Technische Hochschule Ingolstadt SS 2025
Study Program "Autonomous Vehicle Engineering"
Programming 2

Assignment 4 -- Arrays (part 2)

The aim of the tasks is to learn how to move data items around within array variables.

1) Create two integer arrays (named input and output); the size of the arrays and the initialization of input with arbitrary integers you do as you like (reading in or assigning in the program text; for the demonstration at least ten different numbers should be involved).

At the beginning, all values are valid in input and none in output; we record this in the variables in_len and out_len. Your program should now individually move all values from input to output, logging the state of both arrays in each step, like this:

```
input              -->             output
[ 10 -7 16 -3 -4 -1 17 99 -9 80 ] --> [ ]
[ 10 -7 16 -3 -4 -1 17 99 80 ] --> [ -9 ]
[ 10 16 -3 -4 -1 17 99 80 ] --> [ -7 -9 ]
. . .
[ 99 ] --> [ 80 17 16 10 -1 -3 -4 -7 -9 ]
[ ] --> [ 99 80 17 16 10 -1 -3 -4 -7 -9 ]
```

For each movement, find the smallest element from input (that what is still left), take it out, and then put it into the very first position of the output array (see example above). At the end, there's nothing left in input and all data elements are sitting in the output array.

2) You will have noticed in task 1 that both arrays together have 2*n memory cells, but we store only n data elements at each point in time. Perhaps you thought: this can be done in a more space-saving way! -- That is exactly what you are supposed to do now.

Instead of input and output we use one array called data (of size n), which simultaneously holds both the input and the output data. As you see in the logging, there is no visible borderline between the input on the left and the output on the right side (of course, your program has to know where the borderline is!):

```
array contents   --   input length   --   smallest input item
[ 10 -7 16 -3 -4 -1 17 99 -9 80 ] -- in-len=10; min-item: -9
[ 10 -7 16 -3 -4 -1 17 99 80 -9 ] -- in-len=9; min-item: -7
[ 10 16 -3 -4 -1 17 99 80 -7 -9 ] -- in-len=8; min-item: -4
. . .
[ 99 80 17 16 10 -1 -3 -4 -7 -9 ] -- in-len=1; min-item: 99
[ 99 80 17 16 10 -1 -3 -4 -7 -9 ]
```

Don't forget to test your program with different numbers (also change n, the array size).