Programming Exercises

1. Schreibe eine Funktion, die "Hello world!" ausgibt.
2. Schreibe eine Funktion, die ausgibt, ob eine gegebene Zahl positiv oder negativ ist.
3. Schreibe eine Funktion, die ausgibt, ob eine gegebene Zahl betragsmässig grösser als π ist oder nicht.
4. Schreibe eine Funktion, die zwei gegebene Zahlen nimmt und deren geometrisches beziehungsweise harmonisches Mittel retourniert.
5. Schreibe eine Funktion, die zwei gegebene Zeichenketten nimmt und ausgibt, welche die kürzere ist (Hilfe: `?"nchar"`.)
6. Schreibe eine Funktion, die eine gegebene Zeichenkette so oft zeilenweise ausgibt wie diese Zeichen hat (Hilfe: `?"for"`.)
7. Schreibe eine Funktion, die mit einer gegebenen natürlichen Zahl n beginnend in absteigender Reihenfolge zeilenweise alle natürlichen Zahlen $\leq n$ ausgibt.
8. Für zwei ganze Zahlen `x` und `y` liefern `x %/% y` beziehungsweise `x %% y` das Ergebnis und den Rest bei der ganzzahligen Division von `x` durch `y`. Schreibe eine Funktion, die ausgibt, ob `x` durch `y` (ohne Rest) teilbar ist oder nicht.
9. Schreibe eine Funktion `pow`, die die Potenz x^a retourniert, wobei a standardmäßig gleich eins ist.
10. Schreibe eine Funktion, die die Folge aller Großbuchstaben in einer gegebenen Datei abspeichert.
11. Die Funktion `logstar(x)` ist definiert als die minimale Anzahl n sodass n-maliges logarithmieren von x einen Wert kleiner als 1 liefert. (Also beispielsweise für $x = 123$ ist $\text{logstar}(123) = 3$, denn $\log(\log(123)) = 1.571151 \geq 1$ und $\log(\log(\log(123))) = 0.4518085 < 1$. Implementiere `logstar`.
12. Schreibe eine Funktion, die die gesamte Anzahl von Ziffern in einer gegebenen Folge natürlicher Zahlen berechnet.
13. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl n jene Folge erzeugt, in der aufsteigend jede natürliche Zahl $i \leq n$ i-mal wiederholt wird. (Hilfe: `?rep`; Objekte können vermittels `c()` zu Folgen kombiniert werden.)
14. Der grösste gemeinsame Teiler $\gcd(a, b)$ zweier natürlicher Zahlen a, b kann mit Hilfe des Euklid'schen Algorithmus berechnet werden. Dieser beruht auf den folgenden Beobachtungen:
    (a) Wenn b ein Teiler von a ist, dann ist $\gcd(a, b) = b$.
    (b) Wenn r der Rest bei der ganzzahligen Division von a durch b ist, so ist $\gcd(a, b) = \gcd(b, r)$.

    Schreibe eine Funktion `gcd`, die den grössten gemeinsamen Teiler auf Basis des Euklid'schen Algorithmus berechnet. Siehe auch `http://www.cut-the-knot.org/blue/Euclid.shtml`.
15. Schreibe eine Funktion, die zu einer gegebenen Folge ganzer Zahlen die Anzahl der ungeraden Zahlen in dieser Folge berechnet. (Hilfe: für eine Folge `x` von Wahrheitswerten liefert `sum(x)` die Anzahl der `TRUE`s.)

16. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl deren Ziffernsumme berechnet. Anleitung: die Einerstelle einer natürlichen Zahl kann man durch ganzzahlige Division extrahieren (`%%`) oder eliminieren (`%/%`).

17. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl n jene Folge von Wahrheitswerten erzeugt, in der absteigend für jede natürliche Zahl $i \leq n$ i-mal `TRUE` beziehungsweise `FALSE` wiederholt wird, je nachdem ob i gerade oder ungerade ist.

18. Schreibe eine Funktion `lcm`, die zu zwei gegebenen natürlichen Zahlen deren kleinstes gemeinsames Vielfaches (englisch: Least Common Multiple) liefert. (Hilfe: leite aus den Primfaktorzerlegungen eine einfache Beziehung zwischen den Zahlen und ihrem größten gemeinsamen Teiler ab.)

19. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl liefert ob diese eine Potenz der Zahl 2 ist (also von der Form 2^k mit einer nichtnegativen ganzen Zahl k).

