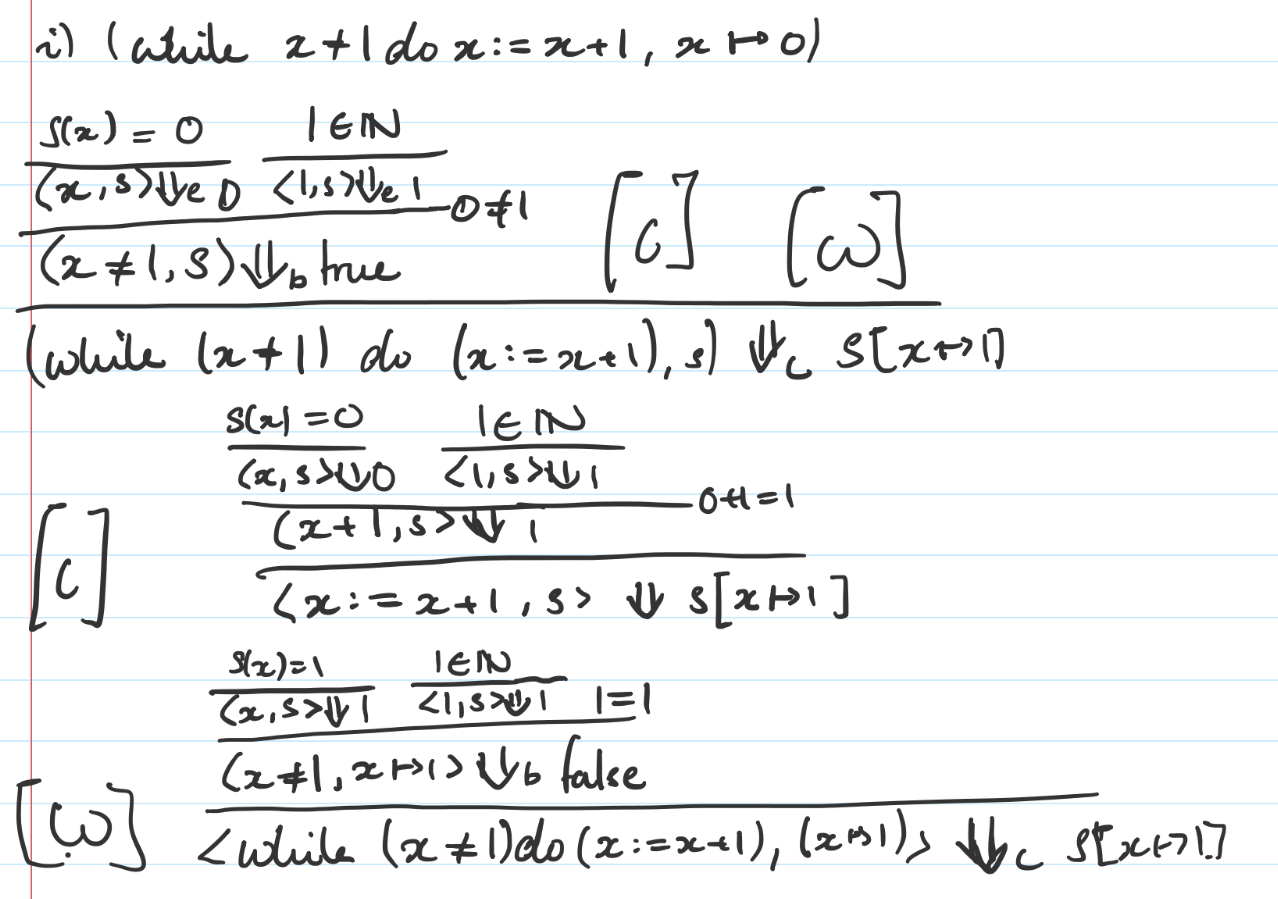
cPlease critique and improve-\*

1. 1. 1. 
      2. ( is total if forall there E exists a v)  
           
          is not total because we can use a variable that is not defined, i.e. is never true. The other way in which is not total is for an expression such as:  
         while true do x := x + 1, as this would never finish.

(I think it’s also important to mention the composition of commands might also lead to a program not terminating, so there’s 3 ways in which is not total)

Alternative answer:

All the command rules only modify the state without actually reducing to a value.



Base Cases

Holds by setting , then holds by .  
  
