

$$\text{power}(x, n) = \begin{cases} 1 & \text{wenn } n = 0 \\ x \cdot \text{power}\left(x^2, \frac{n-1}{2}\right) & \text{wenn } n \text{ ungerade} \\ \text{power}\left(x^2, \frac{n}{2}\right) & \text{wenn } n \text{ gerade} \end{cases}$$

zuerst mal rekursiv.

b) Implementieren Sie eine Potenzfunktion `naiv_power(x, n)`, indem Sie eine Schleife von 1 bis n laufen lassen und bei jedem Durchlauf eine mit 1 Initialisierte Variable mit x multiplizieren (ähnlich wie in der Vorlesung). Berechnen Sie $0,9999999999^{2000000000}$ einmal mit `power(x, n)` von oben und einmal mit `naiv_power(x, n)` (es sollte ca. $0,818731$ raus kommen).

c) Implementieren Sie die Double-and-Add-Methode mit einer Schleife, also ohne rekursiven Aufruf.

d) Welches dieser Verfahren ist am langsamsten? Warum?