Functional Program Derivation
Exercises for Day 1

Shin-Cheng Mu Shu-Chun Weng (TA)

2007 Formosan Summer School on
Logic, Language, and Computation
July 4th, 2007

This exam sheet is worth 20 points in total.

1.1 The Expand/Reduce Transformation

1. **(1 point)** What does this function do?

$$\begin{array}{lcl} \mathit{descend}\,0 & = & [\,] \\ \mathit{descend}\,(n+1) & = & (n+1){:}\mathit{descend}\,n \end{array}$$

2. **(1 point)** Consider the definition $f = \mathit{sum} \cdot \mathit{descend}$

   (a) Describe in words what this function does.

   (b) Calculate $f\,0$.

   (c) Simplify $f\,(n+1) = \ldots f\,n \ldots$.

   (d) From (b) and (c), synthesise a recursive definition of f.

3. **(1 point)** Recall the datatype definition for internally labelled binary trees:

$$\mathit{data}\;\mathit{iTree}\,\alpha \;=\; \mathit{Null} \mid \mathit{Node}\,\alpha\,(\mathit{iTree}\,\alpha)\,(\mathit{iTree}\,\alpha).$$

   Consider the function *mapiTree* defined below:

$$\begin{array}{lcl} \mathit{mapiTree}\,f\;\mathit{Null} & = & \mathit{Null}, \\ \mathit{mapiTree}\,f\,(\mathit{Node}\;a\;t\;u) & = & \mathit{Node}\,(f\;a)\,(\mathit{mapiTree}\,f\;u)\,(\mathit{mapiTree}\,f\;t). \end{array}$$

   What does this function do?

4. **(1 point)** Define a function *sumiTree* computing the sum of all node values in an *iTree*.

5. **(2 points)** The function $one\, x = 1$ returns 1, what ever the input is. The function *sizeiTree*, computing the size of a tree, can be specified by:

$$sizeiTree \quad = \quad sumiTree \cdot mapiTree\ one.$$

Derive a definition of *sizeiTree* which does not construct an intermediate tree.

1.2 Proof by Induction

1. **(2 points)** Prove $(xs +\!\!+ ys) +\!\!+ zs = xs +\!\!+ (ys +\!\!+ zs)$. Hint: induction on xs.

2. **(2 points)** The function *concat* concatenates a list of lists:

$$\begin{array}{lcl} concat\,[\,] & = & [\,], \\ concat\,(xs : xss) & = & xs +\!\!+ concat\, xss. \end{array}$$

E.g. $concat\,[[1,2],[3,4],[5]] = [1,2,3,4,5]$. Prove that:

$$sum \cdot concat = sum \cdot map\, sum.$$

Hint: you may need one of the properties proved in the lecture.

3. **(2 points)** Prove that $map\, f \cdot map\, g = map\,(f \cdot g)$.

4. **(2 points)** The function *swapTree* is defined by:

$$\begin{array}{lcl} swapiTree\ Null & = & Null, \\ swapiTree\,(Node\ a\ t\ u) & = & Node\ a\,(swapiTree\ u)\,(swapiTree\ t). \end{array}$$

Prove that $swapiTree\,(swapiTree\ t) = t$ for all t.

1.3 Accumulating Parameters

1. **(3 points)** Recall the standard definition of factorial:

$$\begin{array}{lcl} fact\,0 & = & 1, \\ fact\,(n+1) & = & (n+1) \times fact\ n. \end{array}$$

This program also implicitly uses space linear to n in the call stack.

   (a) Introduce $factit\ n\ m = \ldots$ where m is an accumulating parameter.

   (b) Express *fact* in terms of *factit*.

   (c) Construct a space efficient implementation of *factit*.

2. **(3 points)** Given an *iTree*, the following function *flatten* returns a list of all labels in the tree, in left-to-right order:

$$\begin{array}{lcl} \mathit{flatten}\ \mathit{Null} & = & [\,], \\ \mathit{flatten}\ (\mathit{Node}\ a\ t\ u) & = & \mathit{flatten}\ t +\!\!+ [a] +\!\!+ \mathit{flatten}\ u. \end{array}$$

Unfortunately, *flatten* is slow. Let us try to improve it.

(a) Introduce $\mathit{flatcat}\ t\ xs = \ldots$ where xs is an accumulating parameter.

(b) Express *flatten* in terms of *flatcat*.

(c) Construct an efficient implementation of *flatten*. You will need some properties of $(+\!\!+)$ proved in one of the exercises.