Computermathematik – Übungsblatt 7

- **Abgabeschluss:** Di 17. 12. um 23:59
- **Präsentation:** Mi 18. 12. in der Übungsgruppe
- **Abgabeformat:** `.sws` oder `.sagews`

Aufgabe 11 – Primzahl-Zwillinge (4 Punkte)

Wenn sowohl p als auch $p+2$ Primzahlen sind, heißen sie Primzahlzwillinge. Beispielsweise sind 5 und 7 Primzahlzwillinge, ebenso 41 und 43. Erstelle eine Liste aller Primzahlzwillinge unter 5000. Auf welche verschiedenen Arten könnte man so eine Liste berechnen? Welche davon sind vermutlich mehr, welche weniger effizient?

Hint: möglicherweise hilfreiche Funktionen sind `is_prime()`, `primes()`, `prime_range()`, `zip()`.

Aufgabe 12 – Einfach normale Zahlen (4 Punkte)

Eine reelle Zahl heißt **einfach normale Zahl** (zur Basis 10), wenn unter ihren Nachkommastellen jede dezimale Ziffer asymptotisch gleich häufig ist. Beispielsweise ist die periodische Zahl $0.\overline{1234567890}$ eine einfach normale Zahl.

Definiere eine Funktion `how_simply_normal(x, n)`, die experimentell untersucht, "wie einfach normal" eine Zahl x in den ersten n Stellen aussieht. Dazu soll sie eine Statistik der ersten n Nachkommastellen von x erstellen und zählen, wie oft welche Ziffer darin vorkommt. Die genauen Datentypen für x (Zahl? String?) und der Rückgabewert (Liste? Print-Ausgabe? Plot?) sind selbst wählbar.

Interessante Testzahlen sind beispielsweise:

- $\frac{1}{7}, \frac{17}{29}, \frac{1}{n}$, wobei n die persönliche Zahl aus Beispiel 9 ist,
- $\pi, e, \sqrt{2}, \ldots,$
- die Champernowne-Zahl $0.12345678910111213141516\ldots$ sowie die Copeland-Erdős-Zahl $0.2357111317192329\ldots$, deren Nachkommestellen man jeweils bekommt, indem man aufsteigend alle natürlichen Zahlen bzw. alle Primzahlen"aneinanderklebt".

Hint: mit `str()` und `int()` bzw. `R()` (mit `R = RealField(...)`) kann man Zahlen in Strings und wieder zurück übersetzen, um an die dezimalen Nachkommastellen zu kommen.

Bonus – Absolut normale Zahlen (2 Bonuspunkte)

Eine reelle Zahl heißt **absolut normale Zahl**, wenn nicht nur jede Ziffer asymptotisch gleich häufig vorkommt, sondern auch für jede beliebige Länge k jede Ziffernfolge dieser Länge; außerdem muss diese Eigenschaft nicht nur für die dezimalen Ziffern gelten, sondern auch für Nachkommastellen bezüglich jeder anderen Basis.

Definiere eine Funktion `how_absolutely_normal(x, n, k)`, die die Häufigkeit jeder Ziffernfolge der Länge k innerhalb der ersten n Ziffern der dezimalen Nachkommastellen von x berechnet.

Beispiel: für $x = 1/6 = 0.\overline{16}$ kommt unter den ersten $n = 6$ Ziffern der Ziffernstring 1616 zweimal vor (beginnend bei der ersten sowie bei der dritten Nachkommastelle), der Ziffernstring 6161 einmal und jeder andere String der Länge $k = 4$ überhaupt nicht.