Übung 4

a) Schreiben Sie ein Programm, welches das Maximum von zehn eingegebenen Zahlen berechnet. (Verwenden Sie im Folgenden *break* bzw. *continue*.) Dabei…

i. soll bei Eingabe einer negativen Zahl sofort abgebrochen werden;
ii. sollen negative Zahlen ignoriert werden.

b) Gegeben seien die folgenden Deklarationen:

```
int x = 1, y = 2;
bool z = true;
```

Zu was werten die folgenden Ausdrücke aus (jeweils unabhängig voneinander und nicht nacheinander)? Erst überlegen, dann ausprobieren!

- y++*5+y
- y*5%++y
- y++-y--
- x*5<y||z&&x>y
- x=y=y+1

c) Die Fakultät, $n!$, einer Zahl $n \in \mathbb{N}$ ist das Produkt aller Zahlen von 1 bis n:
$n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n$ (wobei 0! = 1).

Schreiben Sie ein C-Programm zur Berechnung der Fakultät für eine eingegebene Zahl *n*. Bis zu welchem Wert von *n* reicht `int` als Datentyp aus ohne Überlauf bzw. `unsigned int`? Bis zu welchem *n* reicht `long long` (je vorzeichenbehaftet und vorzeichenlos)?

d) Implementieren Sie ein Programm zur Berechnung der Kreiszahl π in zwei Varianten (verwenden Sie für alle Nicht-Ganzzahlen den Datentyp `double`), und zwar mit Hilfe…

i. der Leibniz-Reihe mit 1.000.000 Summanden: $\frac{\pi}{4} = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$
ii. des Wallis'schen Produktes mit 1.000.000 Faktoren: $\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \ldots$

e) Bei der Verwendung von Gleitkommazahlen kann es manchmal Probleme geben, was hier nachzuvollziehen ist. Geben Sie jeweils das Ergebnis der jeweiligen Addition aus[1], falls der gesamte Vergleich wahr ist, und versuchen Sie, sich die Ausgabe zu erklären:

- 0.1 + 0.2 == 0.3
- 0.1 + 0.3 == 0.4

Addieren Sie weiterhin jeweils die ersten *n* (10000, 100000 bzw. 1000000) Summanden der harmonischen Reihe ($\sum_{i=1}^{\infty} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots$) erst mit `float` und dann mit `double` Werten. Vergleichen Sie mit den jeweils korrekten Annäherungen unten. Was fällt auf?

9.7876060360443822
12.0901461298634279
14.3927267228657236

Mehr Informationen zum Thema Floating-Point-Ungenauigkeiten gibt es z.B. unter:

- http://www.stat.cmu.edu/~brian/711/week03/perils-of-floating-point.pdf
- https://stackoverflow.com/questions/588004/is-floating-point-math-broken

[1] Floating-Point-Zahlen kann man je nach Formatierungsbedarf über %f, %e oder %g ausgeben. Die Anzahl der gewünschten Nachkommastellen bei %f lässt sich, z.B. für *6* Nachkommastellen (Default), so angeben: %.*6*f