Exercise 1. (7+7+7 points)
This exercise is concerned with sorting.
Consider the pseudo code for insertion sort:

\begin{verbatim}
Algorithm insertionSort(A, n):
  for j := 2 to n do
    key := A[j]
    i := j - 1
    while i ≥ 1 and A[i] > key do
      A[i + 1] := A[i]
      i := i - 1
      A[i + 1] := key
\end{verbatim}

(a) Apply insertion sort to \([4, 5, 2, 3, 1]\).
Clearly indicate what happens in each iteration of the for-loop; you may use arrows or other pictures if this adds to understanding.

(b) What is the best-case time complexity of insertion sort in terms of \(\Theta\)?
Explain your answer informally.

(c) Give the worst-case time complexity in terms of \(O\) of insertion sort, selection sort, merge sort, quicksort, heap sort (in that order).
You do not have to motivate your answer.

Exercise 2. (8+8 points)
This exercise is concerned with merge sort.

(a) Explain informally the working of merge sort; mention an important drawback.

(b) Give a recurrence equation describing the time complexity of merge sort.
Explain for every term in the recurrence equation why it is there.
Exercise 3.  (7 + 7 points)

(a) Give pseudo code for an otherwise unspecified procedure that takes as input a natural number \( n \), gives no output, and runs in \( \Theta(n^2) \).

(b) What is the worst-case time complexity in terms of \( \mathcal{O} \) of searching in a hash table containing \( n \) keys, where collision is solved by chaining?

Exercise 4.  (8+8 points)
This exercise is concerned with linear data structures.

(a) Implement a stack using two queues; each queue has operations enqueue and dequeue.
   Give and shortly explain the worst-case time complexity of your procedures pop and push in terms of \( \mathcal{O} \).

(b) Implement a stack using a singly linked list without sentinel. For a node \( x \) we have \( x.next \) and \( x.key \) available, and for the list \( L \) we have \( L.head \) available.
   Give and shortly explain the worst-case time complexity of your procedures pop and push in terms of \( \mathcal{O} \).

Exercise 5.  (8+7+8 points)
This exercise is concerned with tree-shape data structures.

(a) What is the height of a (max-)heap with \( n \) elements?

(b) Consider the max-heap given by the array \([5, 4, 3, 2, 1]\).
   Depict the removal of 5 from the max-heap. Draw every step.

(c) Add the nodes with keys \( 3, 5, 4, 1, 2 \) in that order one by one to an initially empty binary search tree. Draw every step.

The mark for the midterm is (the total number of points plus 10) divided by 10.