

Madhuri Pyreddy
17 October 2019

CSCI 117 Lab 8

Part One:

a1. local X Y Generate Display DisplayH in

```
fun {Generate N}  
  fun {$} (N#{Generate (N+1)}) end  
end
```

```
proc {Display X N}  
  fun {DisplayH Z Num}  
    if (Num == 0) then nil  
    else  
      (V#F) = {Z} in  
      (V|{DisplayH F (Num-1)})  
    end  
  end  
  local L in  
    L = {DisplayH X N}  
    skip Browse L  
  end  
end
```

//Testing

```
X = {Generate 3}  
{Display X 5}
```

end

a2. local X Y Generate Display DisplayH Times in

```
fun {Generate N}  
  fun {$} (N#{Generate (N+1)}) end  
end
```

```
fun {Times X Y}  
  fun {$}  
    (V#F) = {X} in  
    ((V*Y)#{Times F Y})  
  end  
end
```

```

proc {Display X N}
  fun {DisplayH Z Num}
    if (Num == 0) then nil
    else
      (V#F) = {Z} in
      (V|{DisplayH F (Num-1)})
    end
  end
local L in
  L = {DisplayH X N}
  skip Browse L
end
end

```

```

//Testing
X = {Generate 3}
Y = {Times X 3}
{Display Y 5}
end

```

a3. local X Y Generate Display DisplayH Merge in

```

fun {Generate N}
  fun {$} (N#{Generate (N+1)}) end
end
Merge = fun {$ X Y}
fun {$}
  (V#F) = {X}
  (U#H) = {Y} in
    if (V < U) then (V#{Merge F Y})
    else
      if (V > U) then (U#{Merge X H})
      else (V#{Merge F H})
    end
  end
end
end
end
proc {Display X N}
  fun {DisplayH Z Num}
    if (Num == 0) then nil

```

```

    else
      (V#F) = {Z} in
      (V|{DisplayH F (Num-1)})
    end
  end
local L in
  L = {DisplayH X N}
  skip Browse L
end
end

```

```

//Testing
X = {Generate 3}
Y = {Generate 5}
{Display {Merge X Y} 8}
end

```

a4. local X Y Generate Display DisplayH Merge Times H in

```

fun {Generate N}
  fun {$} (N#{Generate (N+1)}) end
end
fun {Times X Y}
  fun {$}
    (V#F) = {X} in
    ((V*Y)#{Times F Y})
  end
end
Merge = fun {$ X Y}
  fun {$}
    (V#F) = {X}
    (U#H) = {Y} in
    if (V < U) then (V#{Merge F Y})
    else
      if (V > U) then (U#{Merge X H})
    else (V#{Merge F H})
    end
  end
end
end
end
H = fun {$}

```

```

(1#{Merge {Times H 2} {Merge {Times H 3} {Times H 5}}})
end
proc {Display X N}
  fun {DisplayH Z Num}
    if (Num == 0) then nil
    else
      (V#F) = {Z} in
      (V|{DisplayH F (Num-1)})
    end
  end
end
local L in
  L = {DisplayH X N}
  skip Browse L
end
end

//Testing
{Display H 10}
end

