Lab 4 – Exercises

EXERCISE 1. Consider the provided folder `AList(bsortqsort) template`, which contains an implementation of bubble sort and an incomplete implementation of quicksort on lists represented as unbounded arrays.

(i) Implement the missing member functions for quicksort. You need to rename `alistq_template.cpp`.

The completed implementation is provided in the folder `AList(bsortqsort)` after the lab.

(ii) Test your implementation by using either the provided program `sortlistmgt.cpp` or by creating your own test program.

EXERCISE 2. Consider the provided folder `DList(mergesort) template`, which contains an implementation of bubble sort and incomplete in-place implementation of mergesort on doubly linked lists.

(i) Implement the missing member functions for mergesort. You need to rename `dlistsort_template.cpp`.

The completed implementation is provided in the folder `DList(mergesort)` after the lab.

(ii) Test your implementation by using either the provided program `sortdlistmgt.cpp` or by creating your own test program.

EXERCISE 3. Consider the provided folder `AList(radixsort) template`, which contains an incomplete implementation of radixsort on lists represented as unbounded arrays.

(i) Implement the missing member functions for radixsort. You need to rename `alistradix_template.cpp`.

The completed implementation is provided in the folder `AList(radixsort)` after the lab.

(ii) Test your implementation by using either the provided program `rsortlistmgt.cpp` or by creating your own test program.

EXERCISE 4. Assignment 3 includes the implementation of rotation and selection operations on lists using divide-and-conquer algorithms.

(i) Discuss how to approach these implementations on unbounded arrays using `AList`.

(ii) Discuss how to approach these implementations on linked lists using `DList`.