Data Structures 2018
Exercise 1, solutions (Week 37)

Algorithm 1 Maximum($A[0, 1, \ldots, n-1]$)

```
1 precondition: $n \geq 1$
$a \leftarrow A[0]$
for $i \leftarrow 1$ to $n - 1$ do
    if $a \leq A[i]$ then
        $a \leftarrow A[i]$
    end if
end for
return $a$
```

Algorithm 2 Minimum($A[0, 1, \ldots, n-1]$)

```
precondition: $n \geq 1$
$a \leftarrow A[0]$
for $i \leftarrow 1$ to $n - 1$ do
    if $a \leq A[i]$ then
        $a \leftarrow A[i]$
    end if
end for
return $a$
```

2 Check maxmin.java file.

3 (a) If we have positive real numbers $a_1, a_2, \ldots, a_n$ given, arithmetic mean can calculated as follows:

$$\frac{1}{n}\sum_{i=1}^{n} a_i.$$

Algorithm AriAve evaluates the arithmetic mean from the elements of an array A.

(b) If we have positive real numbers $a_1, a_2, \ldots, a_n$ given, harmonic mean can be calculated as follows:

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{a_i}}.$$

Algorithm HarAve evaluates the harmonic mean from the element of an array A.

(c) If we have positive real numbers $a_1, a_2, \ldots, a_n$ given, geometric mean can be calculated as follows:

$$\sqrt[n]{a_1 a_2 \cdots a_n}.$$

Algorithm GeoAve evaluates the geometric mean from the elements of an array A.

```
AriAve ($A[0, 1, \ldots, n-1]$)
precondition: $n > 0$
1: $result \leftarrow 0$
2: $sum \leftarrow 0$
3: for $i \leftarrow 0$ to $n-1$ do
4:     $sum \leftarrow sum + A[i]$
5: end for
6: $result \leftarrow sum/n$
7: return $result$
```

```
HarAve ($A[0, 1, \ldots, n-1]$)
precondition: $n > 0$
 1: $result \leftarrow 0$
 2: $sum \leftarrow 0$
 3: for $i \leftarrow 0$ to $n-1$ do
 4:     $sum \leftarrow sum + 1/A[i]$
 5: end for
 6: $result \leftarrow n/sum$
 7: return $result$
```

```
GeoAve ($A[0, 1, \ldots, n-1]$)
precondition: $n > 0$
 1: $result \leftarrow 0$
 2: $prod \leftarrow 1$
 3: for $i \leftarrow 0$ to $n-1$ do
 4:     $prod \leftarrow prod \cdot A[i]$
 5: end for
 6: $result \leftarrow \sqrt[n]{prod}$
 7: return $result$
```

4 Check Averages.java file.

5 There are different ways to describe the asked algorithm. Here is presented a simple but slightly inefficient way to do it. Example algorithm checks with each element in an array one by one if the value in process is the mode. The algorithm returns the mode value found last.

```
FindMode ($A[0, 1, \ldots, n-1]$)
precondition: $n > 0$
$modePos \leftarrow 0$
$modeCount \leftarrow 1$
for $i \leftarrow 1$ to $n-1$ do
    $count \leftarrow 0$
    for $j \leftarrow 0$ to $n-1$ do
        if $A[j] = A[i]$ then
            $count \leftarrow count + 1$
        end if
    end for
    if $count \geq modeCount$ then
        $modePos \leftarrow i$
        $modeCount \leftarrow count$
    end if
end for
return $A[modePos]$
```

6 There are many ways to accomplish the task. For example, Algorithm 1 is one possible solution. In the sorting task you can use, for instance, bubble-sort (algorithm 3) or selection-sort (algorithm 4). However, both of them are rather slow. More efficient sorting algorithms will be presented later on this course.

Algorithm 3 Sorts array A with bubble-sort. If the array contains elements that are in wrong order, they will be exchanged. The $(i+1)$th element will be in the proper place after the ith iteration. The algorithm can be stopped if no more exchanges take place. Letters T and F correspond to the logical constants 'true' and 'false', respectively.

```
BubbleSort ($A[0, 1, \ldots, n-1]$)
$i \leftarrow 0$
**repeat**
    exch $\leftarrow F$
    **for** $j \leftarrow 0$ **to** $n - 2 - i$ **do**
        **if** $A[j] > A[j+1]$ **then**
            exchange $A[j] \leftrightarrow A[j+1]$
            exch $\leftarrow T$
        **end if**
    **end for**
    $i \leftarrow i + 1$
**until** exch $= F$
```

Algorithm 4 Sorts array A with in-place selection-sort. Index m corresponds to the smallest element in the unsorted part of the array (i.e. $A[j], \ldots, A[n-1]$) after each iteration of the inner for-loop. This element will be then exchanged to its proper place.

```
SelectionSort ($A[0, 1, \ldots, n-1]$)
1: for $j \leftarrow 0$ to $n-2$ do
2:    $m \leftarrow j$
3:    for $i \leftarrow j+1$ to $n-1$ do
4:       if $A[i] < A[m]$ then
5:          $m \leftarrow i$
6:       end if
7:    end for
8:    exchange $A[m] \leftrightarrow A[j]$
9: end for
```

Algorithm 5 Computes the median of array A.

```
SelectMedian ($A[0, \ldots, n-1]$)
   BubbleSort($A$)
   if $n$ is odd then
      $r \leftarrow A[(n-1)/2]$.
   else
      $r \leftarrow (A[n/2] + A[n/2-1])/2$.
   end if
   return $r$
```

7 Check files Mode.java and Median.java.

8 See Alphaorder.java file.