10. Die Hofstadter-Funktion ist rekursiv definiert (n natürliche Zahl):

    Rekursionsanfang: hof(1) = 1
    hof(2) = 1

    Rekursionsvorschrift: $hof(n) = hof[n - hof(n-1)] + hof[n - hof(n-2)],\ n>2$

    Formuliert man den Algorithmus zur Berechnung der Hofstadter-Funktion als Python-Programm mit rekursivem Funktionsaufruf, haben wir die Erfahrung gemacht, daß die Rechenzeit für große Werte von n sehr schnell wächst; der Grund ist die mit n sehr schnell wachsende Anzahl gleichzeitig aktiver Aufrufe der Funktion hof.

    Dieses ungünstige Laufzeitverhalten läßt sich umgehen, indem man den Algorithmus zur Berechnung der Hofstader-Funktion iterativ formuliert.

    Vorschlag zur iterativen Formulierung:
    Definiere ein array a mit den Komponenten a[0], a[1], a[2], ... und setze a[0] = hof(1) = 1, a[1] = hof(2) = 1.
    Den weiteren Komponenten a[2], a[3], . . . werden in dieser Reihenfolge die Werte hof(3), hof(4), ... zugewiesen.

    Konzipiere und teste das iterativ formulierte Python-Programm!

11. Zusatzaufgabe:

    Die Fibonacci-Folge $\{a_i\}$ ist wie folgt definiert:

    $$\mathbf{a_1 = a_2 = 1}$$

    $$\mathbf{a_n = a_{n-1} + a_{n-2}} \quad \text{für } n \geq 3$$

    Schreibe und teste ein Python-Programm zur Berechnung der Fibonacci-Folge.

12. Der als Python-Programm formulierte Algorithmus
    sorting_by_direct_selection.py.txt auf
    https://kalle2k.lima-city.de/computerscience/Informatik_12/sorting/
    sortiert ein array von Zufallszahlen aufsteigend, d. h. die sortierte Liste beginnt mit dem kleinsten Element.

    Modifiziere das Programm so, daß das Sortieren absteigend erfolgt.