1)

a)

<E, s> -> <E’, s’>

<x := E, s> -> <x := E’, s’>

<x := n, s> -> <skip, s[x->n]>

<C_1, s> -> <C_1’, s’>

<C_1;C_2, s> -> <C_1’;C_2, s’>

<skip;C_2, s> -> <C_2, s>

<B, s> -> <B’, s’>

<if B then C_1 else C_2, s> -> <if B’ then C_1 else C_2, s’>

<if true then C_1 else C_2, s> -> <C_1, s>

<if false then C_1 else C_2, s> -> <C_2, s>

<while B do C, s> -> <if B then (C;while B do C) else skip, s>

b)

i) {-2}

<x := x-1, s[x -> -1]>
-> <x := -1-1, s[x -> -1]>
-> <x := -2, s[x -> -1]>
-> <skip, s[x -> -2]>

ii) {1}

<x := (-1) * x, s[x -> -1]>
-> <x := (-1) * -1, s[x -> -1]>
-> <x := 1, s[x -> -1]>
-> <skip, s[x -> 1]>

iii) {1, -2}

<x := x-1 or x := (-1) * x, s[x -> -1]>
-> <x := x-1, s[x -> -1]>
-> the same as (i)

<x := x-1 or x := (-1) * x, s[x -> -1]>
-> <x := (-1) * x, s[x -> -1]>
-> the same as (ii)

iv)
Let W = while x <=0 do (x := -1 or x := (-1) * x)
Let C = x := -1 or x := (-1) * x

<while x <= 0 do C, s[x -> -1]>
-> <if x <= 0 then C;W else skip, s[x -> -1]>
-> <if -1 <= 0 then C;W else skip, s[x -> -1]>
-> <if true then C;W else skip, s[x -> -1]>
-> <C;W, s[x -> -1]>
-> <x := (-1) * -1;W, s[x -> -1]>
-> <x := 1;W, s[x -> -1]>
-> <skip;W, s[x -> 1]>
-> <while x <= 0 do x := -1 or x := (-1) * x, s[x -> 1]>
-> <if x <= 0 then (x := -1 or x := (-1) * x;W) do x := -1 or x := (-1) * x, s[x -> 1]>
Mate this is long
^^^^ 😭

c)
Base case: n = 0
Is easy

Inductive case: n = k
Assume the inductive hypothesis.

<C_1, s> ->^{k} <C_1’, s’> => <C_1;C_2, s> ->^{k} <C_1;C_2, s’>

To show
<C_1, s> -> $^{k+1}$ <C_1’’, s’’> => <C_1;C_2, s> -> $^{k+1}$ <C_1’’;C_2, s’’>

Assume

$<C_1, s> \to^{k+1} <C_1'', s''> = <C_1, s> \to^k \to <C_1', s'> \to <C_1'', s''>$

Then by I.H we have $<C_1;C_2, s> \to^k <C_1';C_2, s'>$

And

$<C_1;C_2, s> \to^k <C_1';C_2, s'> \to <C_1'';C_2, s''>$ by the rule fella

Done

d)
i)
ii)

4)
a)
i)

f is computable iff there exists a register machine M that halts iff f(x_1, ...)↓, and in that case R_0 = f(x_1, ...)↓.

iii) f(x) = 2x

Register machine explanation: By decrementing R1 from x to 0, we +2 to R2 each loop. When R1 is 0, we can safely say that R2 holds 2x. We then just shift the contents of R2 into R0.

b)
We adapt (4aiii) to run g(x, z) = (2^x)*z. This can be thought of as doing f(z), x times.

L0: R1- => L1, L6
L1: R2- => L2, L4
L2: R3+ => L3
L3: R3+ => L1
L4: R3- => L5, L0
L5: R2+ => L4
L6: R2- => L7, L8
L7: R0+ => L6
L8: HALT

Register machine explanation: We count down R1 from x to 0, each time we run the doubling function for register R2, using R3 as a scratch, of which we save the doubled result back into R2 for convenience for the next iteration. Once this is done (when we reach L6), we can say that we have (2^x*z) in register R2 so we just move this over to R0.

c)
i)

```
L0: R2- => L1, L3
L1: R3+ => L2
L2: R3+ => L0
L3: R3- => L4, L5
L4: R2+ => L3
L5: R2+ => L6
# <- Invariant: R1 holds x, R2 holds 2y+1 ->
L6: R1- => L7, L12
L7: R2- => L8, L10
L8: R3+ => L9
L9: R3+ => L7
L10: R3- => L11, L6
L11: R2+ => L10
# <- Invariant: R2 holds (2^x)*(2y+1) ->
L12: R2- => L13, L14
L13: R0+ => L12
L14: HALT
```

Register machine explanation: h(x, y) = (2^x)*(2y+1) = g(x, (2y + 1)) = g(x, f(y) + 1).

As such, we just need to write some instructions to change the contents of R2 from y to 2y + 1 by running f(y) + 1 from L0 to L5. After that is done, R1 will still contain x, R2 now contains 2y + 1 and then we can just run the solution from (4b) from L6 downwards.

ii)

Let $List\,\mathbb{N}$ be the set of all finite lists of natural numbers.

For $\ell \in List\,\mathbb{N}$, define $\ulcorner \ell \urcorner \in \mathbb{N}$ by induction on the length of the list ℓ:

$$\begin{cases} \ulcorner [] \urcorner \triangleq 0 \\ \ulcorner x :: \ell \urcorner \triangleq \langle\langle x, \ulcorner \ell \urcorner \rangle\rangle = 2^x(2 \cdot \ulcorner \ell \urcorner + 1) \end{cases}$$

Gives a bijection with N.

iii)

Since $\langle\langle -,- \rangle\rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}^+$, $\langle -,- \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $\ulcorner - \urcorner : List\,\mathbb{N} \to \mathbb{N}$ are bijections, the functions $\ulcorner - \urcorner$ from bodies to natural numbers and $\ulcorner - \urcorner$ from RM programs to $\mathbb{N}$ are bijections.

WHERE IS THE REST 😩😩😩😩😩