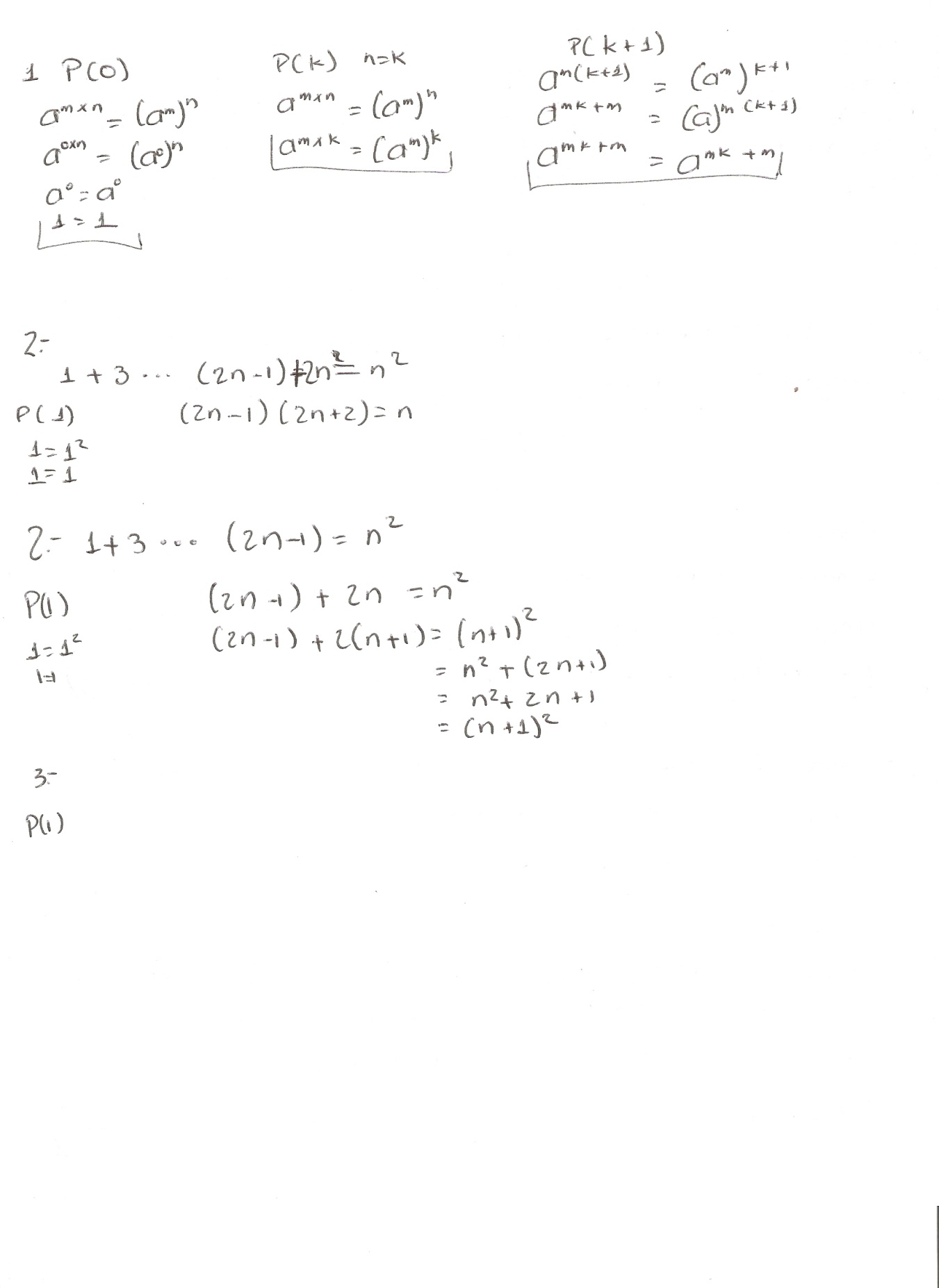
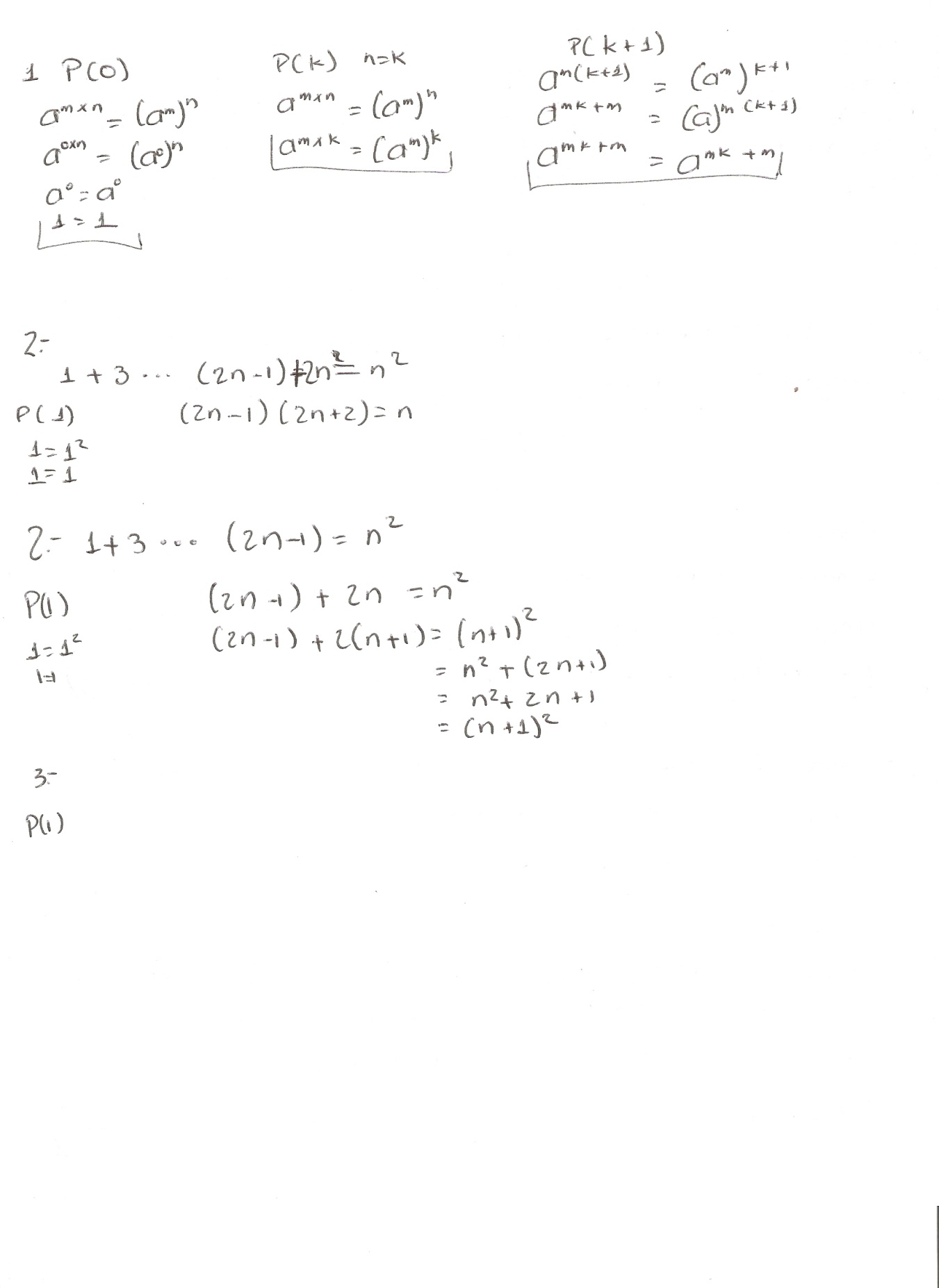
**Exercise 1.** Let *a* be an arbitrary real number. Prove, for all natural numbers

