Problem sheet 2

Course 320201 Fundamental Computer Science I, Dr. Holger Kenn
e-mail: h.kenn@iu-bremen.de, tel.:+49 421 200 3112

This problemsheet's solution is to be handed in Friday, September 19th *before the lecture*, either clearly readable on paper or as a *PDF* file via e-mail to h.kenn@iu-bremen.de.

1.) Dangers of asymptotic notation
Find the error in the following proof of the (false) theorem $2^n = O(1)$

Proof by induction: The theorem is true for $n = 1$ since $2 = O(1)$. Let's now assume that $2^{n-1} = O(1)$. We have to show that $2^n = O(1)$ but this is obvious since

$$2^n = 2 \cdot 2^{n-1} = O(2^{n-1}) = O(1).$$

(2p)

2.) Removing multiple occurences
Sorting is often only used to eliminate multiple occurences. Let's assume that there are n elements $x_1, \ldots, x_n \in U$ given and that we want to remove all but one occurence of each element from the input.

2.1) Show that if there is a total order on U (that can be evaluated in constant time for two elements) the problem can be solved in $O(n \lg n)$ time and write down an algorithm that solves the problem. (Let's assume that there is a PRINT function that prints something to the output. For information about total orders, see page 1077 in Cormen/Leiserson/Rivest/Stoll)

(2p)

2.2) Let's now assume that there is no total order but just an equality test on U, e.g. a comparision function for photos. Show that it is necessary to do $\binom{n}{2}$ pairwise comparisions to solve the problem and write down an algorithm that solves the problem.

(3p)

2.3) Finally, let's now assume that $x_1, \ldots, x_n$ are integers $\in \mathbb{Z}$. Show that the problem can now be solved in $O(n)$ time. You can assume an infinite ammount of memory for this but no pre-initialisation of that memory. Again, write down an algorithm that solves the problem.

(3p)