```

40. Hamming Problem

x_1	24	x_2	36	x_3	60
	30		40		80
	40		48		96
	48		60		120
	60		72		144
	72		80		160
	80		96		192
	96		120		240
	120		144		288
	144		160		320
	160		192		384
	192		240		480
	240		288		576
	288		320		640
	320		384		768
	384		480		960
	480		576		1152
	576		640		1280
	640		768		1536
	768		960		1920
	960		1152		2304
	1152		1280		2560
	1280		1536		3072
	1536		1920		3840
	1920		2304		4608
	2304		2560		5120
	2560		3072		6144
	3072		3840		7680
	3840		4608		9216
	4608		5120		10240
	5120		6144		11904
	6144		7680		14336
	7680		9216		17088
	9216		10240		20080
	10240		11904		23328
	11904		14336		27872
	14336		17088		33728
	17088		20080		40960
	20080		23328		49600
	23328		27872		59680
	27872		33728		71200
	33728		40960		85280
	40960		49600		101920
	49600		59680		121280
	59680		71200		143520
	71200		85280		168800
	85280		101920		197120
	101920		121280		228480
	121280		143520		272640
	143520		168800		329600
	168800		197120		399360
	197120		228480		482880
	228480		272640		580160
	272640		329600		691200
	329600		399360		826240
	399360		482880		986240
	482880		580160		1172160
	580160		691200		1384960
	691200		826240		1624640
	826240		986240		1902080
	986240		1172160		2218240
	1172160		1384960		2673600
	1384960		1624640		3278080
	1624640		1902080		4032640
	1902080		2218240		4947200
	2218240		2673600		6031840
	2673600		3278080		7396480
	3278080		4032640		9051840
	4032640		4947200		11007040
	4947200		5801600		13272000
	5801600		6912000		15856000
	6912000		8262400		18868800
	8262400		9862400		22320000
	9862400		11721600		26220800
	11721600		13849600		30572800
	13849600		16246400		35385600
	16246400		19020800		40659200
	19020800		22182400		47393600
	22182400		26736000		55588800
	26736000		32780800		65244800
	32780800		40326400		77472000
	40326400		49472000		92281600
	49472000		58016000		109772800
	58016000		69120000		130048000
	69120000		82624000		154208000
	82624000		98624000		182256000
	98624000		117216000		214288000
	117216000		138496000		250304000
	138496000		162464000		290416000
	162464000		190208000		334624000
	190208000		221824000		383040000
	221824000		267360000		435776000
	267360000		327808000		492928000
	327808000		403264000		554496000
	403264000		494720000		620576000
	494720000		580160000		691200000
	580160000		691200000		776384000
	691200000		826240000		876224000
	826240000		986240000		990832000
	986240000		1172160000		1110208000
	1172160000		1384960000		1244448000
	1384960000		1624640000		1393552000
	1624640000		1902080000		1557536000
	1902080000		2218240000		1736384000
	2218240000		2673600000		1930112000
	2673600000		3278080000		2138720000
	3278080000		4032640000		2362208000
	4032640000		4947200000		2600576000
	4947200000		5801600000		2853824000
	5801600000		6912000000		3121952000
	6912000000		8262400000		3405056000
	8262400000		9862400000		3703136000
	9862400000		11721600000		4016192000
	11721600000		13849600000		4344224000
	13849600000		16246400000		4687232000
	16246400000		19020800000		5045216000
	19020800000		22182400000		5418176000
	22182400000		26736000000		5806112000
	26736000000		32780800000		6209024000
	32780800000		40326400000		6626912000
	40326400000		49472000000		7059776000
	49472000000		58016000000		7506608000
	58016000000		69120000000		7967408000
	69120000000		82624000000		8442080000
	82624000000		98624000000		8930624000
	98624000000		117216000000		9433040000
	117216000000		138496000000		9949232000
	138496000000		162464000000		10479200000
	162464000000		190208000000		11022944000
	190208000000		221824000000		11580464000
	221824000000		267360000000		12151760000
	267360000000		327808000000		12736832000
	327808000000		403264000000		13335680000
	403264000000		494720000000		13948304000
	494720000000		580160000000		14574688000
	580160000000		691200000000		15214832000
	691200000000		826240000000		15868736000
	826240000000		986240000000		16536384000
	986240000000		1172160000000		17217776000
	1172160000000		1384960000000		17912912000
	1384960000000		1624640000000		18621792000
	1624640000000		1902080000000		19344416000
	1902080000000		2218240000000		20080784000
	2218240000000		2673600000000		20830896000
	2673600000000		3278080000000		21594752000
	3278080000000		4032640000000		22372352000
	4032640000000		4947200000000		23163696000
	4947200000000		5801600000000		23968784000
	5801600000000		6912000000000		24787712000
	6912000000000		8262400000000		25620480000
	8262400000000		9862400000000		26466976000
	9862400000000		11721600000000		27327200000
	11721600000000		13849600000000		28201152000
	13849600000000		16246400000000		29088832000
	16246400000000		19020800000000		29990240000
	19020800000000		22182400000000		30905376000
	22182400000000		26736000000000		31834240000
	26736000000000		32780800000000		32776832000
	32780800000000		40326400000000		33733152000
	40326400000000		49472000000000		34703296000
	49472000000000		58016000000000		35687264000
	58016000000000		69120000000000		36685056000
	69120000000000		82624000000000		37696672000
	82624000000000		98624000000000		38721120000
	98624000000000		117216000000000		39758400000
	117216000000000		138496000000000		40808512000
	138496000000000		162464000000000		41871456000
	162464000000000		190208000000000		42947232000
	190208000000000		221824000000000		44035840000
	221824000000000		267360000000000		45137280000
	267360000000000		327808000000000		46251552000
	327808000000000		403264000000000		47378656000
	403264000000000		494720000000000		48518592000
	494720000000000		580160000000000		49671360000
	580160000000000		691200000000000		50836960000
	691200000000000		826240000000000		52015392000
	826240000000000		986240000000000		53206640000
	986240000000000		1172160000000000		54410704000
	1172160000000000		1384960000000000		55627584000
	1384960000000000		1624640000000000		56857280000
	1624640000000000		1902080000000000		58099792000
	1902080000000000		2218240000000000		59365120000
	2218240000000000		2673600000000000		60643264000
	2673600000000000		3278080000000000		61934224000
	3278080000000000		4032640000000000		63238000000
	4032640000000000		4947200000000000		64554592000
	4947200000000000		5801600000000000		65883904000
	5801600000000000		6912000000000000		67226032000
	6912000000000000		8262400000000000		68580976000
	8262400000000000		9862400000000000		69948736000
	9862400000000000		11721600000000000		71329312000
	11721600000000000		13849600000000000		72722704000
	13849600000000000		16246400000000000		74128912000
	16246400000000000		19020800000000000		75547936000
	19020800000000000		22182400000000000		76979776000
	22182400000000000		26736000000000000		78424432000
	26736000000000000		32780800000000000		79881904000
	32780800000000000		40326400000000000		81352192000
	40326400000000000		49472000000000000		82835304000
	49472000000000000		58016000000000000		84331232000
	58016000000000000		691200000000000		

```
*Main> G (v2, f2) = f1
```

```
*Main> v
```

```
0
```

```
*Main> v1
```

```
1
```

```
*Main> v2
```

```
0
```

```
b2. interleave :: [Gen Int] -> Gen Int
```

```
interleave (x:xs) = let G(v1,f1) = x in
```

```
    G(v1, (interleave (xs ++ [f1])))
```

```
*Main> G (v1,f) = interleave [gen 3, gen 7, gen 13]
```

```
*Main> G (v1,f1) = f
```

```
*Main> G (v2,f2) = f1
```

```
*Main> G (v3,f3) = f2
```

```
*Main> v
```

```
0
```

```
*Main> v1
```

```
7
```

```
*Main> v2
```

```
13
```

```
*Main> v3
```

```
4
```

