P;1ai. <(x := 1), s> -> <skip, s[x -> 1]>

ii. <while true do skip, s> -> <if true then (skip; while true do skip) else skip, s>

-> <skip; while true do skip, s> -> <while true do skip, s>

Loops this forever

iii. <(x := 1) or (while true do skip), s>

Case 1: LHS chosen

<(x := 1) or (while true do skip), s> -> <x := 1, s> -> <skip, s[x -> 1]>

Case 2: RHS chosen

<(x := 1) or (while true do skip), s> -> <while true do skip, s> -> …

Infinite loop as in part 1aii

b. To diverge, must be infinite loop

As shown in part ii, we get back to another configuration with <while true do skip, s’> for some state s’. However, the command skip does not change the original state s and, by the semantics of While-Or, no other command is introduced to the loop that will change the state.

Therefore s’ = s and <while true do skip, s> ->\* <while true do skip, s>

Alternative approach:

For <while true do skip, s> to converge, there exists some k in the positive natural numbers (clearly zero not applicable) such that <(while true do skip)> ->k <skip, s’>.

Assume <while true do skip, s> converges.

Repeating steps in (1aii), it then means that:

<(while true do skip), s> ->k <skip, s’>

<if true then (skip; while true do skip) else skip, s> ->(k-1) <skip, s’>.

<skip; while true do skip, s> ->(k-2) <skip, s’>

<while true do skip, s> ->(k-3) <skip, s’>

<(while true do skip), s> therefore only converges if k = k - 3, which is never true. Hence must diverge.

c. <(x := 1), s> and <(x := 1) or (while true do skip), s> can be

As shown in part 1ai, <(x := 1), s> ->\* <skip, s[x -> 1]>

As shown in part 1aiii, on one path, <(x := 1) or (while true do skip)> ->\* <skip, s[x -> 1]>

Therefore, <(x := 1), s> ~ <(x := 1) or (while true do skip), s>

No others are as <while true do skip, s> ↛\* <skip, s’> as shown in part 1aii.

Nd

1di. **NOTE**: Mistake in the exam question, should be ‘C’ in the place of “skip” (see end of this document)

Proof 1: <C, s[x -> 0, z -> 0]> ->\* <C, s[x -> n, z -> 0]> ∀ n ∈

Base Case: n = 1

<C, s[x -> 0, z -> 0]> -> <if (z = 0) then (x := x + 1; (z := 0) or (z := 1); C) else skip, s>

-> <x := x + 1; (z := 0) or (z := 1); C, s>

-><skip; (z := 0) or (z := 1); C, s[x -> 1]>

-><(z := 0) or (z := 1); C, s[x -> 1]>

-> <(z := 0; C, s[x -> 1]>

-> <skip; C, s[x -> 1, z -> 0]>

-> <C, s[x -> 1, z -> 0]>

Therefore, holds for base case

Assume true for n = k, k ∈

For n = k + 1:

<C, s[x -> 0, z -> 0]> ->\* <C, s[x -> k, z -> 0]>, by IH

By the semantics of While-Or, we must therefore have that:

<C, s[x -> 0, z -> 0]> ->\* <(z := 0) or (z := 1); C, s[x -> k, z -> 0]>

<(z := 0) or (z := 1); C, s[x -> k, z -> 0]>

-> <z := 0; C, s[x -> k, z -> 0]>

-> <skip; C, s[x -> k, z -> 0]>

-> <C, s[x -> k, z -> 0]>

-> <if (z=0) then (x := x + 1; (z := 0) or (z : = 1); C) else skip, s[x -> k, z -> 0]>

-> <x := x + 1; (z := 0) or (z : = 1); C, s[x -> k, z -> 0]>

-> <skip; (z := 0) or (z := 1); C, s[x -> k + 1, z -> 0]>

-> <(z := 0) or (z := 1); C, s[x -> k + 1, z -> 0]>

-> <z := 0; C, s[x -> k + 1, z -> 0]>

-> <skip; C, s[x -> k + 1, z -> 0]>

-> <C, s[x -> k + 1, z -> 0]>

Therefore, holds for inductive case.

