# Design and Analysis of Algorithms

Exercises 2/8

1. Derive time complexity function for the following algorithm.

```
CleverExample( x: INTEGER );

  IF x ≤ 1 THEN RETURN 1;
  ELSE
    i := 12;
    WHILE i >1 DO
       CleverExample( x-12 );
       i := i-1;
    i := 12;
    WHILE i >1 DO
       CleverExample( x / 2 );
       i := i-1;
```

2. Analyze the time complexity of *Quicksort* using substitution method for the best-case behavior assuming partition always provides equal size subsets. Derive time complexity also for worst case. Show counterexample how this worst-case behavior happens. What is the probability of the worst case happening?

3. Implement ***partitioning* algorithm** (used in Quicksort and in Selection problems) in any programming language. Test it for randomly generated input and report output times for problems of size $N$=100-100 000. How large problems can your program solve in 1 second? You do not need to implement Quicksort. Only consider the partition step that is used in Quicksort.

4. Divide-and-conquer algorithms divide the problem usually into two equal halves. What happens if we divide into $k$ equal size sub-problems so that $b$=$c$? Derive time complexity for the special cases (a) $k$=8, (b) $k$=$N$/2.

5. Peasant multiplication algorithm works as follows. Given two numbers, say $a$=64 and $b$=64, multiplied the first one by 2, and divide the second one by 2 until it reaches 1. Maintain book-keepings of the reminders, and finally, sum them up.

Example 1: ($a$,$b$)→(64,64)→(128,32)→(256,16)→(512,8)→(1024,4)→(2048,2)→(<u>4096</u>,1) = **4096**.
Example 2: ($a$,$b$)→(<u>3</u>,11)→(<u>6</u>,5)→(12,2)→(<u>24</u>,1) = **33**.

What is the time complexity of this algorithm?