McMaster University
Department of Computing and Software
Dr. W. Kahl

SFWR ENG 3E03
Exercise Sheet 7

Design and Selection of Programming Languages

30 October 2006

Exercise 7.1 — Defining Haskell Functions

(a) Define the Haskell function *interleave* :: [*a*] → [*a*] → [*a*]

such that *interleave xs ys* evaluates to a list alternatingly containing elements of *xs* and *ys*.

E.g., *interleave* "1234" "abcdefg" = "1a2b3c4defg"

(b) Write a Haskell program Interleave.hs that accepts two file or three file names as command-line arguments, and writes the interleaving of the lines of the first two files into the third file, when given, or to standard output otherwise.

(c) Define the Haskell function *wc* :: *String* → (*Integer* , *Integer* , *Integer*)

such that *wc s* = (*charCount* , *wordCount* , *lineCount*), iff the three tuple members are, repectively, the number of characters, words, and lines in *s*.

Challenge: Calculating the three counts separately keeps the whole *s* in memory — this could occupy gigabytes. Write a version of *wc* that processes its argument string only once (typically via direct recursion).

(d) Write a Haskell program WordCount.hs that behaves like the unix utility `wc` (at least without flags), namely calculating and displaying the counts for all its arguments, e.g.:

```
SE3E03/2006 $ wc Sheet*.lt
   391   1728  10938 Sheet1.lt
  1277   4689  35751 Sheet2.lt
   455   1688  11497 Sheet3.lt
   646   2989  16739 Sheet4.lt
   477   2086  11730 Sheet5.lt
   536   2493  13229 Sheet6.lt
   348   1468   8542 Sheet7.lt
  4130  17141 108426 total
```

If no arguments are given, standard input is counted instead.

```
SE3E03/2006 $ wc < Sheet7.lt
 348 1468 8542
```