Therefore, holds for all n ∈

Proof 2: <C, s[x -> 0, z -> 0]> ->\* <skip, s[x -> n, z -> 1]> ∀ n ∈

Base Case: n = 1

<C, s[x -> 0, z -> 0]> -> <if (z = 0) then (x := x + 1; (z := 0) or (z := 1); C) else skip, s>

-> <x := x + 1; (z := 0) or (z := 1); C, s>

-><skip; (z := 0) or (z := 1); C, s[x -> 1]>

-><(z := 0) or (z := 1); C, s[x -> 1]>

-> <(z := 1; C, s[x -> 1]>

-> <skip; C, s[x -> 1, z -> 1]>

-> <C, s[x -> 1, z -> 1]>

-> <if (z = 0) then (x := x + 1; (z := 0) or (z := 1); C) else skip, s[x -> 1, z -> 1]>

-> <skip, s[x -> 1, z -> 1]>

Therefore holds for base case

Assume true for n = k, k ∈

For n = k + 1:

<C, s[x -> 0, z -> 0]> ->\* <skip, s[x -> k, z -> 1]>, by IH

By the semantics of While-Or, we must therefore have that:

<C, s[x -> 0, z -> 0]> ->\* <(z := 0) or (z := 1); C, s[x -> k, z -> 0]>

<(z := 0) or (z := 1); C, s[x -> k, z -> 0]>

-> <z := 0; C, s[x -> k, z -> 0]>

-> <skip; C, s[x -> k, z -> 0]>

-> <C, s[x -> k, z -> 0]>

-> <if (z=0) then (x := x + 1; (z := 0) or (z : = 1); C) else skip, s[x -> k, z -> 0]>

-> <x := x + 1; (z := 0) or (z : = 1); C, s[x -> k, z -> 0]>

-> <skip; (z := 0) or (z := 1); C, s[x -> k + 1, z -> 0]>

-> <(z := 0) or (z := 1); C, s[x -> k + 1, z -> 0]>

-> <z := 1; C, s[x -> k + 1, z -> 0]>

-> <skip; C, s[x -> k + 1, z -> 1]>

-> <C, s[x -> k + 1, z -> 1]>

-> <if (z = 0) then (x := x + 1; (z := 0) or (z := 1); C) else skip, s[x -> k + 1, z -> 1]>