*m* and *n*, that *am×n* = (*am*)*n*.



**Exercise 2.** Prove that the sum of the first *n* odd positive numbers is *n*2.



**Exercise 3.** Prove that

\_*n*

*i*=1 *ai* = (*an*+1 *−* 1)*/*(*a −* 1), where *a* is a real

number and *a \_*= 1.

**Exercise 4.** (This problem is from [12], where you can find many more.) The *n*th Fibonacci number is defined as follows:

fib :: Integer -> Integer

fib 0 = 0

fib 1 = 1

fib (n+2) = fib n + fib (n+1)

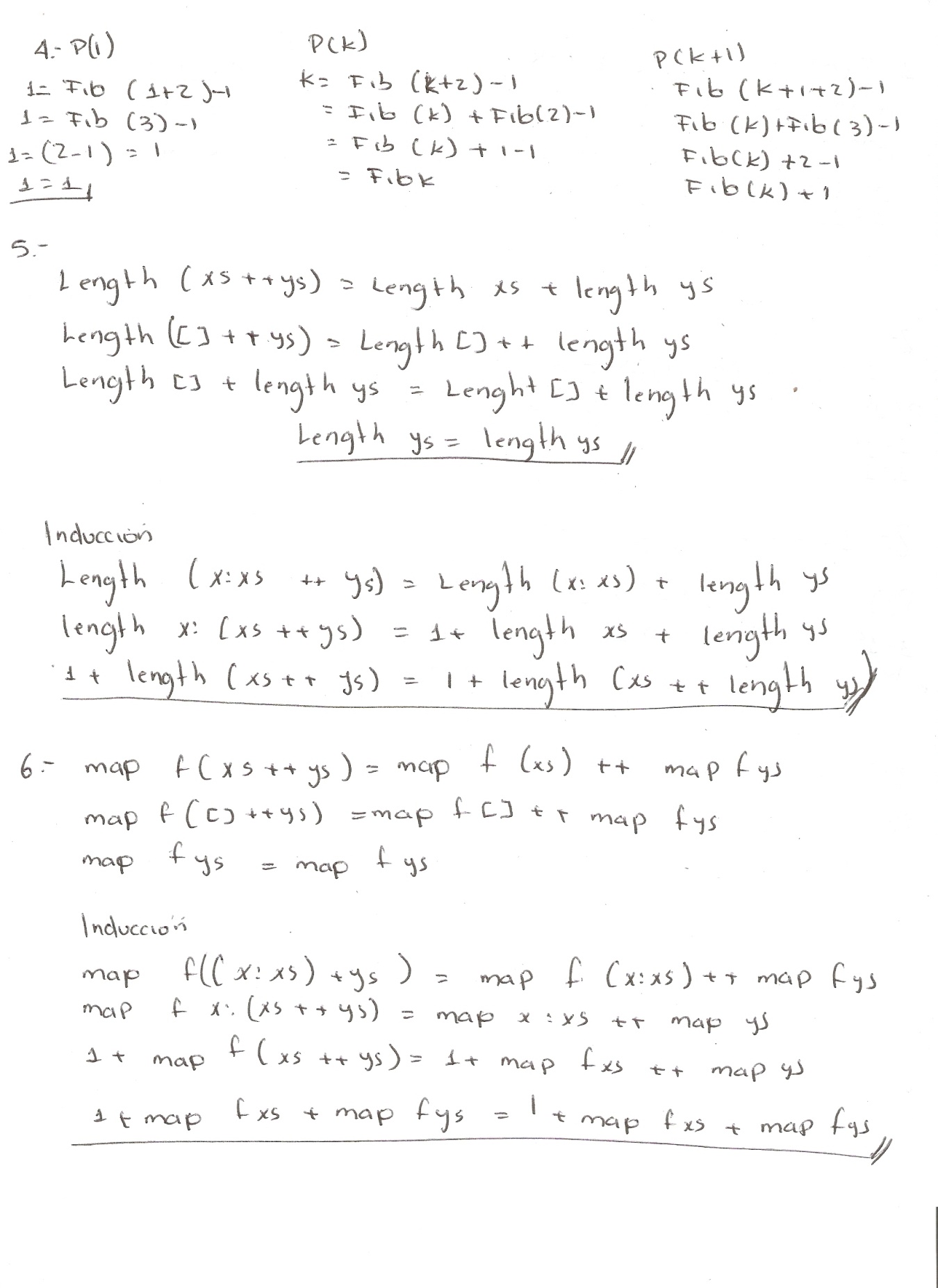
The first fewnumbers in this famous sequence are 0*,* 1*,* 1*,* 2*,* 3*,* 5*, . . .*. Prove

the following:

