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 1

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

After this lab session you should be able to...

- Start the Haskell interpreter.
- Write simple Haskell-expressions and evaluate them.
- Understand and explain how simple expressions are computed and what their types are.
- Understand and explain what basic built-in functions in Haskell do.

Instructions

1. Start the Haskell interpreter with the command ghci in the Unix shell.
2. Take notes while you work, to make it easier to show what you have done, and to answer questions from the assistant who grades your lab assignment.

The Task

PROBLEM 1

This problem gets you familiar with using Haskell as a calculator, by playing around with some built-in functions and operators. If you do not recognise them from the lectures or don't remember what they mean, that's ok. If you can't guess what they mean simply based on our suggested computations below, we encourage you to try the same thing but with other values, and see if that helps. If not, have a look in the course literature or the slides for the first few lectures.

Go through the following expressions, one at a time.

- First, think about what you believe the Haskell interpreter will compute -- in other words, what type and value the expression has (or if it has a type and value at all).
- Input the expression into the Haskell interpreter. Which reply did you get? Was it what you thought it would be? If not -- think about why! If you do not understand why you obtain a certain reply, ask a lab assistant to explain.
- Check the type of the expression with :type - for instance, :type 1477 (or `:t 1477`)

- Write down the answer -- both type and value, or the error message if you obtained one.

1. 1477
2. 3 + 2
3. 15 / 2
4. div 15 2
5. 15 `div` 2
6. 15 div 2
7. 15.0 / 2.0
8. 15.0 `div` 2.0
9. 13.4 / 1.8
10. 2.71 > 5.67
11. "x" ++ "yz"
12. x ++ "yz"
13. show 7
14. read "7" :: Integer
15. truncate 23.6
16. take 4 "Computer Science!"
17. drop 4 "Computer Science!"
18. take 7 drop 9 "Computer Science!"
19. take 7 (drop 9 "Computer Science!")
20. "kalle" >= "andersson"
21. "anders" < "andersson"
22. length "soffbord"
23. 2 >= 6 || 2.71 > 5.67
24. 2 >= 6 || 2.71 < 5.67
25. 65 < 150 || 19
26. 1 `div` 0 < 3 && 2.0 < 1.0
27. 2.0 < 1.0 && 1 `div` 0 < 3
28. not (2.71 > 5.67)

PROBLEM 2

Which computations do the functions listed below perform? Use the results from Problem 1 or try on your own with other examples! Write down the answers. If a function can be used with different types of arguments (eg. >), how does it work for each of the types? (E.g., how does > work on integers? Floating-point numbers? Strings?)

1. /
2. div
3. >
4. >=
5. ||
6. not
7. ++
8. take
9. drop
10. truncate
11. length

Find a description of the last four functions in the Haskell Documentation https://www.haskell.org/hoogle/.

Haskell documentation is a crucial resource, not only for finding information about known functions, but also for finding other useful functions to help you solve programming problems. Use it freely!

PROBLEM 3

Try the following expressions in Haskell.

1. round 3.4
2. (round)3.4
3. round3.4
4. round 3.4 + 10.0
5. round(3.4 + 10.0)
6. 1 + 2 * 3
7. (1 + 2) * 3
8. 3 * 2 + 1
9. (3 * 2) + 1

10. 7+6
11. 7 + 6
12. 17 `div` 3
13. div (17 3)
14. not (not True)
15. (not not) True

Questions:

1. How can one see which parts of the expressions become arguments to the functions?
2. How do parentheses impact the interpretation of expressions?
3. How does whitespace impact the interpretation of expressions?

PROBLEM 4

What does the Haskell interpreter reply when you compute the following expressions and declarations in the given order? What do the replies mean (if indeed the Haskell interpreter does reply)? Speculate what happens when the Haskell interpreter does not reply with a value.

1. let pi = 3.1415927
2. let radius = 5.0
3. let area = pi * radius * radius
4. area

WHEN YOU ARE DONE

When you are done with all problems (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.