

1. (3 Punkte) Sortiere die Zahlenfolge mit SelectionSort. Schreibe für die ersten drei Durchgänge je eine Zeile.

13 4 92 42 11 7 12

Lösung:

```
4 13 92 42 11 7 12
4 7 92 42 11 13 12
4 7 11 42 92 13 12
```

2. (4 Punkte) Notiere den Code, der für den SelectionSort-Algorithmus fehlt. Gib auch die Stufe der Einrückung an.

```
def selection_sort(a):           #E0
    for i in range(len(a)-1):     #E1
        pos = i                  #E2
        min = a[i]               #E2
                                ???(1)
        if a[j] < min:            #E3
            pos = j              #E4
            min = a[j]           #E4
                                ???(2)
```

Lösung:

```
(1) for j in range(i+1, len(a))   #E2
(2) a[pos], a[i] = a[i], a[pos]   #E2
```

3. (3 Punkte) Sortiere die Zahlenfolge mit BubbleSort. Schreibe für die ersten drei Durchgänge je eine Zeile.

2 22 14 25 1 13 9

Lösung:

```
2 14 22 1 13 9 25
2 14 1 13 9 22 25
2 1 13 9 14 22 25
```

4. (4 Punkte) Notiere den Code, der für den BubbleSort-Algorithmus fehlt. Gib auch die Stufe der Einrückung an.

```
def bubble_sort(a):
    #E0
    ???(1)
    while getauscht:
        #E1
        getauscht = False
        #E2
        for i in range(len(a)-1):
            #E2
            ???(2)
            #E4
            a[i], a[i+1] = a[i+1], a[i]
            #E4
            getauscht = True
```

Lösung:

```
(1) getauscht = True      #E1
(2) if a[i] > a[i+1]:    #E3
```

5. (5 Punkte) Die Liste `a = [24, 4, 17, 88, 42, 12, 7]` wird mit dem rekursiven `mergeSort`-Algorithmus aus dem Unterricht sortiert.

- Wieviel mal wird `merge` aufgerufen?
- Wieviel mal wird `mergeSort` aufgerufen? (der erste Aufruf zählt mit).

Lösung: a. 6 b. 13

Bei jedem Aufruf von `merge` wird eine Liste zurückgegeben. Notiere die Listen in der Reihenfolge, in der sie zurückgegeben werden.

Lösung:

```
4 17
4 17 24
42 88
7 12
7 12 42 88
4 7 12 17 24 42 88
```

6. (5 Punkte) Die Liste `22 41 43 7 42 19` wird mit `quickSort` sortiert. Schreibe die ersten drei Protokollzeilen.

Lösung:

```
22 41 43 7 42 19    - 0-5, pivot=43
22 41 19 7 42 43    - 0-4, pivot=19
7 19 41 22 42 43    - 0-1, pivot= 7
```

7. (2 Punkte) Der nächste Quicksort-Durchgang bearbeitet die Liste von 0-3.

15 8 9 16 18 28 22 38 26

Schreibe die Protokollzeilen vor und nach dem Durchgang.

Lösung:

15 8 9 16 18 28 22 38 26 — 0-3, pivot= 8
8 15 9 16 18 28 22 38 26 — 1-3, pivot= 9

8. (2 Punkte) Quicksort erhält die Liste zur Sortierung. Schreibe die Protokollzeilen vor und nach dem ersten Durchgang.

15 26 22 18 16 28 9 38 8

Lösung:

15 26 22 18 16 28 9 38 8 — 0-8, pivot=16
15 8 9 16 18 28 22 38 26 — 0-3, pivot= 8

9. (2 Punkte) Mache aus der Liste einen Heap nach dem Verfahren aus dem Unterricht.

12 6 3 17 42 5 25 38 9 67 54 1 81

Lösung: 1 6 3 9 42 5 25 38 17 67 54 12 81

10. (4 Punkte) Eine Liste wird mit HeapSort sortiert. Die Zeile zeigt die in einen Heap umgewandelte Liste. Füge für die beiden folgenden Reorganisationen je eine Zeile hinzu.

1 6 4 17 28 33 20 92

Lösung:

4 6 20 17 28 33 92 1
6 17 20 92 28 33 4 1

11. (2 Punkte) Welche Komplexität hat die Laufzeit von BubbleSort im best, worst und average-case? In welcher Komplexitätsklasse ist der zusätzliche Platzbedarf?

Lösung:

best: $O(n)$, average: $O(n^2)$, worst: $O(n^2)$, Platz: $O(1)$