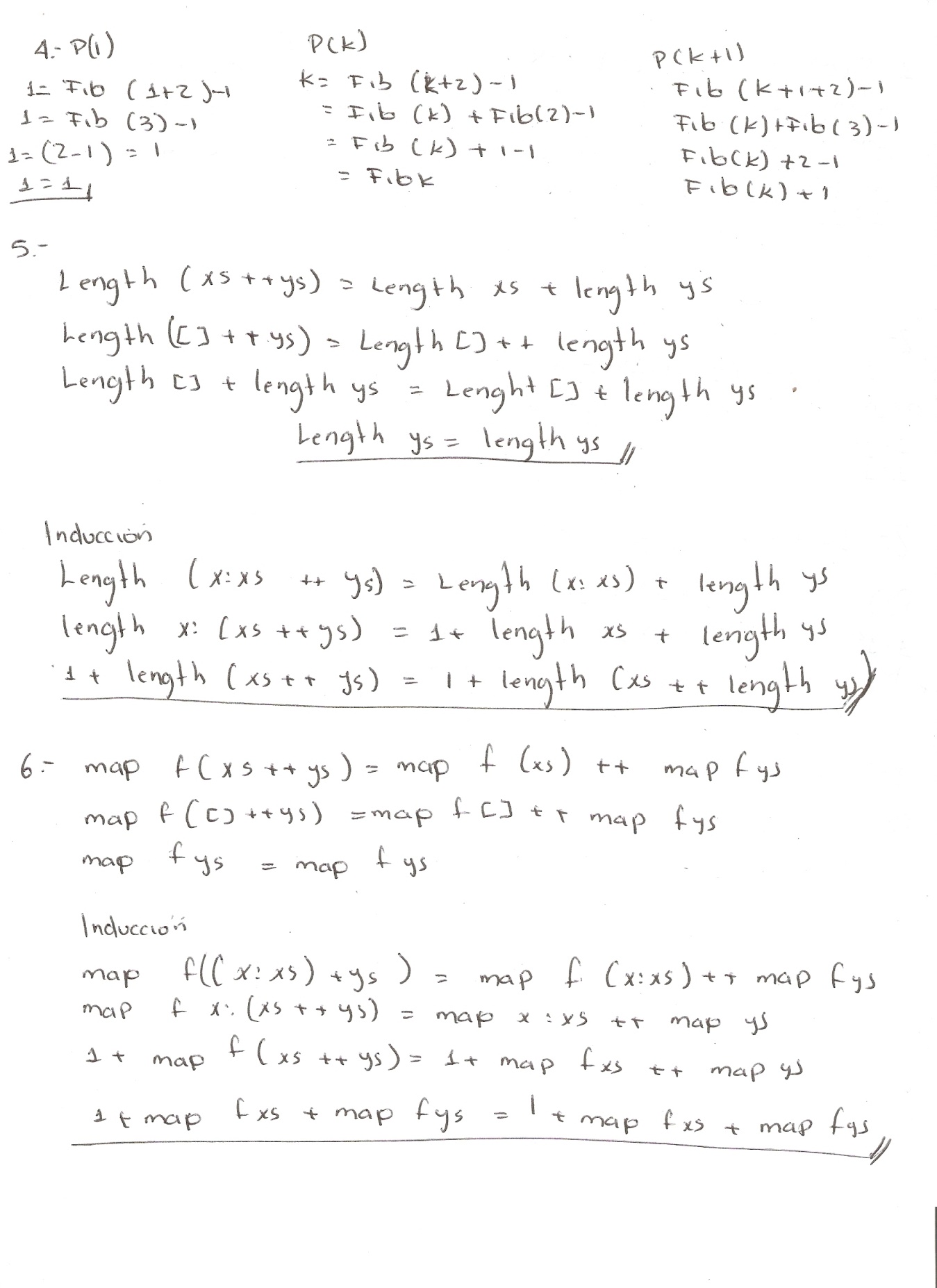
\_*n*

*i*=1