20. Schreibe eine Funktion, die zu zwei gegebenen Zahlen einen Wahrheitswert liefert der angibt ob diese „relativ prim“ sind (also deren grösster gemeinsamer Teiler 1 ist).

21. Schreibe eine Funktion `is_prime`, die zu einer gegebenen natürlichen Zahl liefert ob diese eine Primzahl (also nur durch 1 und sich selbst teilbar ist) oder nicht.

22. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl n alle Primzahlen liefert, die nicht größer als n sind. (Versuche eine effiziente Implementierung.)

23. *Rationale Zahlen* sind Brüche der Form p/q mit $q > 0$ und $\gcd(p, q) = 1$. Wir können solche Zahlen als Folgen `c(p, q)` der Länge 2 repräsentieren. Schreibe eine Funktion `make_rat`, die zu zwei gegebenen ganzen Zahlen `p` und `q` die ensprechende rationale Zahl als Folge der Länge 2 zurückliefert (also beispielsweise `make_rat(19, -95)`: `c(-1, 5)`). Schreibe auch Funktionen `numer` und `denom` die für eine solche rationale Zahl den Zähler und Nenner liefern.

24. Schreibe Funktionen `add_rat` und `mul_rat` für die rationale Addition und Multiplikation rationaler Zahlen.

25. Schreibe eine Funktion, die vermittles einer geeigneten Siebmethode zu einer gegebenen natürlichen Zahl n alle Primzahlen liefert, die nicht größer als n sind.

26. Bestimme die kleinste und grösste ganze Zahl die mit 32-bit integers dargestellt werden können.

27. Zahlen der Form $M_n = 2^n - 1$ heissen *Mersenne*-Zahlen, und sind „Kandidaten“ für Primzahlen. Bestimme die ersten 7 Mersenne'schen Primzahlen.

28. Schreibe eine Funktion, die zu gegebenen natürlichen Zahlen n und k die ersten n Glieder der Reihe $s_i = 1^k + \cdots + i^k$ liefert. (Hinweis: `?cumsum`.)

29. Schreibe eine Funktion, die zu einer gegebenen Folge von ganzen Zahlen liefert ob diese der Größe nach sortiert sind oder nicht.

30. Jede natürliche Zahl n besitzt eine Primfaktorzerlegung der Form $\prod p_k^{e_k}$, wobei die Primzahlen p_k die Zahl n genau e_k mal teilen. Schreibe eine Funktion `factorize` die zu einer gegebenen natürlichen Zahl die Folge der Primfaktoren p_k mit den entsprechenden Vielfachheiten e_k liefert (also beispielsweise für $n = 120$ die Folge `c(2, 2, 2, 3, 5)`).

31. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl x die entsprechende Binärdarstellung (als Folge von 0/1 Zahlen) liefert. (Also e.g. für $x = 6$: `c(1, 1, 0)`.) Implementiere auch die zugehörige Umkehrfunktion.

32. Ein *Kettenbruch* ist ein Ausdruck der Form

$$x_1 + 1/(x_2 + 1/(x_3 + \cdots + 1/(x_{n-1} + 1/x_n)))$$

mit geeigneten ganzen Zahlen $x_1, \ldots, x_n$. Schreibe eine Funktion, die zu einer Folge ganzer Zahlen den Wert des entsprechenden Kettenbruches berechnet. Verwende die Kettenbruchentwicklung

$$(3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, \ldots)$$

von π um geeignete Approximationen von π zu erhalten, und vergleiche sie mit dem Wert `pi`. (Siehe e.g. `http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/cfINTRO.htm` oder auch `http://en.wikipedia.org/wiki/Continued_fraction`.)

33. Schreibe eine Funktion, die zu einer Folge ganzer Zahlen den Wert des entsprechenden Kettenbruches als rationale Zahl berechnet. Welche rationale Approximation ergibt dies für den Anfang der Kettenbruchentwicklung $(3, 7, 15, 1, 292, \ldots)$ von π?

34. Schreibe eine Funktion, die zu einer gegebenen Zahl x zwischen 0 und 1 die Folge der Koeffizienten $a_i \in \{0, 1\}$ in der "besten" dyadisch rationalen Approximation $\sum_{i=1}^{k} a_i/2^i$ der Länge k von x liefert. (Also e.g. für $x = 3/8 = 0/2 + 1/4 + 1/8 + 0/16$ und $k = 4$: `c(0, 1, 1, 0)`.)

35. Schreibe Funktionen die durch wiederholte Multiplikation mit 2 beziehungsweise Division durch 2 versuchen, die grösste und kleinste positive Gleitkommazahl zu bestimmen die in R dargestellt werden kann. Welchen Wert haben diese Zahlen?

