Problem 1

a)

f 𝛜 domain(s’) c 𝛜 domain(s’)

--------------------- ----------------------

<f, s’> ⇓e <n1, s’> <c, s’> ⇓e <n2, s’>

-------------------------------------------------------------------------------------

<f ⨉ c, s’> ⇓e <n1 ⨉ n2, s’>

------------------------------------------------------------------------------------------------------

<f := f ⨉ c, s’> ⇓c <s’[f -> n1 ⨉ n2]>

b)

i)

x is assigned the value of E\_1.  
When x is less than E\_2 or equal to, the command C gets executed.  
Then the command is repeated with E1 set to the value of x+1, until x is greater than E\_2.

The for loop can become infinite when we have the same variable x in E\_2, e.g.

for x from 1 to x do skip

This command will never be able to terminate.

ii)

This part is omitted, but note that the tree is pretty big, and error-prone to draw.

Final value of f = 6 and c = 4

c)

i)

**Incorrect:**

<E\_1, s> -> <E\_1’, s’>

------------------------------------------------------------------------------------------

<for x from E\_1 to E\_2 do C, s> -> <for x from E\_1' to E\_2 do C, s’>

<E\_2, s> -> <E\_2’, s’>

------------------------------------------------------------------------------------------

<for x from n\_1 to E\_2 do C, s> -> <for x from n\_1 to E\_2’ do C, s’>

<n\_1 <= n\_2, s> -> <true, s>

------------------------------------------------------------------------------------------

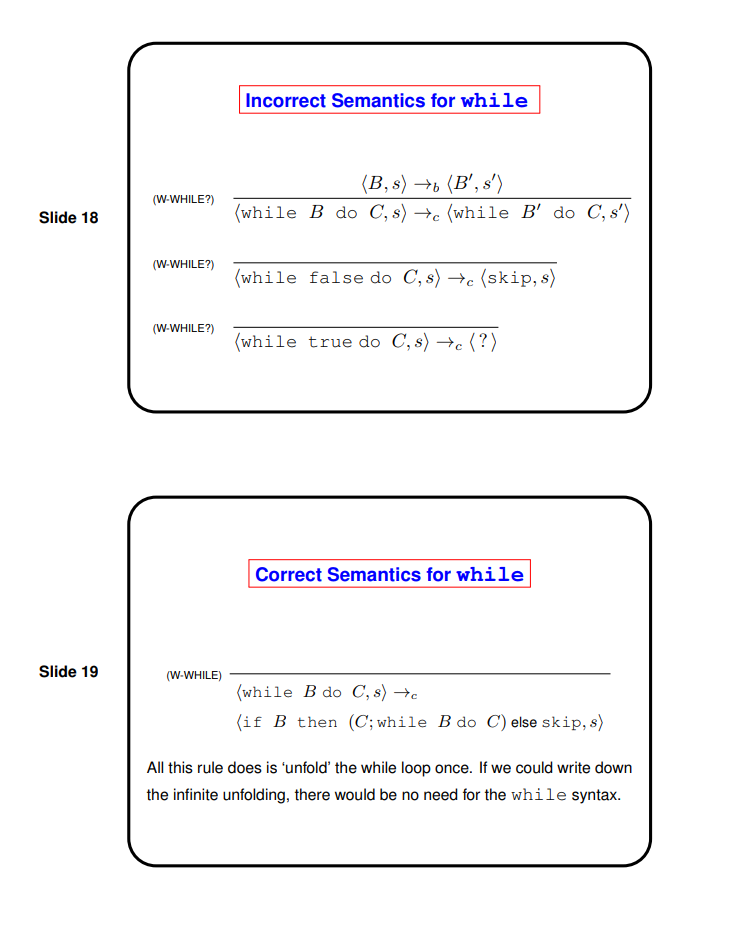
<for x from n\_1 to n\_2 do C, s> -> <x := n\_1; C; for x from x+1 to n\_2 do C, s>

<n\_1 <= n\_2, s> -> <false, s>

------------------------------------------------------------------------------------------

<for x from n\_1 to n\_2 do C, s> -> <s

>

**See slide 18-19 of Lecture 2, the proper way is to unfold.**  


Alternative solution, similar to definition of while:

<for x from E1 to E2 do C, s> ->

<x := E1; if x > E2 skip else (C; for x from x+1 to E2 do C), s>

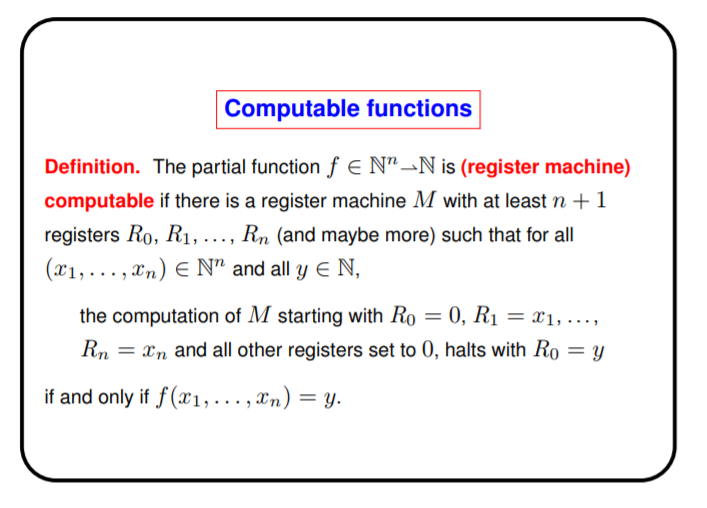
ii)

This part is omitted.