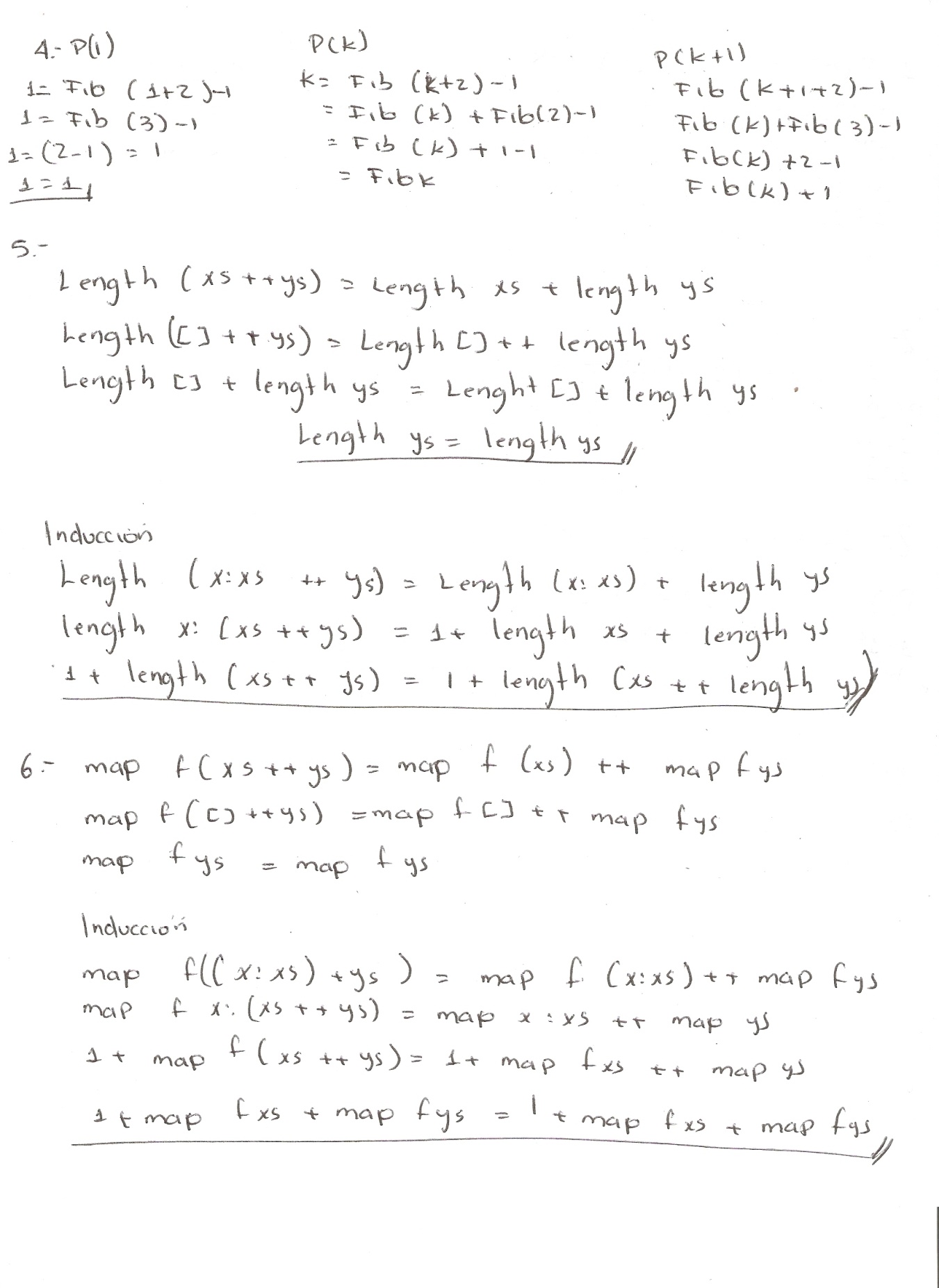
*fib i* = *fib* (*n* + 2) *–* 1