36. Schreibe Funktionen die durch wiederholte Division die kleinsten positiven Gleitkommazahlen x bestimmen sodass in R $1 + x \neq 1$ beziehungsweise $1 - x \neq 1$. Welchen Wert haben diese Zahlen?

37. Finde die größte positive Gleitkommanzahl, die im IEEE 754 64-bit Standard dargestellt werden können. (Siehe `http://en.wikipedia.org/wiki/IEEE_754` und `docs.sun.com/source/806-3568/ncg_goldberg.html`.)

38. Finde die kleinste normalisierte positive Gleitkommazahl (IEEE 754 64-bit) ϵ sodass $1-\epsilon \neq 1$.

39. Finde drei Gleitkommazahlen a, b und c deren Addition nicht assoziativ ist (für die also $(a + b) + c \neq a + (b + c)$).

40. Versuche, $\sum_{n=1}^{\infty} 1/10^n$ möglichst genau zu berechnen.

41. Versuche, $1 - 1/2 + 1/3 - 1/4 + \cdots = \sum_{n=0}^{\infty} (-1)^n/(n + 1)$ möglichst genau zu berechnen.

42. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl n (≤ 26) eine Liste liefert, deren i-tes Element die Folge der ersten i Kleinbuchstaben ist.

43. Wir können Polynome $p(t) = a_0 + a_1 t + \cdots + a_n t^n$ durch die Folgen $(a_0, a_1, \ldots, a_n)$ ihrer Koeffizienten repräsentieren. Schreibe eine Funktion `polymult`, die zu zwei Polynomen p und q die Koeffizientenfolge deren Produktes $p(t)q(t)$ berechnet.

44. Schreibe eine Funktion `polyderiv`, die zu einem Polynom die Koeffizientenfolge seiner Ableitung berechnet. Verwende dies in Kombination mit `polyroot`, um die Wendepunkte eines gegebenen Polynoms zu berechnen.

45. Zeichenketten kann man vermittels `paste` zusammenfügen. Schreibe eine Funktion, die zu einer gegebenen natürlichen Zahl x die entsprechende Binärdarstellung als 0/1 Zeichenkette liefert. (Also e.g. für $x = 6$: `"110"`.)

46. Schreibe eine rekursive Funktion, die einer gegebenen natürlichen Zahl x die entsprechende Oktaldarstellung (als Folge der Ziffern von 0 bis 7) liefert.

47. Schreibe eine *rekursive* Funktion, die zu einer gegebenen natürlichen Zahl die Folge ihrer Ziffern liefert, also beispielsweise für $n = 12345$ die Folge `c(1, 2, 3, 4, 5)`. (Anleitung: bedenke, wie man die Einerstelle extrahiert beziehungsweise eliminiert.)

48. Die Funktion `logstar(x)` ist definiert als die minimale Anzahl n sodass n-maliges logarithmieren von x einen Wert kleiner als 1 liefert. (Also beispielsweise für $x = 123$ ist logstar(123) = 3, denn $\log(\log(123)) = 1.571151 \geq 1$ und $\log(\log(\log(123))) = 0.4518085 < 1$. Implementiere `logstar` als rekursive Funktion.

49. Schreibe eine rekursive Funktion, die den grössten gemeinsamen Teiler zweier natürlicher Zahlen vermittels des euklidischen Algorithmus berechnet.

50. Die Fibonacci Zahlen F_n sind durch die Rekursion $F_n = F_{n-1} + F_{n-2}$ with $F_1 = F_2 = 1$ gegeben. Schreibe eine Funktion, die für eine gegebene natürliche Zahl n die n-te Fibonacci Zahl liefert.

51. Schreibe eine Funktion, die für eine gegebenen natürliche Zahl n die Folge der ersten n Fibonacci Zahlen liefert. (Anleitung: überlege, wie man von der Folge der ersten $n-1$ Zahlen auf die Folge der ersten n Zahlen kommt.)

52. Schreibe eine Funktion `count_chars_in_table`, die zu jedem Element einer gegeben Folge `x` von Zeichenketten und einer Folge `table` jeweils die Anzahl der Zeichen in dem Element die in `table` vorkommen liefert. (Also beispielsweise `count_chars_in_table("Wien", letters[1:13])` $\Rightarrow 2$, weil nur ‚e' und ‚i' unter den ersten 13 Kleinbuchstaben sind.) Siehe `?match` zur Vereinfachung.