Two cases to consider, ,  
Apply . So set , holds.  
,  
Apply . So set , holds.  
  
Inductive step  
 and hold.  
  
Only one rule applies. By this rule (can’t be bothered to write it out), since we have , and where exists (either as a natural number or error), so holds.

* + 1. Assuming we already have we propagate *error* forward.  
         
       Imagine all the existing rules and add:  
         
          
         
       + while rules which also have error propagating ‘left to right’.
    2. is still not total because we can create situations where there is an infinite loop.  
       e.g. <while true do x:=x+1, x->0>  
         
       We could adjust c’ to catch this example, but can always reformulate the condition or loop body to get around this. Hence c’ is not total.

1. 1. 1. ~~Since the function is f: NxN -> N I’m guessing that and are the argument registers and that is the result register.  
         With this assumption, the function computed is , i.e. addition.~~  
         By definition on the slides, this might be a trick question.  
          is 0 by default, and contains the value that f computes once the machine halts. Then , correspond to the two arguments. The machine does not make use of , but that doesn’t matter.  
           
         f returns the first of its arguments, i.e. copies into .
      2. L0 = R1- → L1, L2 = ⟪3, ⟨1,2⟩⟫ = 152  
         L1 = R0+ → L0 = ⟪0, 0⟫ = 1  
         L2 = HALT = 0  
           
         Prog = list(152, 1, 0) = ⟪152, ⟪1, ⟪0, **0**⟫⟫⟫ = ⟪152, ⟪1, 1⟫⟫ = ⟪152, 6⟫ =
      3. R\_0 is the result, 0 by default  
         R\_1 is the program (same as above)  
         R\_2 is the list of arguments ⟪2,⟪1, 0⟫⟫=⟪2, 2⟫= .   
           
         All other registers are set to 0.
   2. 1. ~~I found it more intuitive to convert the rules into actual lambda calculus  
           
         To show (SKK)x = Ix  
           
         (SKK)x = by RED-S  
         = Ix from RED-I~~

*Alternatively, using their rules:*

b) i. SKK) x -> Kx (Kx)

-> x

ii. (SII) x -> Ix (Ix)

-> x (Ix)

-> x x

iii. S’ S’ = (S (Kx) (SII)) S’

-> (Kx) S’ ((SII) S’)

-> x ((SII) S’)

-> x (IS’ (IS’))

-> x (S’ (IS’))

-> x (S’ S’)

iii. *Alternative: Last 3 steps can be replaced with single step using rule from part ii*

S’ S’ = (S (Kx) (SII)) S’

-> (Kx) S’ ((SII) S’)

-> x ((SII) S’)

**-> x (S’ S’)**

Let P(n) ≡ (SII) S’ = xn (S’ S’). We will show that for all n ∈ N+.P(n) holds by induction over n.

*Base case:*

We must show that P(1) holds. We have that:

(SII) S’ -> IS’ (IS’)

-> S’ (IS’)

-> S’ S’

-> x (S’ S’) by previous part

= x1 (S’ S’) by definition provided

*Inductive case:*

We must show that P(k+1) holds, assuming P(k) ≡ (SII) S’ = xk (S’ S’) (not sure how we can really use this) as our inductive hypothesis. We have that:

(SII) S’-> IS’ (IS’)

-> S’ (IS’)

-> S’ S’

-> x (S’ S’)

-> x (x (S’ S’))

-\*> x ( … x (x (S’ S’)))

(k times)

-> x ( … x (x2 (S’ S’)))

(k - 1 times)

-> x ( … x (x3 (S’ S’)))

(k - 2 times)

-\*> x (xk (S’ S’))

= xk+1 (S’ S’) by definition provided

\*\*\*

Proving P(k+1) using P(k):

Could apply the provided definition of equality - show that xk+1 (S’ S’) →\* z and (SII)S’ →\* z for some common z.

xk+1 (S’ S’) →\* x(xk (S’ S’)) by def  
 →\* x((SII) S’) by inductive hypothesis P(k)  
 →\* x(S’S’) by Q2(b)(ii)  
 →\* S’S’ by first part of this question (show S’S’ = x(S’S’))

(SII)S’ →\* S’S’ by Q2(b)(ii)

By definition of the equality relation between combinator terms provided on the question paper, xk+1 (S’ S’) = (SII) S’, as required.

\*\*\*