

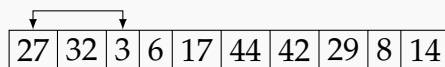
Aufgabe 4

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen *aufsteigend* sortiert werden. Das Feld habe n Elemente A[1] bis A[n]. Der folgende Algorithmus sei gegeben:

```
1  var A : array[1..n] of integer;
2
3  procedure selection_sort
4  var i, j, smallest, tmp : integer;
5  begin
6      for j := 1 to n-1 do begin
7          smallest := j;
8          for i := j + 1 to n do begin
9              if A[i] < A[smallest] then
10                 smallest := i;
11            end
12            tmp = A[j];
13            A[j] = A[smallest];
14            A[smallest] = tmp;
15        end
16    end
```

- (a) Sortieren Sie das folgende Feld mittels des Algorithmus. Notieren Sie alle Werte, die die Variable *smallest* jeweils beim Durchlauf der inneren Schleife annimmt. Geben Sie die Belegung des Feldes nach jedem Durchlauf der äußeren Schleife in einer neuen Zeile an.

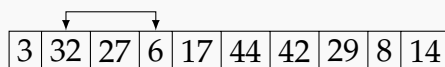
Ausgang



27	32	3	6	17	44	42	29	8	14
----	----	---	---	----	----	----	----	---	----

nach 1. Durchlauf (j = 1)

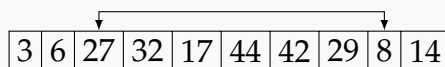
smallest: (1) 3



3	32	27	6	17	44	42	29	8	14
---	----	----	---	----	----	----	----	---	----

nach 2. Durchlauf (j = 2)

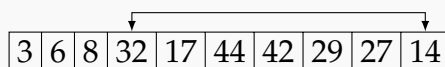
smallest: (2) 3 4



3	6	27	32	17	44	42	29	8	14
---	---	----	----	----	----	----	----	---	----

nach 3. Durchlauf (j = 3)

smallest: (3) 5 9



3	6	8	32	17	44	42	29	27	14
---	---	---	----	----	----	----	----	----	----

nach 4. Durchlauf (j = 4)

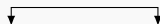
smallest: (4) 5 10

3	6	8	14	17	44	42	29	27	32
---	---	---	----	----	----	----	----	----	----

nach 5. Durchlauf ($j = 5$)

smallest: (5) -


3	6	8	14	17	44	42	29	27	32
---	---	---	----	----	----	----	----	----	----



nach 6. Durchlauf ($j = 6$)

smallest: (6) 7 8 9

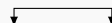
3	6	8	14	17	27	42	29	44	32
---	---	---	----	----	----	----	----	----	----



nach 7. Durchlauf ($j = 7$)

smallest: (7) 8

3	6	8	14	17	27	29	42	44	32
---	---	---	----	----	----	----	----	----	----



nach 8. Durchlauf ($j = 8$)


smallest: (8) 10

3	6	8	14	17	27	29	32	44	42
---	---	---	----	----	----	----	----	----	----

nach 9. Durchlauf ($j = 9$)

smallest: (9) 10

3	6	8	14	17	27	29	32	44	42
---	---	---	----	----	----	----	----	----	----



fertig

3	6	8	14	17	27	29	32	42	44
---	---	---	----	----	----	----	----	----	----

- (b) Der Wert der Variablen *smallest* wird bei jedem Durchlauf der äußeren Schleife mindestens ein Mal neu gesetzt. Wie muss das Feld *A* beschaffen sein, damit der Variablen *smallest* ansonsten niemals ein Wert zugewiesen wird? Begründen Sie Ihre Antwort.

Wenn das Feld bereits aufsteigend sortiert ist, dann nimmt die Variable *smallest* in der inneren Schleife niemals einen neuen Wert an.