53. Schreibe eine Funktion, die zu einer gegebenen Folge von Zeichenketten die Folge der jeweiligen Anzahlen der Großbuchstaben liefert.

54. Schreibe eine Funktion, die herausfindet, wie viele Elemente einer gegebenen Folge von Zeichenketten aus mehreren Wörtern bestehen. Der Standarddatensatz `state.name` gibt die Namen der US-amerikanischen Bundesstaaten: wieviele davon bestehen aus mehreren Wörtern?

55. Wieviele US-amerikanische Bundesstaaten haben Namen mit mehr Vokalen als Konsonanten?

56. Welche US-amerikanischen Bundesstaaten haben Namen, in denen mehr als 3 ‘s’ vorkommen?

57. Wie viele US-amerikanische Bundesstaaten haben Namen, in denen ein Doppelvokal vorkommt?

58. Die Funktion `colors()` liefert alle dem Namen nach verfügbaren Farben. Berechne wie viele verschiedene “Rot” darunter sind.

59. Schreibe eine Funktion `string_reverse`, die eine gegebene Zeichenkette umdreht (e.g., `string_reverse("ABC")` sollte `"CBA"` liefern).

60. Schreibe eine Funktion, die den häufigsten Buchstaben in einer gegebenen Datei (e.g., http://statmath.wu.ac.at/~hornik/local.html) findet. (Hilfe: Dateien und URIs kann man mit `readLines` zeilenweise einlesen; `?table` für Häufigkeitstabellen.)

61. Schreibe eine Funktion, die zu einer gegebenen Folge von Zeichenketten die Folge der Anzahlen der Wörter (entstanden durch Zerlegung der Zeichenketten anhand der Leerzeichen) liefert. (Anleitung: `?strsplit`, `?lapply` oder eine explizite Schleife.)

62. Schreibe eine Funktion `wc`, die abzählt, wie wiele (verschiedene) Wörter in einer gegebenen Datei vorkommen. (Auf Interpunktionszeichen nicht vergessen!)

63. Schreibe eine Funktion, die abzählt, wie oft ein gegebenes Wort in einer gegebenen Datei vorkommt (unabhängig von Groß- oder Kleinschreibung).

64. Schreibe eine (rekursive) Funktion `count(pat, str)`, die abzählt, wie oft eine Zeichenkette `pat` in einer zweiten Zeichenkette `str` vorkommt, also beispielsweise `count("ab", "abab cd cdab")` $\Rightarrow 3$. (Anleitung: `?regexpr`.)

65. Schreibe eine rekursive Funktion zur Erzeugung aller „Kombinationen“ von k aus n Elementen in der Form von 0/1 „Inzidenz“ Zeichenketten (Zeichenketten die aus k `"1"` und $n-k$ `"0"` bestehen; das i-te Zeichen gibt an ob das i-te Element in der jeweiligen Kombination vorkommt (`"1"`) oder nicht). Beispielsweise für $n = 3$ und $k = 2$: `c("110", "101", "011")`.

66. Schreibe eine Funktion `polyprint`, die anhand der gegebenen Koeffizientenfolge $(a_0, a_1, \ldots, a_n)$ eines Polynoms dieses in der üblichen Form $a_0 + a_1 x + \cdots + a_n x^n$ ausgibt, also e.g. für die Folge `c(-1, 0, 1)`: `-1 + x^2`. (Beachte: Koeffizienten und Exponenten mit dem Wert 1 werden üblicherweise nicht gezeigt.)

67. Schreibe eine Funktion `rat`, die zu zwei gegebenen ganzen Zahlen `p` und `q` die ensprechende rationale Zahl p/q als Folge der Länge 2 mit geeigneter Klasse (e.g., `"rat"`) zurückliefert.

68. Schreibe eine (S3) `print` Methode für rationale Zahlen.

69. Schreibe (S3) Methoden zur Addition (`+`) und Multiplikation (`*`) von rationalen Zahlen.

70. Schreibe eine Funktion `polynomial`, die zu einer gegebenen Koeffizientenfolge $(a_0, a_1, \ldots, a_n)$ eines Polynoms $a_0 + a_1 x + \cdots + a_n x^n$ dieses als Koeffizientenfolge mit geeigneter Klasse (e.g., `"polynomial"`) zurückliefert.

71. Schreibe (S3) Methoden zur Addition (`+`) und Multiplikation (`*`) von Polynomen.

72. Schreibe eine (S3) `print` Methode für Polynome.