Aufgabe 1

Zeige schrittweise, wie die Zahlen in der Liste `L = [9, 3, 6, 8, 7]` mit dem Selectionsort-Verfahren aufsteigend sortiert werden. Wie viele Vergleiche und Vertauschungen sind dafür insgesamt nötig?

Aufgabe 2

Zeige schrittweise, wie die Zahlen in der Liste `L = [9, 3, 6, 8, 7]` mit dem Insertionsort-Verfahren aufsteigend sortiert werden. Wie viele Vergleiche und wie viele Verschiebungen sind dafür insgesamt nötig?

Aufgabe 3

Zeige schrittweise, wie die Zahlen in der Liste `L = [9, 3, 6, 8, 7]` mit dem Bubblesort-Verfahren aufsteigend sortiert werden. Wie viele Vergleiche und Vertauschungen sind dafür insgesamt nötig?

Aufgabe 4

(a) Zeige, wie der Partitionierungsschritt von Quicksort den folgenden Listenausschnitt verarbeitet. Protokolliere dein Vorgehen, indem du jeweils eine neue Zeile beginnst, wenn ein Element (evtl. auch an Ort und Stelle) getauscht wird.

...	2	8	3	9	4	6	...

(b) Welchen Wert gibt der Partitionierungsschritt zurück wenn das erste Element der obigen Liste (also 2) den Index $i = 100$ hat?

Aufgabe 5

Zeige, wie Quicksort die Liste

`4, 8, 7, 1, 3, 9, 5`

sortiert. Protokolliere jeden Vertauschungsschritt und mache deutlich, in welcher Reihenfolge die Teillisten sortiert werden.

Aufgabe 6

Wie viele Vergleiche und wie viele Vertauschungen benötigt Selectionsort für eine Liste mit 7 Elementen,

(a) die aufsteigend sortiert ist,

(b) die absteigend sortiert ist?

Aufgabe 7

Wie viele Vergleiche und wie viele Speicheroperationen benötigt Insertionsort für eine Liste mit 7 Elementen,

(a) die aufsteigend sortiert ist,

(b) die absteigend sortiert ist?

Aufgabe 8

Wie viele Vergleiche und wie viele Vertauschungen benötigt Bubblesort für eine Liste mit 7 Elementen,

(a) die aufsteigend sortiert ist,

(b) die absteigend sortiert ist?

Aufgabe 9

Wie viele Vergleiche benötigt Quicksort für den Partitionierungsschritt einer Teilliste mit 20 Elementen?

Aufgabe 10

Welcher Sortieralgorithmus wird mit dem folgenden Python-Programm implementiert?

```
def sort(A):
    n = len(A)
    for i in range(0, n-1):
        for j in range(0, n-i-1):
            if A[j] > A[j+1]:
                A[j], A[j+1] = A[j+1], A[j]
```

Aufgabe 11

Welcher Sortieralgorithmus wird mit dem folgenden Python-Programm implementiert?

```
def sort(A):
    n = len(A)
    for i in range(0, n-1):
        k = i
        for j in range(i+1, n):
            if A[j] < A[k]:
                k = j
        A[k], A[i] = A[i], A[k]
```

Aufgabe 12

Bestimme anhand der ersten 4 Verarbeitungsschritte, um welchen Sortieralgorithmus es sich handelt und gib eine kurze Begründung. Beachte:

- Nicht alle Algorithmen sortieren die Liste bis zum Ende.
- Es werden nur Schritte dargestellt, die *potenziell* Elemente vertauschen oder verschieben. Dabei kann es vorkommen, dass ein Element mit sich selbst vertauscht oder um null Positionen verschoben wird.
- Ein Algorithmus wird höchstens einmal verwendet.

(a)

4	5	1	3	2
4	5	1	3	2
1	4	5	3	2
1	3	4	5	2
1	2	3	4	5

(b)

4	5	1	3	2
1	5	4	3	2
1	2	4	3	5
1	2	4	3	5
1	2	4	3	5

(c)

4	5	1	3	2
1	5	4	3	2
1	2	4	3	5
1	2	3	4	5
1	2	3	4	5

Aufgabe 13

Gib die ungefähre Sortierdauer der von uns behandelten Sortieralgorithmen für den besten und den schlechtesten Fall in Abhängigkeit der Arraylänge n und mit einer Proportionalitätskonstante C an.

	Worst Case	Best Case
Selectionsort		
Bubblesort		
Quicksort		
Insertionsort		

Aufgabe 14

Eine Programm, das Arrays mit der Laufzeit $C \cdot n^2$ sortiert, benötigt auf einem Computer ungefähr 0.1 Sekunden, um eine Liste mit 1000 Zufallszahlen zu sortieren. Schätze, wie lange dasselbe Programm benötigt, um auf dem gleichen Computer eine Liste mit 7000 Zufallszahlen zu sortieren.

Aufgabe 15

Beschreibe, für welche Art von Listen der Quicksort-Algorithmus die Worst Case-Laufzeit aufweist?

Aufgabe 16

Beschreibe zwei Pivotstragegien für das Quicksort-Verfahren, um die Worst Case-Laufzeit zu verhindern.

Aufgabe 17

Bestimme den Median der folgenden Listen.

(a) `[8, 12, 27, 3, 16, 29, 17]`

(b) `[23, 14, 7, 9, 55, 6, 2, 12]`

Aufgabe 18

Schreibe eine Python-Funktion `isSorted(A)`, mit einer Liste `A` als Argument, die `True` zurückgibt, wenn die Liste bereits sortiert ist und `False` sonst.

Aufgabe 19

Schreibe eine Python-Funktion `findMinPos(A)`, mit einer Liste `A` als Argument, die den Index des kleinsten Elements in der Liste als Wert zurückgibt.