Prof. Dr. Jasmin Blanchette
Dr. Martin Desharnais-Schäfer
Dr. Michael Kirsten
Elisabeth Lempa

Ludwig-Maximilians-Universität München
Institut für Informatik
Discussion on 03.12.2025
Homework due on 10.12.2025 at 16:00

Possible solution for Exercise Sheet 8 in Scientific and Technical English for Computer Scientists

The exercise sheets consist of in-class exercises and homework. The in-class exercises take place during the second half of the lecture time slots. The homework, which is optional and ungraded, can be submitted via the "Homework" section in Moodle. The homework is subject to peer review.

Unless indicated otherwise, generative artificial intelligence assistants such as ChatGPT may be used, as long as you acknowledge how you use them as specified by the Institute's policy on plagiarism.[1] However, you may not use such tools to generate peer reviews for you. In addition, we strongly recommend that you do not use them to generate entire solutions, since that would defeat the purpose of the exercises.

In-class exercise 8-1 ***Telephone Game*** The following two pages present the Java code for the bubble sort and selection sort algorithms.

a) Form a pair with another student.

b) Assign one of the algorithms to you and the other one to your teammate.

c) Read your algorithm and only that one.

d) Write a description of the inner workings of your algorithm. You can assume that the reader knows about the `swap` function.

e) Exchange the descriptions once you and your teammate have finished writing.

f) Write a Java program that exactly follows your teammate's description.

g) Once you and your teammate have finished programming, exchange your programs and compare them with those given on the following pages.

h) Discuss how effective the descriptions were at conveying the algorithms.

[1] `https://www.medien.ifi.lmu.de/lehre/Plagiate-IfI.pdf`

```java
static public void bubbleSort(int[] a) {
    final int n = a.length;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1]) {
                swap(a, j, j + 1);
            }
        }
    }
}
```

```java
static public void selectionSort(int[] a) {
    final int n = a.length;
    for (int i = 0; i < n - 1; i++) {
        int min = i;
        for (int j = i + 1; j < n; j++) {
            if (a[j] < a[min]) {
                min = j;
            }
        }
        if (min > i) {
            swap(a, i, min);
        }
    }
}
```

POSSIBLE SOLUTION:

The `bubbleSort` function implements an in-place, comparison-based sorting algorithm that iterates through a given array, compares adjacent elements, and exchanges them if they are in the wrong order. At each iteration, larger elements "bubble" toward the end of the array until, finally, the array is sorted. The function takes an array of integers as its sole parameter and does not return any value. It consists of the following parts:

Initialization In the initialization phase, the array's length is stored in the variable n.

Outer loop The algorithm consists of an outer loop that iterates over all integers i such that $0 \leq i < n$.

Inner loop Inside the outer loop, there is an inner loop that iterates over all integers j such that $0 \leq j < n - i - 1$.

Comparison and swap Inside the inner loop, a condition checks whether the element at position j in the array is strictly greater than the element at position $j + 1$. In that case, the two elements are exchanged using the auxiliary `swap` function.

The `selectionSort` function implements an in-place, comparison-based sorting algorithm that iteratively extends a sorted prefix of the array by searching for the smallest element beyond the sorted prefix and exchanging it with the first unsorted element, thereby extending the sorted prefix. At each iteration, the size of the sorted prefix increases by one until the prefix encompasses the entire array. The function takes an array of integers as its sole parameter and does not return any value. It consists of the following parts:

Initialization In the initialization phase, the length of the array is stored in the variable n.

Outer loop The algorithm consists of an outer loop that iterates over all integers i such that $0 \leq i < n - 1$. The loop ensures that the prefix of the array between 0 and i is sorted. At the beginning, $i = 0$ and the prefix is empty.

Inner loop Inside the outer loop, the variable m contains the index of the smallest element found beyond the prefix. It is first initialized to i. Then, there is an inner loop that iterates over all integers j such that $i + 1 \leq j < n$. Inside the inner loop, a conditional is used to check whether the element at position j in the array is strictly smaller than the element at position m. If the condition is true, the value j is assigned to the variable m.

Comparison and swap After the inner loop, the variable m now contains the index of the smallest element beyond the prefix. A conditional now checks whether the position m is strictly greater than the position i. If the condition is true, the two elements are exchanged using the auxiliary `swap` function.

Discussion Each description starts with a brief high-level explanation of its algorithm. This allows the reader to quickly know what the function implements (i.e., sorting algorithms), what the main characteristics are (i.e., the sorting is in-place and based on comparisons), and what the general ideas are (e.g., iterations over the array, large elements "bubbling" toward the end, the creation of a sorted prefix).

The descriptions are largely independent of the programming language: For example, they could be used to describe Python programs. This is achieved by using a structural and mathematical description of the steps and concepts. The mathematical description is also often easier to understand (e.g., a loop that iterates over all integers i such that $0 \leq i < n$ versus `for (int i = 0; i < n; i++)`). A perhaps even more readable alternative would have been to use pseudocode.

The descriptions also explain concepts not immediately apparent in the code. For example, when describing the outer loop of the `selectionSort` function, we clearly identified the sorted prefix of the array and presented the special case of the empty prefix.