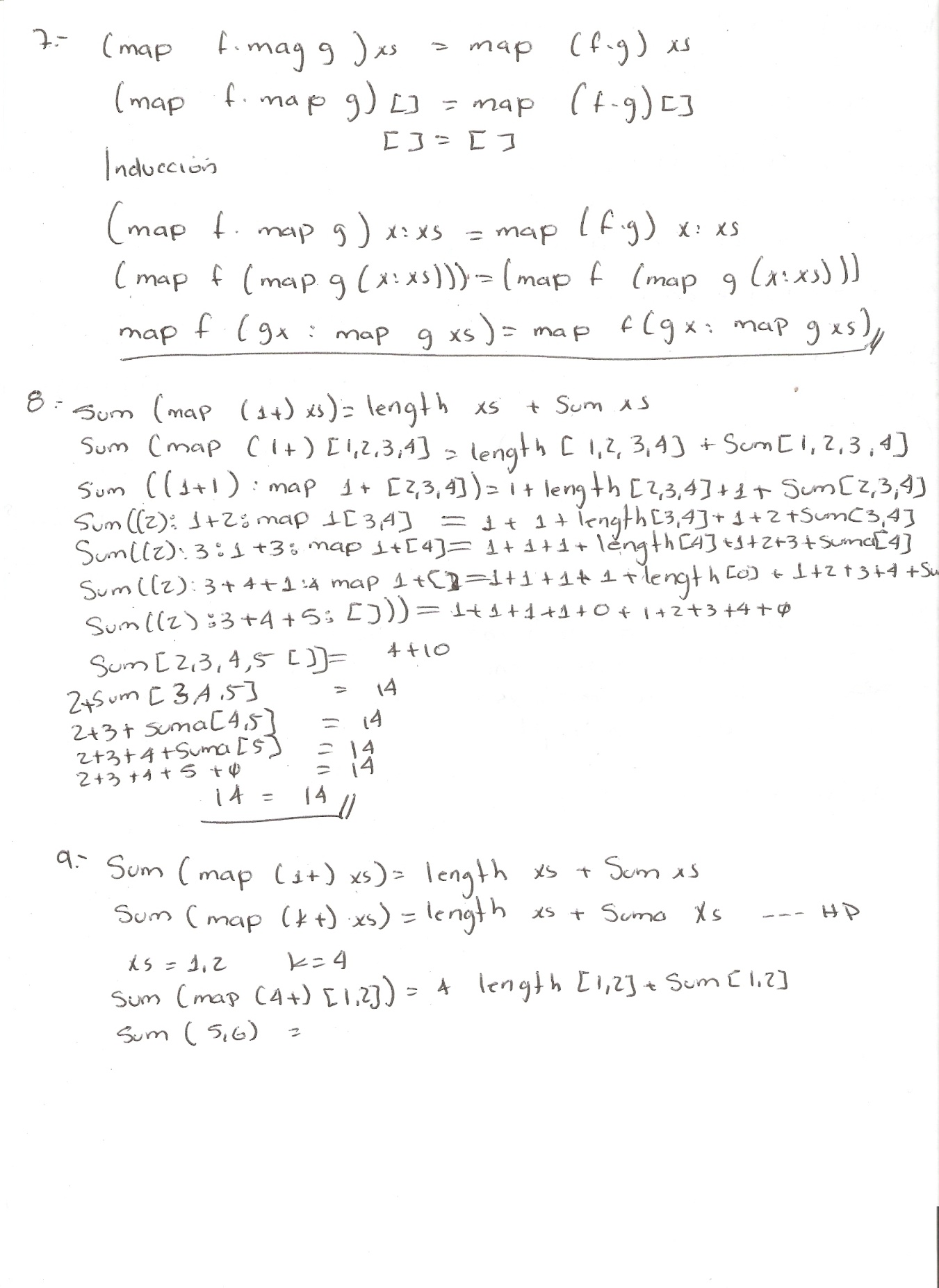
**Exercise 5.** Prove Theorem 16.



