LIITE A

```
quicksort(array:Array,
          left:integer, right:integer)

  if |left-right| < 3
    järjestä millä tahansa järjestämismenetelmällä
  else
    split = partition(array,left,right)

    if (split-1)-left > 0
      quicksort(array, left, split-1)
    if right-(split+1) > 0
      quicksort(array, split+1, right)

  endif
end

partition(array:Array, left:integer,
          right:integer) : integer

  leftCursor  = left
  %pivot-alkio on piilossa oikealla
  rightCursor = right - 1
  pivotValue  = array[right]

  do
    while (array[leftCursor] < pivotValue)
      leftCursor = leftCursor + 1

    while (rightCursor >= 0 AND
           array[rightCursor] >= pivotValue)
      rightCursor = rightCursor - 1

    if (leftCursor < rightCursor)
      vaihda taulukon array elementit keskenään kohdissa
          leftCursor, rightCursor

  while (leftCursor < rightCursor)

  vaihda taulukon array elementit keskenään kohdissa
          leftCursor, right

 palaa (return) funktiosta paluuarvona:leftCursor
```

Helsinki University of Technology
Department of Computer Science and Engineering
T-106.1220/1223 Data Structures and Algorithms T/Y

EXAMINATION
10.8.2009

Write on each paper your name, student number *with the control letter*, degree programme, and the course code and name. Also write the date, hall, the number of papers you return, and your *signature*.

1) Ten Questions (10 x 1p)

This is a compulsory part of the final exam. You need to get at least 5p out of the maximum 10p so that the rest of the exam will be checked. However, this part alone is not enough to pass the whole exam. On the other hand, in order to get 5p, you are not required to give "the exactly correct answer", but more or less show that you have understood the functionality of the code fragments related to this part. Thus, pay attention to the reasoning. Refer to the code line numbers if possible.

In Appendix A, you can see the Quicksort algorithm (it's the same as in the corresponding TRAKLA2 exercise) that sorts the given array in ascending order. Read through all the questions below without answering them and after that familiarize yourself with the code throughout. After this, answer all the questions and take time for pondering and explaining your reasoning. Note, however, that all the questions refer to the algorithm in Appendix A. In addition, the claims in the questions can be justified to be either true or false, thus the *argumentation* is the only thing that matters for the points!

a) Let N denote the *input size* of the algorithm when it is called with an unsorted NrrNy. Define N in terms of the parameters Nh t and Nh t t.
b) What does the algorithm do in lines 26-27? How about in lines 29-31?
c) The algorithm is called with an array [2, 1, 9, 0, 5, 2, 3]. What is the order of the items after the line 7 is executed? Give also intermediate phases.
d) What does the algorithm do in lines 10 and 12?
e) *Argue* whether it is true or false: the input and output for the algorithm is the one and the same NrrNy.
f) *Argue* whether it is true or false: if the size of the array is strictly less than 3, it is not worth to be sorted by this quicksort algorithm.
g) *Argue* whether it is true or false: this algorithm runs in "NNNbg*l*N" time.
h) Is this algorithm *stable*? Why?
i) *Anayse* the running time of the algorithm in case it is called with an array that is initially in *descending order*.
j) *Anayse* the running time of the algorithm in case it is called with an array that contains only equal keys.

Bonus exercise (do only if you still have time after the questions 2-4):

k) *How many times* the algorithm calls the subroutine NNrtrtron (if the size of the input is N)? Order of magnitude with argumentation suffice for an answer. Give the order of magnitude in *Big O notation*.

2) Terminology (2p + 2p + 2p + 2p)

Define the following *concepts* (4 x 1p). In addition, *give an example* of each (4 x 1p).

a) Abstract Data Type (ADT)
b) Priority queue
c) Binary heap
d) Heap-order property

3) Binary Search Trees (1p + 1p + 3p + 3p)

a) *Define* the concept Binary Search Tree.
b) Draw an example of a complete binary search tree of height 4.
c) What kind of binary search trees you know? How do they differ from each other? *Deal with at least three different data structures.*
d) *Compare* the time complexities (best case, average case, and worst case) of the data structures in (3c) according to the insert operation in terms of Big Oh Notation. Draw a matrix in which you have the time complexities as columns and the data structures as rows.

4) Determining Minimun Spanning Tree (4p + 4p)

a) Explain *an algorithm* that computes the minimum spanning tree for a graph. Argue what is the time complexity of this algorithm, if the graph has V vertices and E edges?

b) Lets consider the following undirected graph that has the vertices A-F and edges AB(2), AD(2), AE(6), BC(4), BE(3), CE(1), CF(4), DE(1), EF(3). The weight of an edge follows in the parenthesis. Show how the algorithm you just explained finds the mininum spanning tree for the given graph.

APPENDIX A

```
quicksort(array:Array,
          left:integer, right:integer)

  if |left-right| < 3
    sort by any method
  else
    split = partition(array,left,right)

    if (split-1)-left > 0
      quicksort(array, left, split-1)
    if right-(split+1) > 0
      quicksort(array, split+1, right)

  endif
end

partition(array:Array, left:integer,
          right:integer) : integer

  leftCursor  = left
  %Pivot is hidden in the right
  rightCursor = right - 1
  pivotValue  = array[right]

  do
    while (array[leftCursor] < pivotValue)
      leftCursor = leftCursor + 1

    while (rightCursor >= left AND
           array[rightCursor] >= pivotValue)
      rightCursor = rightCursor - 1

    if (leftCursor < rightCursor)
      swap array elements at
          leftCursor, rightCursor

  while (leftCursor < rightCursor)

  swap array elements at
          leftCursor, right

 return from function with value:leftCursor
```