- (c) Welche Auswirkung auf die Sortierung oder auf die Zuweisungen an die Variable *smallest* hat es, wenn der Vergleich in Zeile 9 des Algorithmus statt $A[i] < A[\text{smallest}]$ lautet $A[i] \leq A[\text{smallest}]$? Begründen Sie Ihre Antwort.

Der Algorithmus sortiert dann nicht mehr *stabil*, d. h. die Eingabereihenfolge von Elementen mit *gleichem Wert* wird beim Sortieren nicht mehr *bewahrt*.

- (d) Betrachten Sie den Algorithmus unter der Maßgabe, dass Zeile 9 wie folgt geändert wurde:

```
1  if A[i] > A[smallest] then
```

Welches Ergebnis berechnet der Algorithmus nun?

Der Algorithmus sortiert jetzt absteigend.

- (e) Betrachten Sie die folgende *rekursive* Variante des Algorithmus. Der erste Parameter ist wieder das zu sortierende Feld, der Parameter n ist die Größe des Feldes und der Parameter $index$ ist eine ganze Zahl. Die Funktion $min_index(A, x, y)$ berechnet für $1 \leq x \leq y \leq n$ den Index des kleinsten Elements aus der Menge $\{A[x], A[x+1], \dots, A[y]\}$

```
1  procedure rek_selection_sort(A, n, index : integer)
2  var k, tmp : integer;
3  begin
4  if (Abbruchbedingung) then return;
5    k = min_index(A, index, n);
6    if k <> index then begin
7      tmp := A[k];
8      A[k] := A[index];
9      A[index] := tmp;
10   end
11   (rekursiver Aufruf)
12 end
```

Der initiale Aufruf des Algorithmus lautet: `rek_selection_sort(A, n, 1)`

Vervollständigen Sie die fehlenden Angaben in der Beschreibung des Algorithmus für

- die Abbruchbedingung in Zeile 4 und

`n = index` bzw `n == index`

Begründung: Wenn der aktuelle Index so groß ist wie die Anzahl der Elemente im Feld, dann muss / darf abgebrochen werden, denn dann ist das Feld sortiert.

- den rekursiven Aufruf in Zeile 11.

`rek_selection_sort(A, n, index + 1)`

Am Ende der Methode wurde an die Index-Position $index$ das kleinste Element gesetzt, jetzt muss an die nächste Index-Position ($index + 1$) der kleinste Wert, der noch nicht sortierten Zahlen, gesetzt werden.

Begründen Sie Ihre Antworten.

```

3  import static org.bsclangaul.helfer.Konsole.zeigeZahlenFeld;
4
5  public class SelectionSort {
6
7      public static void selectionSort(int[] A) {
8          int smallest, tmp;
9
10         for (int j = 0; j < A.length - 1; j++) {
11             System.out.println("\nj = " + (j + 1));
12             smallest = j;
13             for (int i = j + 1; i < A.length; i++) {
14                 if (A[i] < A[smallest]) {
15                     smallest = i;
16                     System.out.println(smallest + 1);
17                 }
18             }
19             tmp = A[j];
20             A[j] = A[smallest];
21             A[smallest] = tmp;
22             zeigeZahlenFeld(A);
23         }
24     }
25
26     public static void rekSelectionSort(int[] A, int n, int index) {
27         int k, tmp;
28
29         if (index == n - 1) {
30             return;
31         }
32         k = minIndex(A, index, n);
33         if (k != index) {
34             tmp = A[k];
35             A[k] = A[index];
36             A[index] = tmp;
37         }
38         rekSelectionSort(A, n, index + 1);
39     }
40
41     public static int minIndex(int[] A, int x, int y) {
42         int smallest = x;
43         for (int i = x; i < y; i++) {
44             if (A[i] < A[smallest]) {
45                 smallest = i;
46             }
47         }
48         return smallest;
49     }
50
51     public static void main(String[] args) {
52         int[] A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
53         selectionSort(A);
54
55         A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
56         rekSelectionSort(A, A.length, 0);
57         zeigeZahlenFeld(A);
58     }
59 }
60

```