COMP313-08A Programming Languages

First Haskell coursework

The first few questions of this coursework are intended to get you started using the hugs interpreter with some simple definitions. There are some larger problems later on in the coursework. I strongly suggest that you read through my "background" notes on Haskell, accessible from the course web site as http://www.cs.waikato.ac.nz/~stever/Haskell notes.pdf, since some of the questions refer to ideas in those notes which we may not yet have covered in the lectures.

1 What is the type of `(2 *)`?
2 What value (remember that functions are values) does `(2 *)` have?

3. Create a file called ex1 and type in the following definitions:

```
twice f x = f (f x)
square n = n * n
naturals = [0..]
ones = 1 : ones
```

and then load this file using the load command. Once the file is loaded, try to answer the following questions:

a. What is the type of `twice`?
b. What is the value of `twice not True`?
c. what is the type of `square`?
d. what is the type of `twice square`?
e. what is the value of `twice square 4`?
f. what is the type of the pre-defined function `head`?
g. what is the type of `head naturals`?
h. what is the value of `head naturals`?
i. draw a picture (using boxes for elements and arrows to show ordering) of the list `naturals`.
j. do the same for the list `ones`.

4. What expression, using only the function `square` and the number `2`, has the value `16`?

5. What expression, using only the list `naturals`, the function `(2 *)` and the function `map`, represents the list of all even natural numbers?

6. Define a function

```
capitalize :: Char -> Char
```

which given any alphabetic character returns its capital form. On numbers and punctuation it should leave the character alone.

You will find the definitions

```
ord :: Char -> Int
ord = fromEnum

chr :: Int -> Char
chr = toEnum
```

useful. So,

```
capitalize ’a’ = ’A’
capitalize ’e’ = ’E’
capitalize ’D’ = ’D’
capitalize ’{’ = ’{’
capitalize ’3’ = ’3’
```

7. Using only the pre-defined function `map` and the function `capitalize` you gave in question eight, define a function

```
capitalize_string :: String -> String
```

which takes a string and applies `capitalize` to each element. So,

```
capitalize_string “Hello, Steve” = “HELLO, STEVE”
capitalize_string “Happy Birthday on the 31st” =

                                   “HAPPY BIRTHDAY ON THE 31ST”
```

8. Using only `foldr`, the Boolean operation `(||)` and `False`, define a function

```
or_list :: [Bool] -> Bool
```

so that

```
or_list [b1, b2,...,bn] = b1 || b2 ||...|| bn
```

Note that

```
or_list [] = False
```

9.
i. Define, using only a list comprehension, the list `[1..k]` for some `k`, the string `“sheep\n“`and the pre-defined function `concat`, a function

```
flock :: Int -> String
```

which, when given a number, say five, as argument, is such that

```
putStr (flock 5)
```

gives

```
sheep
sheep
sheep
```

```
sheep
sheep
```

i.e. five sheep, one on each line, as value.

ii. Now define

```
flock2 :: Int -> String
```

which does the same thing but WITHOUT using a list comprehension, but using anything else you like.

iii. Using only a list comprehension, the string `"sheep "`, lists of natural numbers and the pre-defined function `concat`, define the function `a_row_of_sheep :: Int -> String`

so that `a_row_of_sheep 5`, for example, has value

```
sheep sheep sheep sheep sheep
```

iv. By considering your answers to i and iii and generalizing, define a function

```
big_flock :: Int -> String
```

so that `putStr (big_flock 5)`, for example, has value

```
sheep
sheep sheep
sheep sheep sheep
sheep sheep sheep sheep
sheep sheep sheep sheep sheep
```

10.
For this question do all of Exercise 2.5 on page 33 of the textbook.

Your answers for all the questions above are due at 1000 on Wednesday 26th March 2008.

You must submit your answers as a plain text file via Moodle. This **MUST** be a plain text file (**not** a PDF, **not** a MS-format file or any other sort of particular format) since we will want to load your solutions and try them out in hugs.