-> <skip, s[x -> k + 1, z -> 1]>

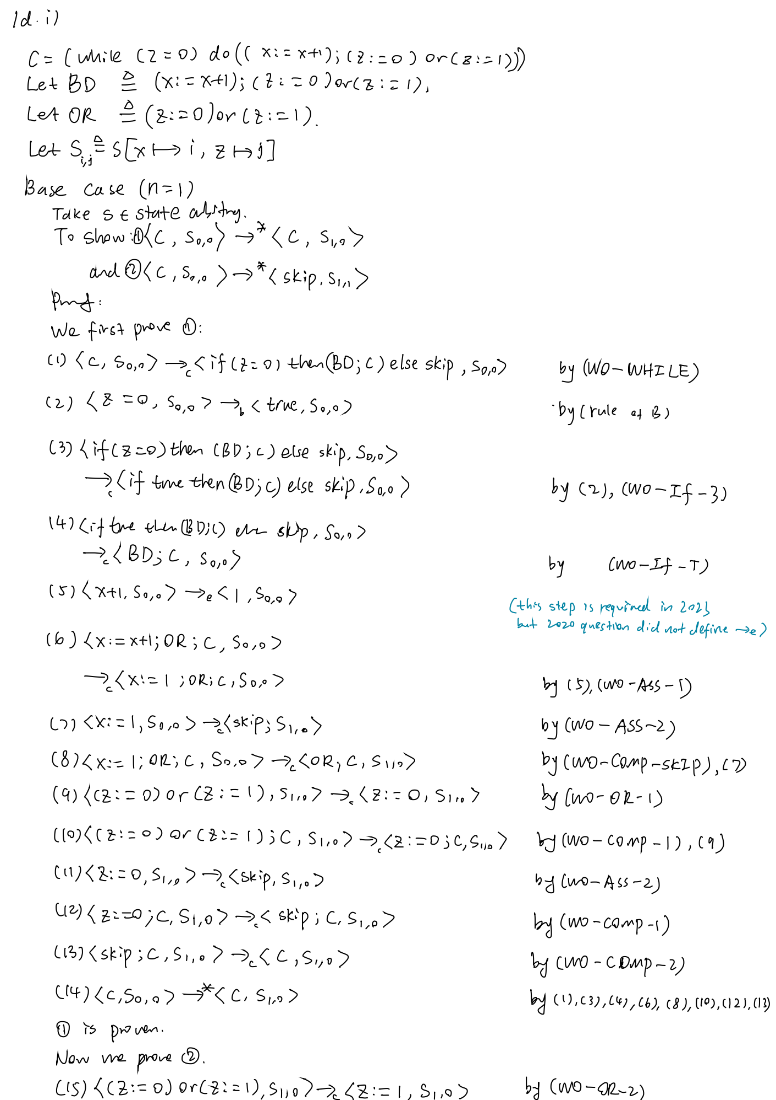
Therefore holds for inductive case.

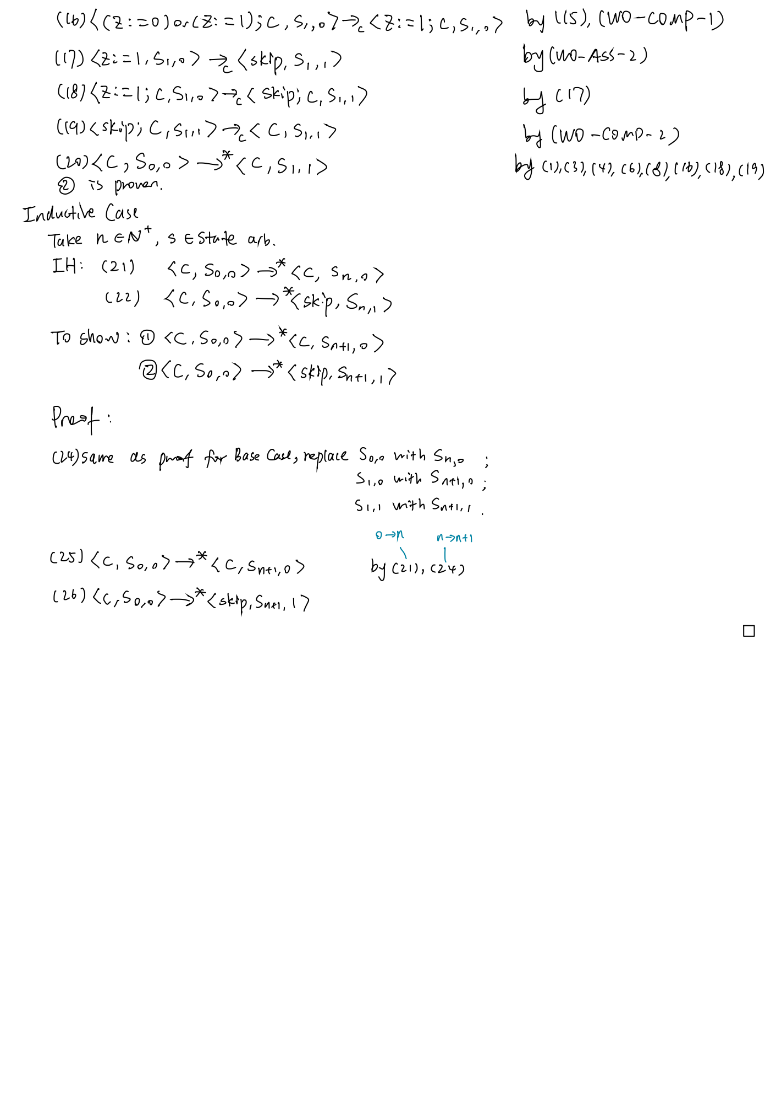
Therefore holds for all n ∈ N+

Alternative 1di) (in 2023 format):

