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 13

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

AFTER THIS LAB YOU SHOULD BE ABLE TO ...

- write type class definitions
- write instance definition for various types

BEFORE THE LAB:

- Read the slides of lectures 20
- Read the whole assignment first.

INSTRUCTIONS:

- Define a type class that has a function that computes and approximation the size of elements of a data type in bytes.
- Define instances of that type class for various data types.
- Apply the definition to various values.

TASKS

The goal of this exercise is to write a type class (Size) that has a single function (size) that computes an approximation of the number of bytes in memory an element of a given Haskell data type takes up (use arbitrary precision integers `Integer` to cope with big data). Recall that a byte is 8 bits.

The basic rules are:

- A value of type `Int` take 4 bytes (we are assuming a 32 bit architecture)
- A value of type `Integer` (big integer) takes precisely the number of bytes required to store the integer + 1 byte to record how many bytes are consumed. The number of bytes required to store the integer `n` is ceiling (logBase 2 (fromInteger n) / 8.0)
- `Chars take 2 bytes`
- `()` takes 1 byte --- nothing takes less space than a byte

- `Bool`eans take 1 byte.
- Values of types declared using `data` construct require whatever space is required to store the data of the data type, plus 1 additional byte to indicate which branch of the data type is being considered, that is, to store the constructor. For example, given `data G = O | G Int`, the value `O` takes 1 byte and the value `G 10` takes 1 + 4 = 5 bytes.
  Note that *lists* fall into this category. A list can be seen as datatype data [a] = [] | (:) a [a], where [] and (:) are special constructor names.
- A `newtype` takes precisely the amount of memory required to store values of the type being tagged no space is required to store theconstructor.
- Pairs, triples etc take precisely the space required to store the elements of the tuple.
- The size of functions cannot be computed.

PROBLEM 1

Write a type class declaration for the type class `Size` that has a single function size, which, when applied to a value of a type in the type class will give the size of that value as an `Integer`.

PROBLEM 2

Write instances of the `Size` type class for the following types:

- `Int`
- `Integer`
- `Bool`
- `Char`
- `()`
- `[a]`
- `newtype F a = F (Int, a)`
- `data Tree a = Leaf a | Branch (Tree a) (Tree a)`
- `(a,b)`
- `(a,b,c)`

PROBLEM 3

Calculate the size of

- `10 :: Int`
- `4 ^ 200 :: Integer`
- `((), ())`

- `"World"`
- `F (10, 4 ^ 200) :: F Integer`
- `([1,2,3], "Hello World", Branch (Leaf (1, "Hello")) (Leaf (10000, "World"))) :: ([Int], String, Tree (Int, String))`

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.