Algorithms. Exercises from Week 1

1. For very good reasons, integers are written in a positional numeral system with a base $b \geq 2$. Usually we take the decimal system with $b = 10$. The binary system has base $b = 2$. But in principle every fixed b may be used. Question: Do the time bounds of arithmetic operations (addition, multiplication, etc.) in O-notation depend on b? Explain your yes- or no-answer; give precise arguments.

2. Suppose you have an algorithm with the following two parts: The first part consists of two nested for-loops where the loop indices run through at most n different values (such as: "for $i = 1$ to n do for $j = i$ to n do [some elementary operation with i and j] endfor endfor"). The second part is just one such for-loop with at most $3n$ values. What would be the time complexity of the entire algorithm in O-notation? Explain in detail.

3. Suppose you have three algorithms (for whatever problem) that need $O(n^2)$, $O(2^n)$, and $O(n!)$ time, respectively. Discuss: When the speed of your machine is doubled, how does this affect the sizes of inputs you can handle within a fixed time budget?

4. Let x and y be integers with n digits. We consider operations with digits as elementary operations. What are the worst-case time complexities of the following calculations: computing $x + 1$, computing $x - 1$, comparing x and y? (Comparing means to decide whether $x < y$, $x = y$, or $x > y$.)

5. For a given n, how many digit operations do you need to compute the sum $1 + 2 + 3 + \cdots + n$? Try to come up with a bound in O-notation which is as good as possible. But first think twice how you compute the requested sum. (Once in 1785 the eight-years old Carl Friedrich Gauß surprised his teacher by solving this problem by what we would call a fast algorithm today ...)