This proof is more formative

In 2023 I think we need to refer to which rule is used in each step. But this proof took me 30 minutes so I won’t do this in the exam if it only counts for half of 40%.





1dii. <x := somenumber; z := 1, s> -> <skip; z := 1, s[x -> n]>

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

Therefore, <x := somenumber; z := 1, s> ->\* <skip, s[x -> n, z -> 1]

<x := 0; z := 0; C, s> -> <skip; z := 0; C, s[x -> 0]>

-> <z := 0; C, s[x -> 0]>

-> <skip; C, s[x -> 0, z -> 0]>

-> <C, s[x -> 0, z -> 0]>

Therefore, <x := 0; z := 0; C, s> ->\* <C, s[x -> 0, z -> 0]>

As <x := 0; z := 0; C, s> ->\* <C, s[x -> 0, z -> 0]> and, by part 1di,

<C, s[x -> 0, z -> 0]> ->\* <skip, s[x -> n, z -> 1]>:

<x := 0; z := 0; C> ->\* <skip, s[x -> n, z -> 1]>

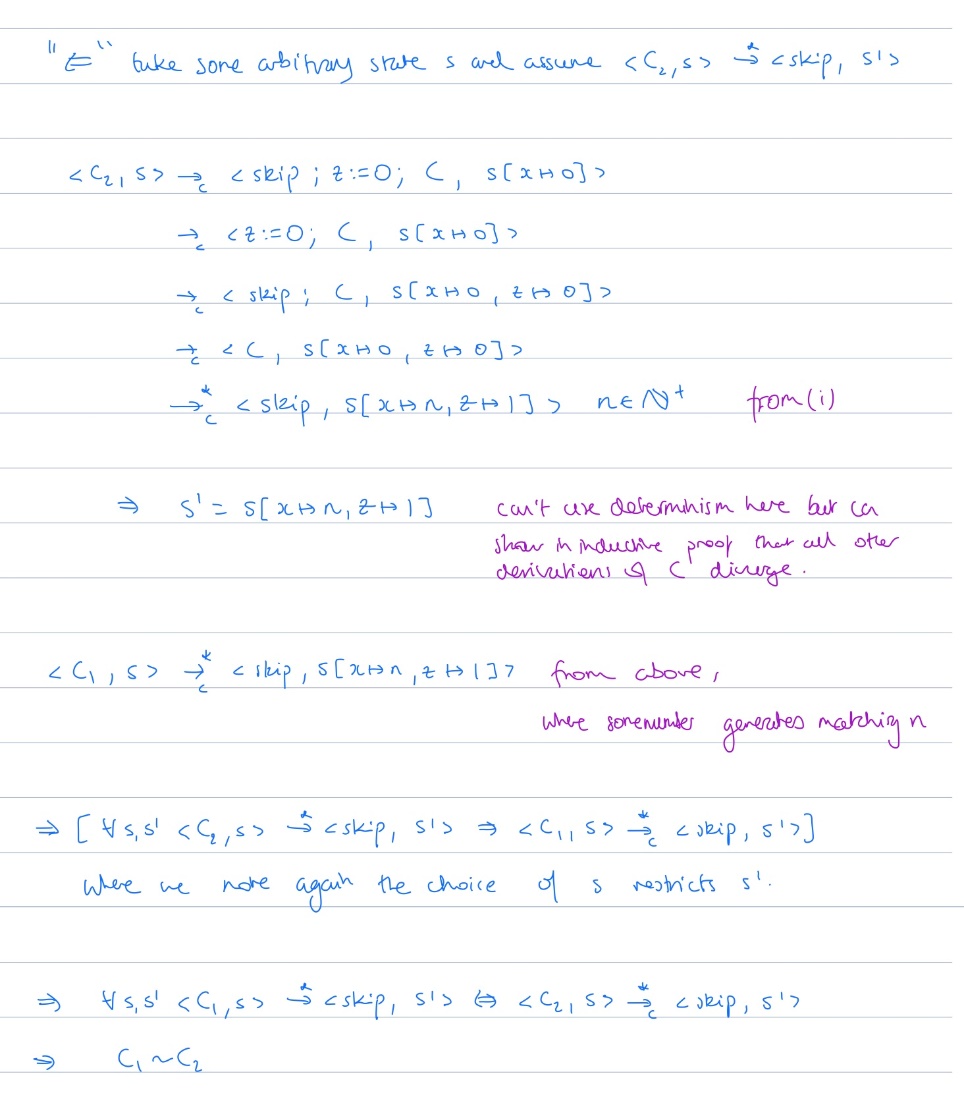
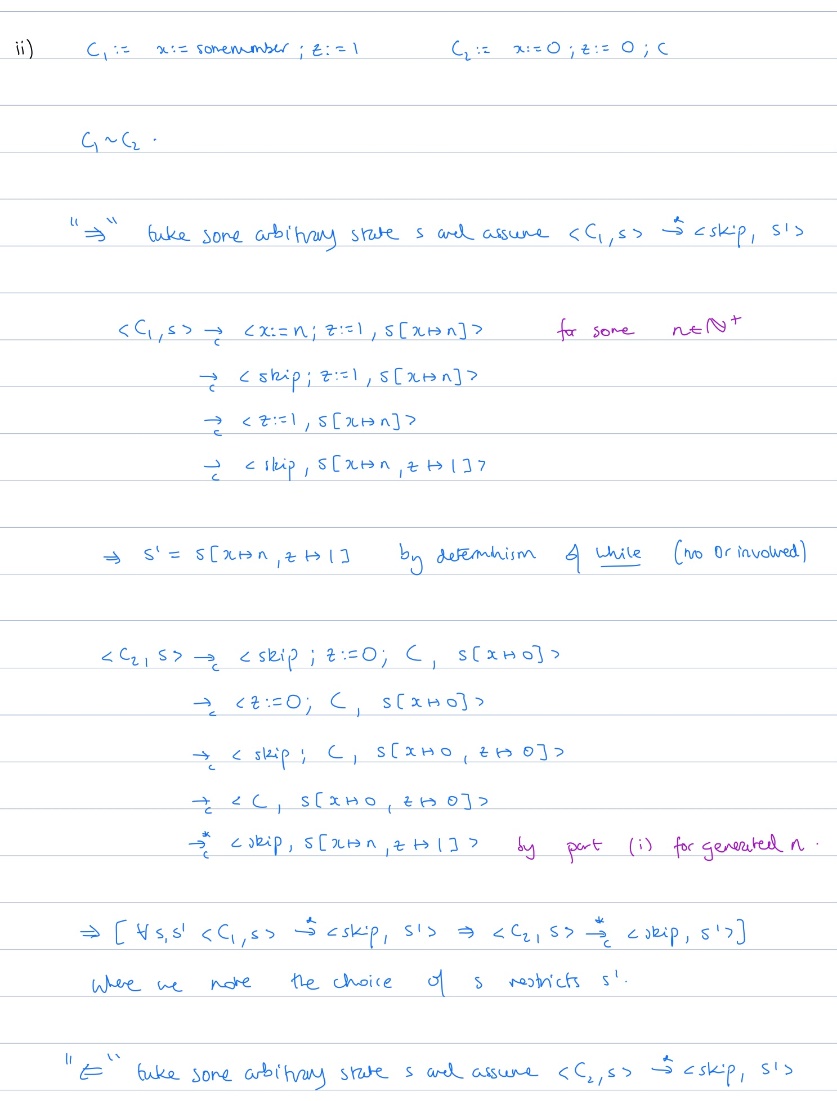
By given *definition*:

