Program Design and Data Structures, 20.0 c

Course code: 1DL201, Report code: DL201, 67%, DAG, NML,
week: 44 - 02 Semester: Autumn 2017
week: 03 - 11 Semester: Spring 2018

LAB ASSIGNMENT 7

This information is not available in English. Now showing the Swedish version.

AFTER THIS LAB YOU SHOULD BE ABLE TO...

- Calculate the run-time recurrence for some Haskell functions.
- Determine the closed form of a recurrence using the substitution and the expansion method.
- Use The Doctor Theorem to determine the closed form of recurrences.

BEFORE THE ASSIGNMENT

- Read the slides for lectures 11 and 12.
- Read the *whole* assignment first.

INSTRUCTIONS

- This assignment exercises the expansion and substitution methods, and mathematical induction.
- Answer the following questions on paper. For each of your answers, please show sufficient detail to demonstrate that you know how to use each method. Show your TA a clearly presented version of your working.
- Challenge problems are **not** part of the assessment. Attempt these when you have finished the other questions. You are advised to try these anyway.
- This assignment exercises the expansion and substitution methods, and mathematical induction.

THE TASK

Problem 1

Determine the recurrence giving the run-time cost of the following functions:

a)

```
foo [] = []
foo [a] = []
foo (a : b : as) = a : foo as
```

b)

```
bar [] = 0
bar (a : as) = bar as + bar as
```

c) (**Challenging**)

```
zap [] = []
zap l @ (a : as) = bar l : zap as
```

Problem 2 (**Challenging**)

Determine the recurrence giving the run-time cost of the following function:

```
take 0 l = []
take n [] = []
take n (a:as) = a : take (n - 1) as
```

How does the result differ from the answers in the previous question?

Problem 3

Apply **both** the expansion method and the substitution method to the following recurrences to derive a closed form.
Prove using induction that your resulting closed form formula equals the recurrence.

a) $N(n) = \begin{cases} 1 & \text{if } n = 0 \\ N(n-1) + n & \text{if } n > 0 \end{cases}$

Hint: When performing the calculation avoid simplifying terms. This will allow the pattern to be more easily seen.

b) (**Challenging**) $M(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ M(n-2) + 1 & \text{if } n > 1 \end{cases}$

Hints:

- Prove $M(2n) = M(2n+1)$. How can this help?
- For the substitution method, consider cases where $n = 2k$ and $n = 2k + 1$separately.

WHEN YOU ARE DONE

When you are done with all problems (the challenges, and the explorations below, are optional), raise your

hand or approach an assistant to have your solution graded.

If you pass this lab at least 30 minutes early and other groups are still working on it, we ask you to **help one other group**. Do not simply share your solution with them, but try to understand the (partial) solutions that they have developed so far (which may be different from yours) and the difficulties that they have. Assist them in coming up with their own solutions. Once the group that you are helping completes the lab assignment, you may leave. (If it is still early, the group that received your help should then stay to help another group.)

EXPLORATIONS

If you have finished the lab exercises and are waiting to be graded, or if you wish to explore programming in Haskell further at any time, or want to get some extra practice, have a look at the Explorations page.

Exploration problems are optional and should only be attempted after other problems are completed.
You don't have to show these answers to the lab assistants for grading.