High-Order function:

val map : ('a → 'b) → 'a list → 'b list.

'a type : contains everything of any type

[1, 'a', "Hello"] polymorphic

int list : [1, 2, 3]

string list : ['a', 'b', ...]

→ let function (x : 'a) = x

↳ x can be any type

generate 1→N :

```
let rec gen start limit =
  if start > limit then
    []  ← empty when start > limit
  else
    start :: gen (start + 1) limit ;

let my_list = gen 1 10 ;
```

1 the 1+1 stacked → 10 max

map f [a1; ... ; an]

↳ list

↳ function f which would be applied to each element in the liste

f :

for each element "ai" in the liste, function f is applie

⇒ would change the whole list

map f [a1, ... an] = [(f a1); ... ; (f an)]

let succ x = x + 1 in map succ [1; 2; 3]

= [2; 3; 4]

f: is a function that we could replace after

take 'a only!

let rec map f liste :

let succ x = x + 1

let result = map [1; 2; 3]

↳ 'a type

return 'b

```
let rec map f l =
  match l with
    | [] -> []
    | hd::tl -> (f hd)::(map f tl)
```

ii

$n = m$ $[n]$

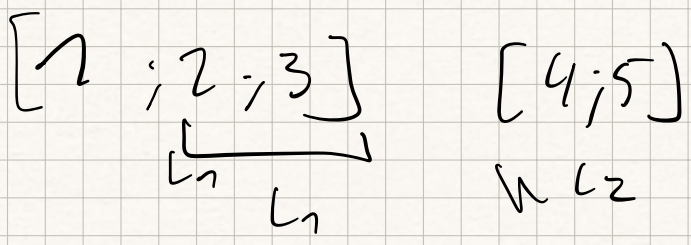
$n > m$

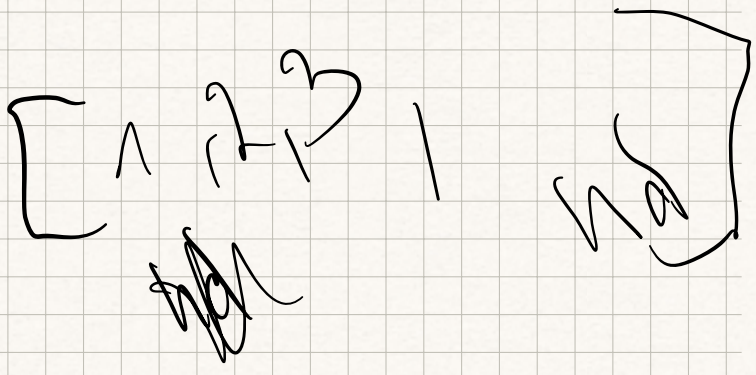
n $[n+1]$

0

$n < m$

m -1 n





[1, 2, 3 4, 5]