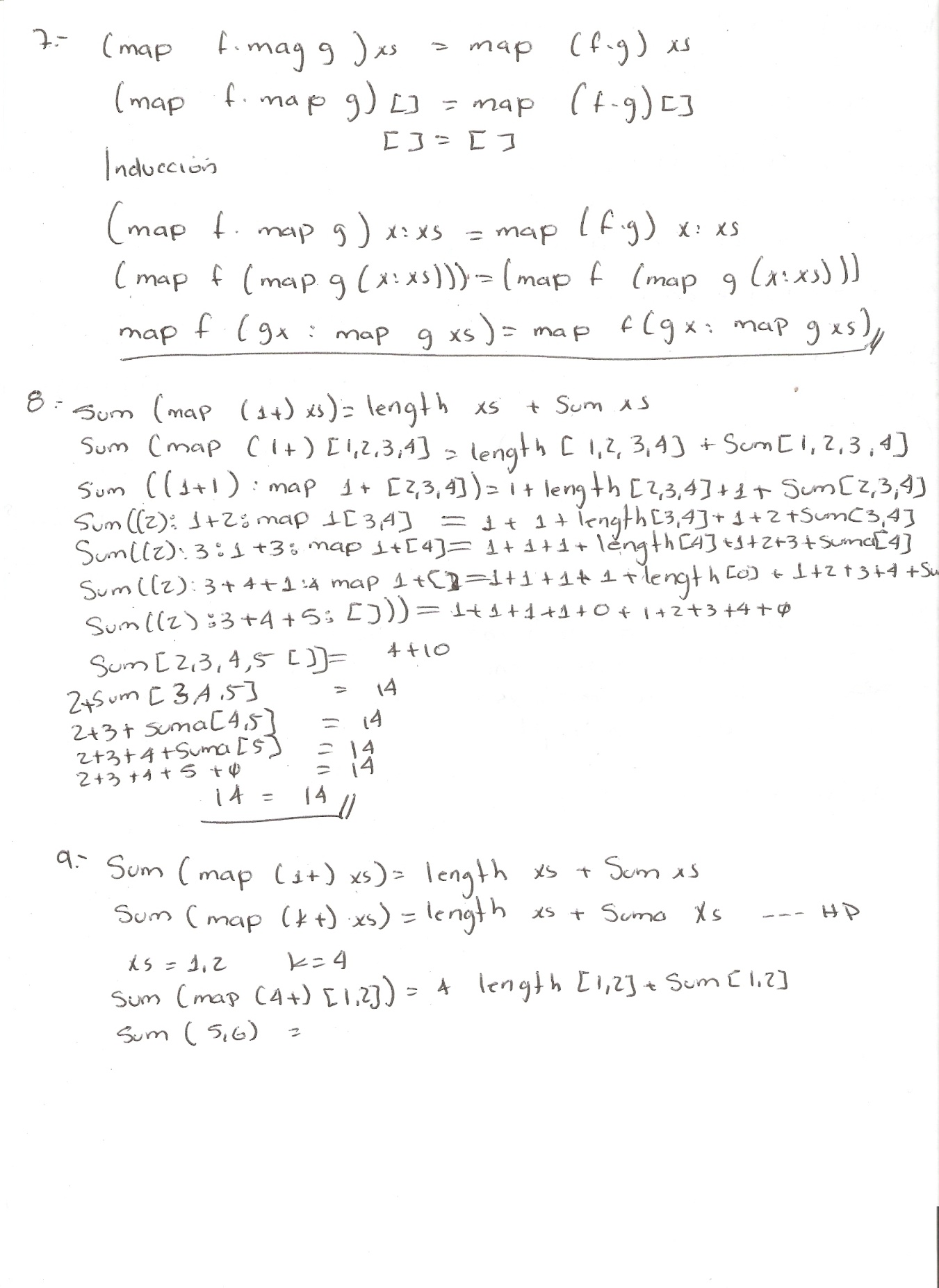
**Exercise 6.** Prove Theorem 18.