Would we have to write <skip; C, s> -> <C, s> for everything? Makes it quite a bit longer

Problem 2

a)

i) 

ii)

L\_0: R\_1- -> L\_1, L\_1

L\_1: HALT

iii)

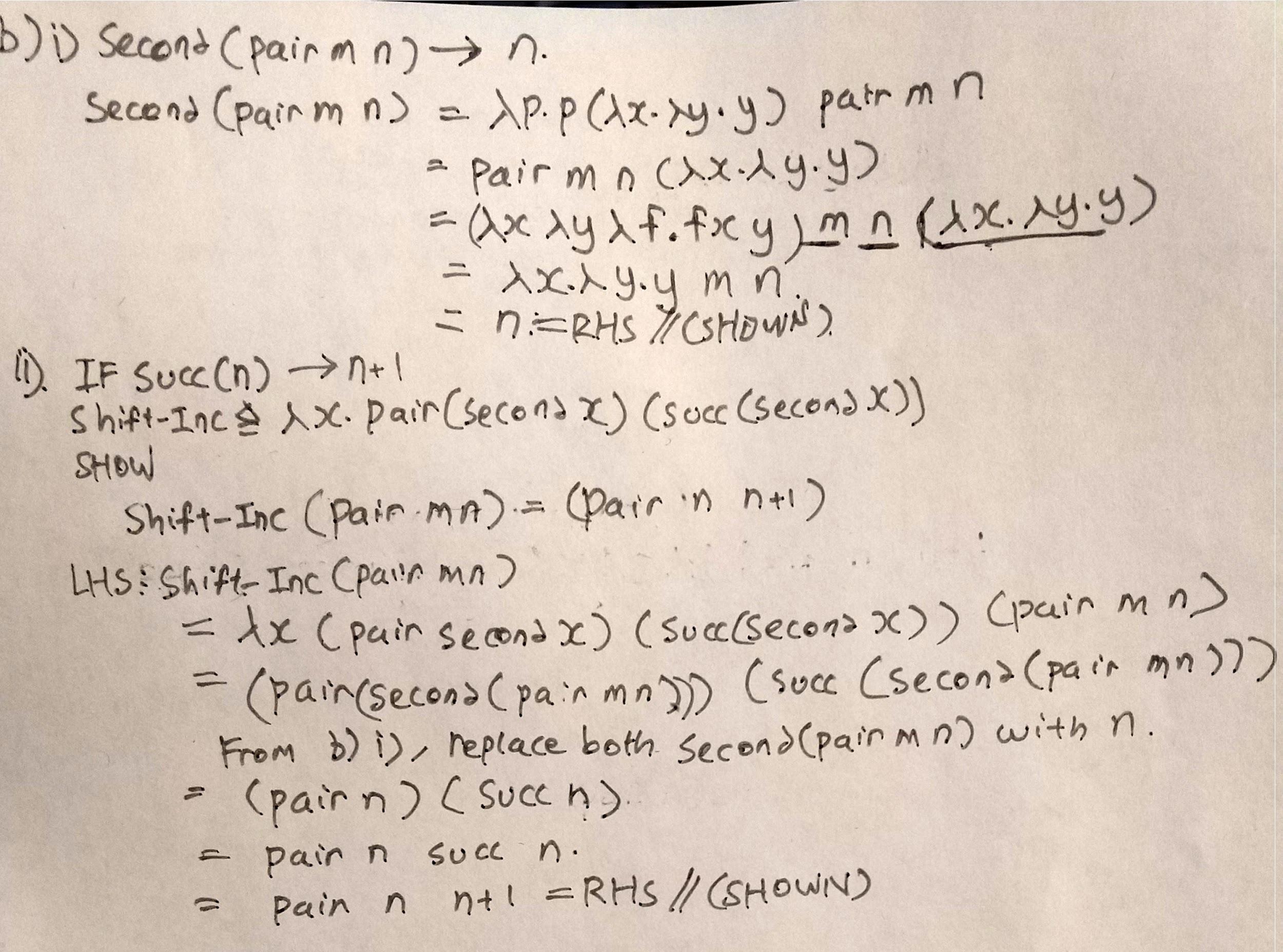
The input is a number n. The output is n = <x, y> = 2^x(2y + 1) - 1 where R\_0 = y

The first instruction means the remaining problem is n+1 = <<x, y>> = 2^x(2y+1).

Also R3 = x

b)

i and ii are pretty standard

If anyone has an answer to this that would be great =)

iii) λp. first(p shift-inc (pair 0 0))

p will be of the form λf. λx. f^n x (which represents n)

shift-inc is passed in as f

pair 0 0 is passed in as x

shift-inc is therefore applied n times to pair 0 0 which leaves pair n-1 n

first (pair n-1 n) reduces to n-1