Aufgabe 1. Lege ein Modul `mymath.c` / `mymath.h` an, in dem du die bisher geschriebenen Funktionen auslagerst.

Wir brauchen im folgenden eine Potenzfunktion, die zwei Fließkommazahlen als Argumente akzeptiert. Falls du diese Funktion gestern geschrieben hast, sollte sie jetzt im `mymath`-Modul verfügbar sein. Andernfalls gibt es die funktion

```
double pow(double x, double y);
```

in der Systemheader `<math.h>`. Im Skript findest du im Anhang eine vollständige Referenz einiger Systembibliotheken.

Aufgabe 2. Implementiere die Riemann'sche Zeta-Funktion für $s > 1$:

$$\zeta(s) := \sum_{k=1}^{\infty} \frac{1}{k^s}$$

Warnung: Diese Reihe konvergiert nicht für $s \geq 1$.

Aufgabe 3. Erweitere das `mymath`-Modul noch um eine Funktion, die zu drei beliebigen Koeffizienten $a, b, c \in \mathbb{R}$ einer quadratischen Gleichung

$$a \cdot x^2 + b \cdot x + c = 0$$

die Lösungen berechnet und die größere Lösung zurück gibt.

Aufgabe 4. a) Implementiere eine Funktion `presentation(n, r)`, die zu einer natürlichen Zahl (im Zehnersystem) n und einer Basis $0 < r < 10$ mit nachfolgendem Algorithmus die Darstellung von n in der Basis r ausgibt: Dividiere durch r, merke dir den Rest, den die Zahl bei der Division lässt. Verfahre mit dem Ergebniss der Division (abgerundet) so weiter. Die Reste geben die Zahl in der neuen Basis. Hier ein Beispiel, wie der Algorithmus für die 99 in der Basis 8 aussehen soll:

$$99 = 12 \cdot 8 + 3 \quad (1)$$
$$12 = 1 \cdot 8 + 4 \quad (2)$$
$$1 = 0 \cdot 8 + 1 \quad (3)$$

Die Darstellung von 99 zur Basis 8 ist also 341 (oder „richtig“ 143). Lagere die Funktion in ein eigenes Modul aus!

b) Was macht dieser Programmschnipsel? Warum?

```
1 for(; n != 0; n /= r) {
2         printf("%i\n",n%r);
3 }
```

c) * Implementiere `presentation2(n, r)` rekursiv s.d. die Ziffern in richtiger (umgekehrter) Reihenfolge ausgegeben werden. Implementiere diese Funktion im gleichen Modul wie `presentation(n, r)`!