Homework Assignment 1

Due Date: January 24, 2021, 23:59

Note. Please note that this semester all assignments are group assignments. Further note that for the grading we will apply a "10%" rule, i.e. the maximum number of points for this assignments is 55, but 50 will be counted as 100%. Points that exceed 50 will be stored in a separate counter and used later for compensation of lost points in other assignments or (if not used up this way) the final exam.

EXERCISE 1. For $k \in \mathbb{N}, k \geq 1$ define an operation *delete_last*(ℓ, k), which deletes the last k list elements in ℓ.

(i) Analyse the amortised complexity of this operation and show that it is in $\Theta(1)$, independent of k.

(ii) Implement a method *delete_last*(ℓ, k) on the class ALIST or DLIST.

total points: 10

EXERCISE 2. Explore structural recursion on list objects, i.e. define an operation $src[e, f, g]$ in the following way:

- If ℓ is the empty list, then $src[e, f, g](\ell) = e$, where $e \in T'$ is some constant.
- If ℓ is a singleton list containing just one element x, then $src[e, f, g](\ell) = f(x)$, where f is a function that maps elements of a set T (the set of list elements) to elements of a set T'.
- If ℓ can be written as the concatenation of two lists, say $\ell = concat(\ell_1, \ell_2)$, then $src[e, f, g](\ell) = g(src[e, f, g](\ell_1), src[e, f, g](\ell_2))$, i.e. apply structural recursion to both sublists separately, then apply the operation $g : T' \times T' \to T'$ to the resulting pair.

(i) Discuss the conditions, under which $src[e, f, g]$ is well-defined—only consider the operation, if it is well-defined.

(ii) Show how to use structural recursion to define operations on lists such as

- determining the length (if not stored),
- applying a function to all elements of a list, and
- creating a sublist of list elements satisfying a condition φ.

(iii) Analyse the time complexity of structural recursion.

(iv) Implement structural recursion on either ALIST or DLIST.

total points: 17

Exercise 3. Consider a sequence data structure, where we are interested in just the following four operations:

- *pushback*(ℓ, x) appends the element x at the end of the sequence;
- *pushfront*(ℓ, x) adds the element x at the front of the sequence;
- *popback*(ℓ) removes the last element x of the sequence and returns it;
- *popfront*(ℓ) removes the first element x of the sequence and returns it.

(i) Show how these operations can be expressed by the operations studied in the lectures and labs.

(ii) Provide a different implementation of a class with these methods in **C++** with improved (amortised) complexity.

total points: 10

Exercise 4.

(i) Implement a function *selectionSort* on the class AList or DList for the selection sort algorithm.

(ii) Implement a function *insertionSort* on the the other of these two classes for the insertion sort algorithm. Use binary search in the inner loop of the method to minimise the number of comparisons.

(iii) Recall the definition of the bubble sort algorithm. In a nutshell, as long as there are list elements $list(i) \geq list(j)$ with $i < j$, swap them until the list is ordered. Implement the bubble sort algorithm by a function *bubbleSort* on both classes AList and DList.

total points: 18