Program Verification Prof. Dr. H. Peter Gumm
Summer 2023

Exercise sheet 09

Deadline: June 24, 8:00 p.m.

Please submit only a Dafny-file **ex09_*your_name*.dfy**.

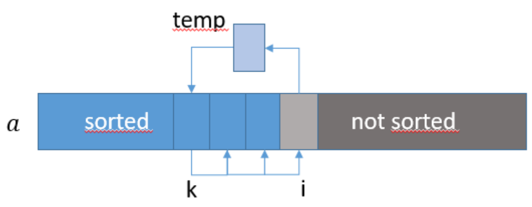
Problem 1 (4 points)**.** Read chapters 14 and 15 in the Book.

Problem 2 (6 pts)**.** Do exercise 14.11 in the book, that is specify, implement and verify a method

```
method rotateLeft<T>(a:array<T>)
  modifies a
```

which moves every entry of the array by one position to the left, i.e. the old value at `a[(i+1)% a.Length]` gets stored at `a[i]`. It may be helpful, to add `assert` statements to your code. Once everything verifies, you can check, which of these can be removed again. (In my solution, I needed to have at last assert statement in my code the condition that was ensured. Study the earlier examples in the book carefully, to get an idea, which invariants you may likely forget.)

Problem 3 (6 pts)**.** (This is essentially Exercise 15.9 in the book.) In class we discussed *InsertionSort*, where the first element of the non-sorted range (position i in the figure below) was bubbled down by repeatedly swapping with its lower neighbour until its correct position was reached (position k in the figure). A variant of this algorithm stores $a[i]$ in a temporary variable, then moves each element of the range $a[k..i]$ up by one position and finally store *temp* at $a[k]$.



Here it is important that you avoid swapping. This saves a lot of memory accesses and makes the algorithm more efficient. Still you must verify, that in the end the array is sorted. (This example took me a bit longer, although it comes out very elegantly in the end. Therefore, you are ***not required*** to prove that `ensures multiset(a[..]) == multiset(old(a[..]))`.)