The command (x := somenumber; z := 1) is equivalent to the command (x := 0; z := 0; C) as:

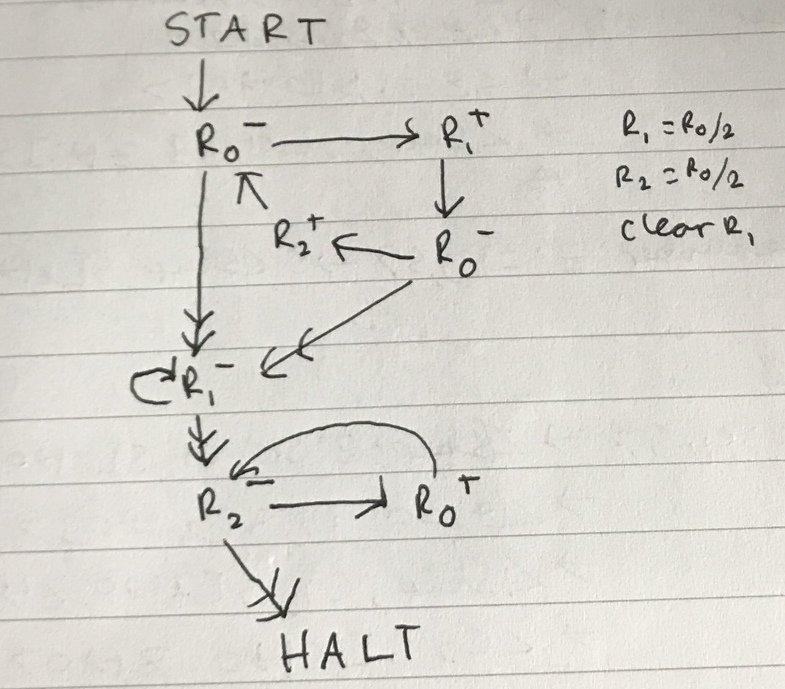
<x := somenumber; z := 1, s> ->\* <skip, s[x -> n, z -> 1]

and

<x := 0; z := 0; C> ->\* <skip, s[x -> n, z -> 1]>



2ai



Performs integer division by 2 on m (DIV 2)

Explanation: With the initial state of R0 being m with R1 = R2 = 0, we repeat the process of taking 1 away from R0 and putting it into R1 or R2 and alternating, until R0 is empty. As R1 is first, it gets the bigger half (if odd). R2 will get the smaller half (m DIV 2). We then empty R1 and move (m DIV 2) into R0.

ii.

(Note: click the equations if they are not rendered correctly)

Γ- -> , ㄱ = <<(2 x 0) + 1, <1, 4>>> = <<1, <1, 4>>> = <<1, 17>> = 2(34 + 1) = 70

Γ -> ㄱ = <<(2 x 1), 2>> = <<2, 2>> = 4(4 + 1) = 20

Γ -> , ㄱ = <<(2 x 0) + 1, <3, 4>>> = <<1, <3, 4>>> = <<1, 71>> = 2(142 + 1) = 286

