KAIST School of Computing 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
Tel. +82-42-350-3502~3507 Fax. +82-42-350-3510

Prof. Dr. Martin Ziegler
+82-42-350-3568
ziegler@kaist.ac.kr
http://m.zie.de

CS500 Jan 13, 2017 PhD Qualifying Exam

70 points = 140%

Assignment 1 (6+5+4+5 points):

a) Specify (!) and describe three *significantly* different algorithms for sorting n given keys, together with their asymptotic computational worst-case costs. (No proofs required.)

b) Set up and justify a recurrence for the number $T(n)$ of steps performed by the recursive sorting algorithm **stoogeSort** shown below when called with **left**=0 and **right**=n-1.

```
1  procedure stoogeSort(int array[], int left, int right)
2  if array[left]>array[right] then swap(array[left],array[right]) fi
3  if (right - left + 1) < 3 then return fi
4  int third := ( right – left + 1 ) / 3     // rounding down
5  stoogeSort(array, left, right-third)
6  stoogeSort(array, left+third, right)
7  stoogeSort(array, left, right-third)
8  endproc
```

c) Let **array[]=(3,6,5,2,1,4)**, initially, and consider the call **stoogeSort(array,0,5)**. Write the contents of the array and variables after execution of (i) line #4, (ii) line #5, (iii) line #6, and (iv) line #7. (Do not expand the recursive calls themselves, though; instead peruse the fact that **stoogeSort** correctly sorts arrays of size up to 4...)

d) Prove that the asymptotic growth of any non-decreasing f:$[1;\infty)\to[1;\infty)$ satisfying $f(n)=b\cdot f(n/a)$ for all $n\geq a$ is $f(n)=\Theta(n^{\ln(b)/\ln(a)})$, if $1<a<b$ are fixed.

Reminder from calculus: $a^x = e^{x\cdot\ln(a)}$, $\log_y(a)=\ln(a)/\ln(y)$, $\ln(3)/\ln(1.5)\approx 2.71$

Assignment 2 (5+5 points):

a) Explain in few sentences, and give examples demonstrating, the differences between (i) a program/code, (ii) an algorithm, and (iii) a heuristic.

b) Briefly explain the difference between (i) cost of an algorithm and (ii) computational complexity of a problem, for instance by comparing Assignments 1a) and 1b).

KAIST School of Computing 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
Tel. +82-42-350-3502~3507 Fax. +82-42-350-3510
Prof. Dr. Martin Ziegler

Problem 3 (4+5+6+4+6 points):

a) What is the *worst-case* cost (=number of bit flips) of once incrementing a binary counter containing any integer between 0 and n-1, asymptotically as $n\to\infty$?
Justify your answer by (i) exhibiting an example where that many bit flips do occur and (ii) by proving that more bit flips cannot occur.

b) Analyze the *amortized* cost of a binary counter with operation **INC**.
That is, when counting in binary from 0 to n, determine the total number of bit flips, divided by n asymptotically as $n\to\infty$. Again, justify your answer!

c) Now analyze the *amortized* cost of a binary counter with both operations **INC** and **DEC**.
That is, determine the total number of bit flips, divided by n, incurred in the worst case by any combination of n calls to **INC** and/or **DEC** asymptotically as $n\to\infty$. Justify!
(Initially the counter contains zero, and decrementing zero returns zero again...)

d) Analyze the *worst-case* cost of the following algorithm, asymptotically as $n\to\infty$.
Given an n-tuple $(b_1\ldots b_n)$ of bits, scan for the (index j of the) first bit that is non-zero; in case all b_j are zero, count to 2^n-1 and stop.

e) Now analyze its asymptotic *average* cost. Justify your answers!

Problem 4 (5+5+5 points): Suppose $\mathcal{A}$ is a randomized algorithm solving the decision problem L in time $t(n)$ with *one*-sided error $\frac{1}{2}$ independently of n: On inputs $\underline{x}\notin L$, $\mathcal{A}$ always correctly reports **false**; but on inputs $\underline{x}\in L$, $\mathcal{A}$ might also report **false** with probability $\frac{1}{2}$.

a) Design and analyze an algorithm $\mathcal{A}'$ that, by repeating $\mathcal{A}$ an appropriate (which?) number N of times, errs with probability $\leq 2^{-100-n}$, where n denotes the (known) length of x.

b) Analyze the (i) *worst* and (ii) *expected* cost of the following randomized 'algorithm':
Flip a fair coin. If it comes out ***heads***, stop; otherwise repeat.

c) Prove $\sum_n n\cdot p^n = p/(1-p)^2$ for all $|p|<1$. Hint: Cancel one p and compare anti-derivatives.