Part 2:

```
local GateMaker AndG OrG NotG A B S IntToNeed Out MulPlex in
```

```
GateMaker = fun {$ F}
```

```
    fun {$ Xs Ys} T
```

```
        GateLoop = fun {$ Xs Ys}
```

```
            case Xs of nil then nil
```

```
            [] |(1:X 2:Xr) then
```

```
                case Ys of nil then nil
```

```
                [] |(1:Y 2:Yr) then
```

```
                    ({F X Y})|{GateLoop Xr Yr}
```

```
                end
```

```
            end
```

```

        end
    in
        T = thread X = {GateLoop Xs Ys} in X end // thread wasn't
added to expressions
    T
end
end

```

```

NotG = fun {$ Xs} T
    Loop = fun {$ Xs}
        case Xs of nil then nil
        [] |(1:X 2:Xr) then ((1+(X*-1))){Loop Xr}
        end
    end
    in T = thread X = {Loop Xs} in X end T
end

```

```

AndG = {GateMaker fun {$ X Y} if (X == 0) then 0 else (X*Y) end end}
OrG = {GateMaker fun {$ X Y} if (X == 1) then 1 else (X+Y) end end}

```

```

fun {IntToNeed L}
    case L of nil then nil
    [] |(1:X 2:Xr) then T W in
        byNeed fun {$} X end W
        T = {IntToNeed Xr}
        (W|T)
    end
end

```

```

fun {MulPlex A B S} R Z T W in
    R = {NotG S}
    Z = {AndG R A}
    T = {AndG S B}
    W = {OrG Z T}
    W
end

```

```

A = {IntToNeed [0 1 1 0 0 1]}
B = {IntToNeed [1 1 1 0 1 0]}

```

```
S = [1 0 1 0 1 1]
Out = {MulPlex A B S}
```

```
// run a loop so the MulPlex threads can finish before displaying Out
```

```
local Loop in
```

```
proc {Loop X}
```

```
  if (X == 0) then skip Basic
```

```
  else {Loop (X-1)} end
```

```
end
```

```
{Loop 1000}
```

```
end
```

```
  skip Browse Out
```

```
end
```

```
a. fun {IntToNeed L}
```

```
  case L of nil then nil
```

```
  [] |(1:X 2:Xr) then T W in
```

```
  byNeed fun {$} X end W
```

```
  T = {IntToNeed Xr}
```

```
  (W|T)
```

```
end
```

```
end
```

```
b. AndG = {GateMaker fun {$ X Y} if (X == 0) then 0 else (X*Y) end end}
```

```
OrG = {GateMaker fun {$ X Y} if (X == 1) then 1 else (X+Y) end end}
```

```
c. fun {MulPlex A B S} R Z T W in
```

```
  R = {NotG S}
```

```
  Z = {AndG R A}
```

```
  T = {AndG S B}
```

```
  W = {OrG Z T}
```

```
  W
```

```
end
```

```
d.
```

d1. It depends on the value of S. If S is zero, it does not need the values of A and B. If S is 1, then it needs the value of the other variables. For instance, if A = 0, B=1, and S=1, then S will take the values of other variables like A= 0 and B=1. If A= 0, B=0, and S=0, then S will not take the values of other variables because it doesn't need it.

d2. Yes, they match up with what I got in (d.1).