**Exercise 1.**

Execute the animation (<http://www.yorku.ca/sychen/research/sorting/quicksortApplet.html>)

several times until you understand the principle behind the algorithm.

Hint: Use ”Step” to show clarifying text.

Hint2: The animation may be blocked by your security system. The blocking can be disabled as follows:

* Start > Control Panel > Java icon > Security > Edit Site List … > Add > Enter URL > OK

**Exercise 2**

a)

Implement the quicksort algorithm in Java. Use the pseudo code on WIKIPEDIA as inspiration (<http://en.wikipedia.org/wiki/Quicksort>).

Hint 1: Choose the element in the middle as the pivot element.

Hint 2: Test with an array of arbitrary numbers (integers).

b)

Measure actual execution times for the sorting implementation.

Hint 1: long time = System.nanoTime() returns a value representing the current time

Hint 2: Perform measurements on several arrays while varying their size.

Hint 3: Compare with SelectionSort. Download from Fronter as a java-file.

c)

Produce a worst case for Quicksort by sorting an (already) sorted array and choosing the first element in the array as the pivot element.

How big an impact does it have on the execution time?

d)

Let the algorithm choose a ”sensible” pivot element (eg: in the middle).

Measure execution times for the sort() method in the Arrays class and compare Quicksort for

* unsorted arrays
* sorted arrays

e)

What algorithm is actually used in the sort() method in the Arrays class?

Hint 1: Look up the answer in the documentation for API (read the doc!)

Hint 2: The sort() method is heavily overloaded. Make a small check for whether the library uses the same algorithm in (read the doc!)

* sort(int[])
* sort(Object[])

**Exercise 3**

See the Merge Sort - animation on <http://en.wikipedia.org/wiki/Merge_sort> several times until you understand the principle behind the algorithm.

Hint: Read the section ”Algorithm”.

**Exercise 4.**

a)

Implement the MergeSort algorithm in Java from the pseudo code (”Top-down implementation using lists”) on WIKIPEDIA (<http://en.wikipedia.org/wiki/Merge_sort>)

Hint: We need a non recursive "start method":

public void merge\_sort\_start (int[] arr)

{

int[] arr\_sorted = merge\_sort\_start(arr);

// Copy elements back to arr

for (int i = 0; i < arr\_sorted.length; i++) {

arr[i] = arr\_sorted[i];

}

}