iii. Γ ㄱ = 1144 = (2y + 1) = (2(71) + 1) = <<3, 71>> = <<(2 x 1) + 1, 71>>

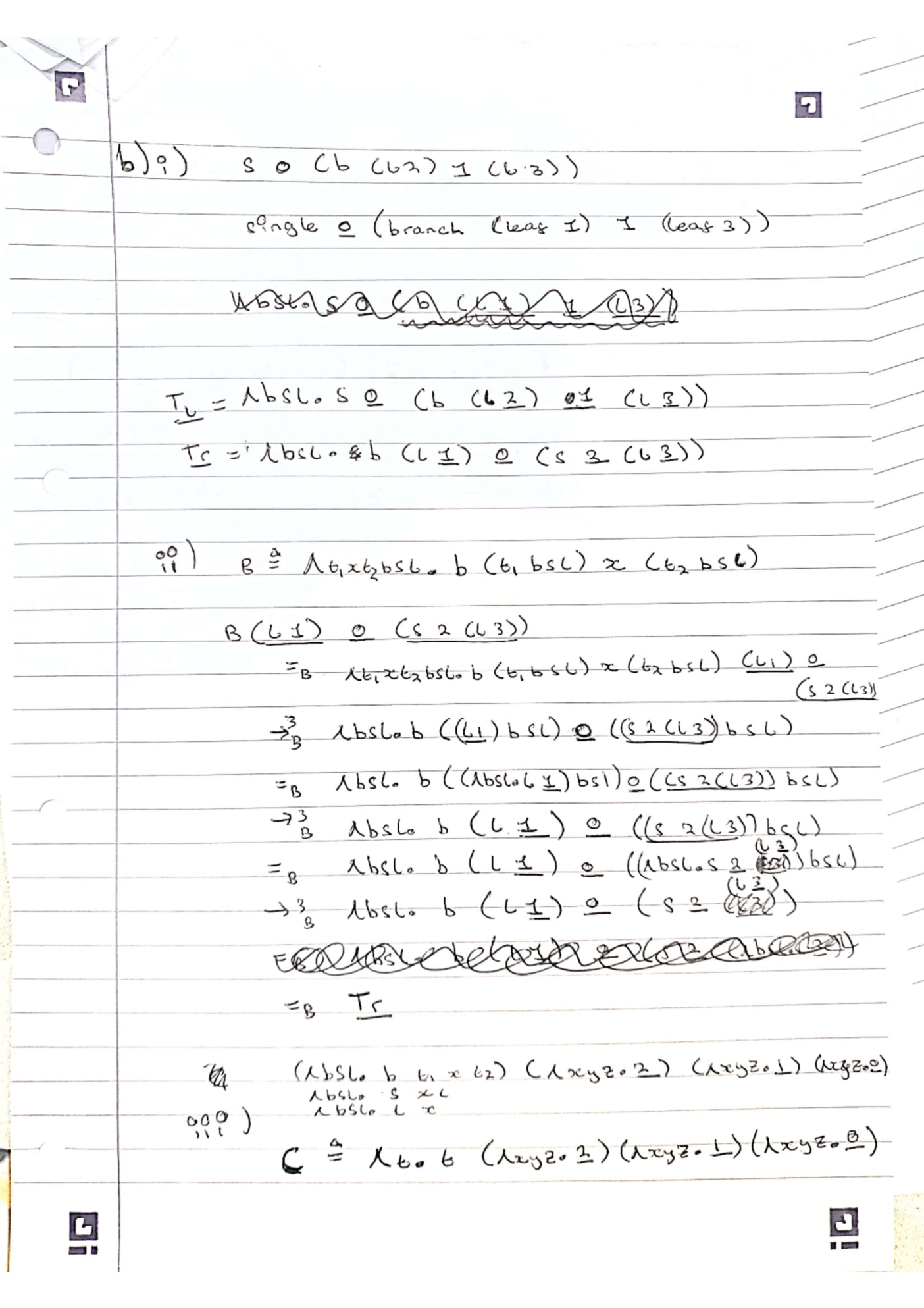
= <<(2 x 1 + 1, <3, 4>>>

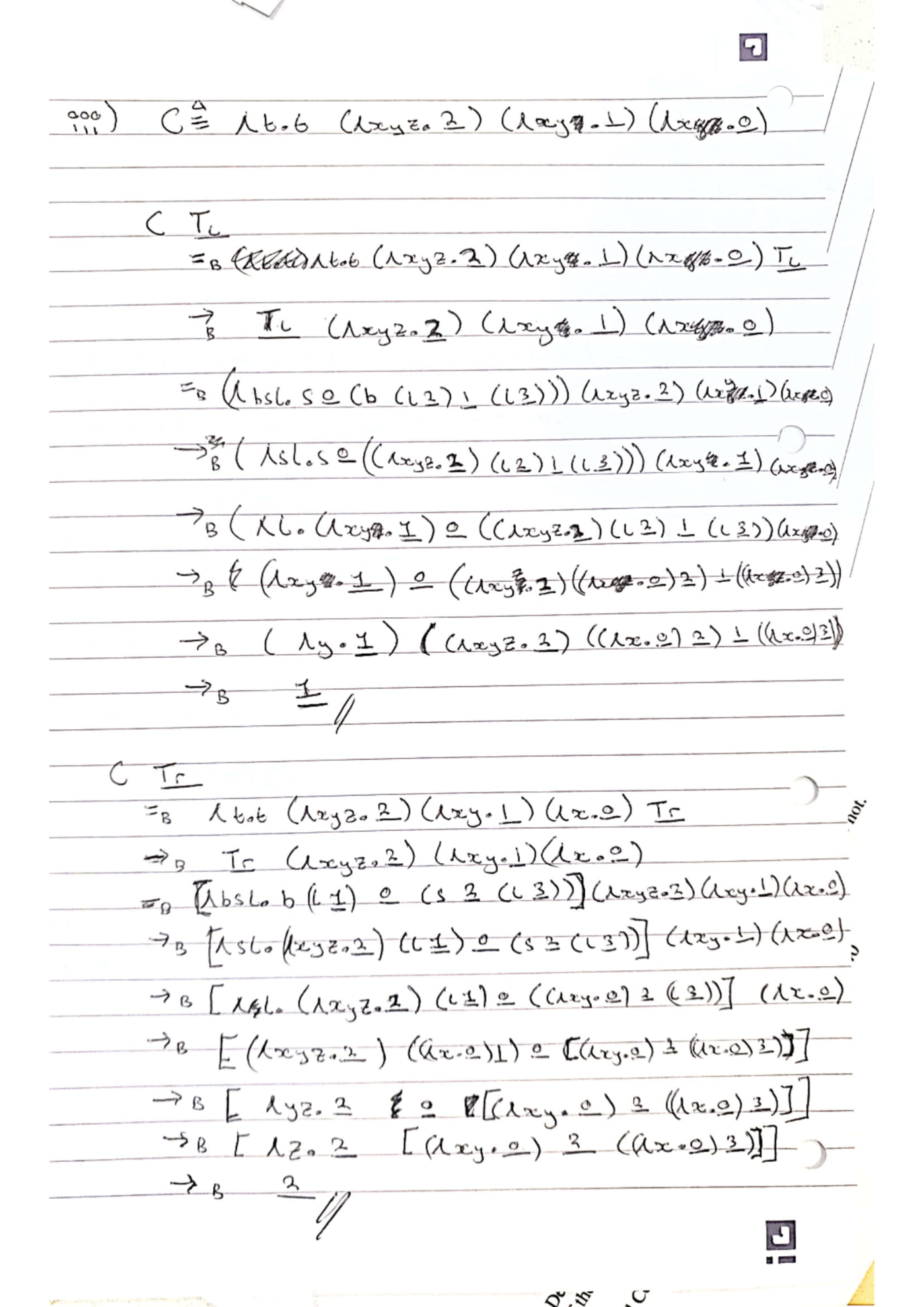
So : -> ,

Γ ㄱ = 448 = (2y + 1) = (2(3) + 1) = <<6, 3>> = <<(2 x 3), 3>>

So : ->

b. Was not taught for 2021





1.hi

1. d) yes

F



1 d) <C, s[x -> 0, z -> 0]> ->\* <C, s[x -> n, z -> 0]> was the intended meaning

000000000000000000