Data Structures and Algorithms **Spring-Summer 2012/13**

Assignment 1

Valeria Fionda, Mouna Kacimi,
Werner Nutt, Simon Razniewski

Recursion and Complexity of Algorithms

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.
You must be prepared to present your solution at the lab. If you are not able to explain your solution, this will be considered as if you had not done your work at all.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Experience shows that Word is in general difficult to use for this kind of task.
For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the Java code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries ("import") if not stated otherwise.
Please, include name and email address in your submission.

1. Recursion

A palindrome is a phrase that reads the same forward and backward (examples: 'racecar', 'radar', 'noon', or 'rats live on no evil star'). By extension we call every string a palindrome that reads the same from left to right and from right to left.
Develop a recursive algorithm that takes as input a string and decides whether the string is a palindrome.
Implement your algorithm in Java. Provide the argument for the test run in a global variable.

(6 Points)

2. Comparison of Running Times

For each function $f(n)$ and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds.

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							

Hint: Some values are easy to calculate. For others, binary search may be a good idea, either by hand or using a program.

(10 Points)

3. Array of Averages

Design an algorithm that achieves the following task: Given an array $A[1..n]$ of floating point numbers, it returns a two-dimensional array, say M, of size $n \times n$ in which the entry $M[i][j]$ for $i \leq j$ contains the *average* of the array entries $A[i]$ through $A[j]$. That is: if $i \leq j$, then

$$M[i][j] = \frac{A[i] + \cdots + A[j]}{j - i + 1},$$

whereas for $i > j$ we have that $M[i][j] = 0$.

1. Describe the algorithm that creates this matrix in pseudocode.
2. What is the running time of your algorithm with respect to the variable n? Give an upper bound of the form $O(f(n))$ and a lower bound of the form $\Omega(g(n))$. Can you characterize the asymptotic running time by by some $\Theta(h(n))$?
3. Implement the algorithm in Java. Provide the argument for the test run in a global variable.

4. Measure the run time of your Java program for random inputs of size $n = 10, 100, 1000$, etc. Does the growth of the running time correspond to your asymptotic estimates?

   (14 Points)

Submission: Until Wed, 13 March 2012, 11:59 pm, to

```
dsa-submissions AT inf DOT unibz DOT it.
```

If you want to submit a hand-written solution, scan it and send it to the email address above.