QUESTION:

Due Thu Feb 15

Define a data type for trees that can have:

leaves, of size 0

unary nodes, of size 1 + size of child

binary nodes, of size 2 + size of children

ternary nodes, of size 3 + size of children

Write a Haskell program that generates all the trees of size n.

Write a Pyhton program that does the same.

Hints: use

http://www.cse.unt.edu/~tarau/teaching/GPL/FP/TreeGen.hs for

your Haskell code

and

http://www.cse.unt.edu/~tarau/teaching/GPL/FP/bin\_mot.py

for your Python code.

PYTHON PROGRAM :

# binary and binary-unary (Mozkin) tree generators

# binary tree of size n

def bin(n) :

if n==0 :

yield n

else :

for k in range(0,n) :

for l in bin(k) :

for r in bin(n-1-k) :

yield (l,r)

# Motzkin tre of size n

def mot (n) :

if n==0 :

yield n

else :

for m in mot(n-1) :

yield [m]

for k in range(0,n-1) :

for l in mot(k) :

for r in mot(n-2-k) :

yield (l,r)

def countFor(f,n) :

for i in range(n) :

count = 0

for t in f(i) :

count+=1

yield count

def countsFor(mes,f,n) :

print(mes)

print([c for c in countFor(f,n)])

print("")

def showFor(mes,f,n) :

print(mes)

for t in f(n) :

print(t)

print("")

showFor('Binary trees',bin,3)

showFor('Motzkin trees',mot,4)

countsFor('Binary trees',bin,12)

countsFor('Motzkin trees',mot,12)

print("done")

def test() :

for n in range(6) :

print(n,list(mot(n)))

HASKELL PROGRAM:

module TreeGen where

data M = Z | L M | A M M deriving (Eq,Read,Show)

-- motzkin numbers [1,1,2,4,9,21,51,127,323,835]

mot 0 = [Z]

mot n | n> 0 =

map L (mot (n-1)) ++

[A x y | k<-[0..n-2], x<- mot k, y<- mot (n-2-k)]

-- large Schroder numbers: [1,2,6,22,90,394,1806,8558,41586]

mot0 0 = [Z]

mot0 n | n> 0 =

[L x |x<-mot0 (n-1)] ++

[A x y |k<-[0..n-1], x<- mot0 k, y<- mot0 (n-1-k)]

-- Catalan numbers: 1, 1, 2, 5, 14, 42, 132, 429, 1430

bin 0 = [Z]

bin n | n>0 = [A x y|k<-[0..n-1], x<- bin k, y<- bin (n-1-k)]

-- counted by Catalan numbers with weight 2 for arity 2 nodes

bin1 0 = [Z]

bin1 n | n>0 = [A x y|k<-[0..n-2], x<- bin1 k, y<- bin1 (n-2-k)]

-- tests

bt = map (length.bin) [0..9]

bt1 = map (length.bin1) [0..9]

mt0 = map (length.mot0) [0..9]

mt = map (length.mot) [0..9]