****

**Exercise 7.** Prove Theorem 19.



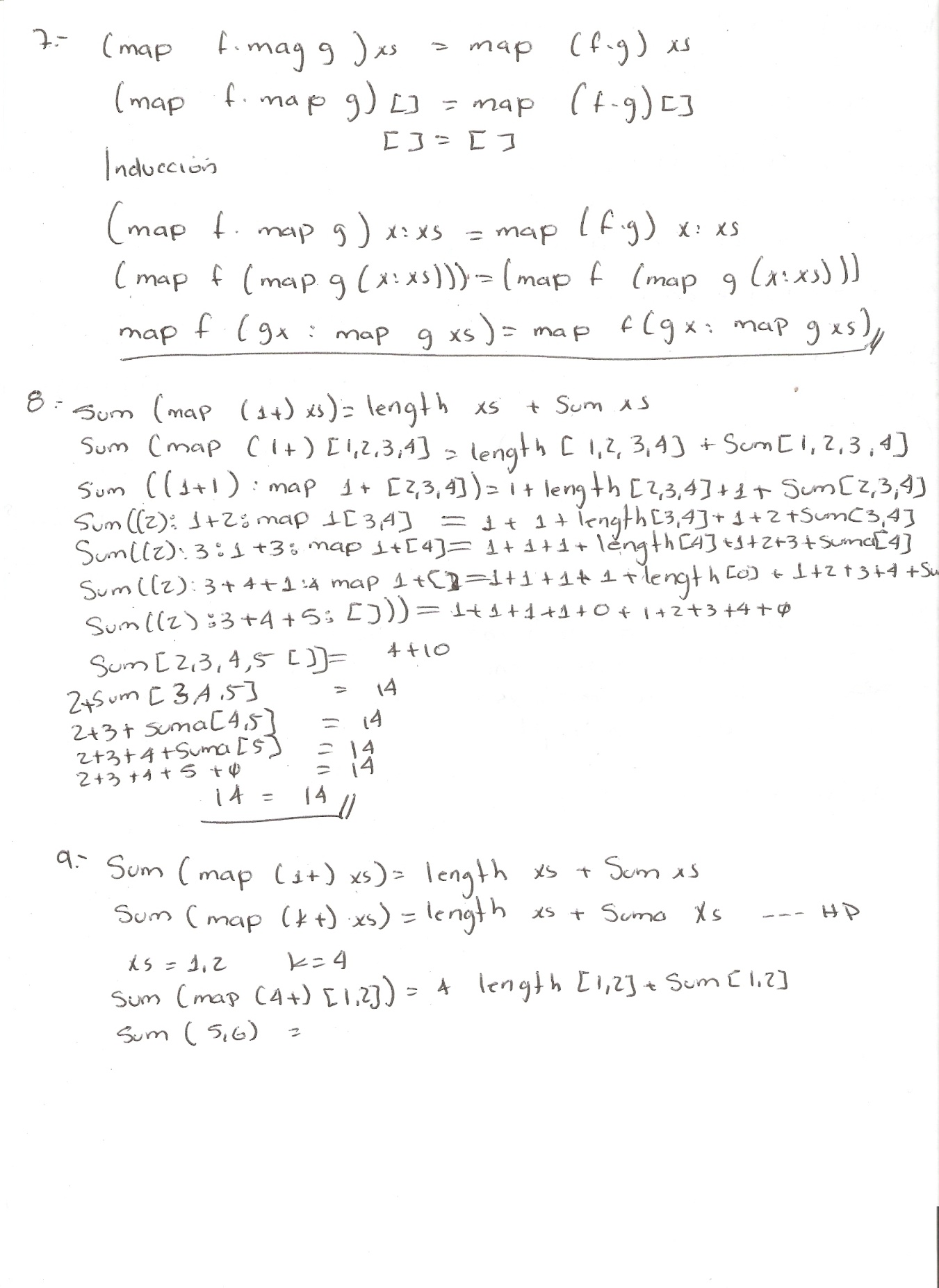
**Exercise 8.** Recall Theorem 20, which says

**

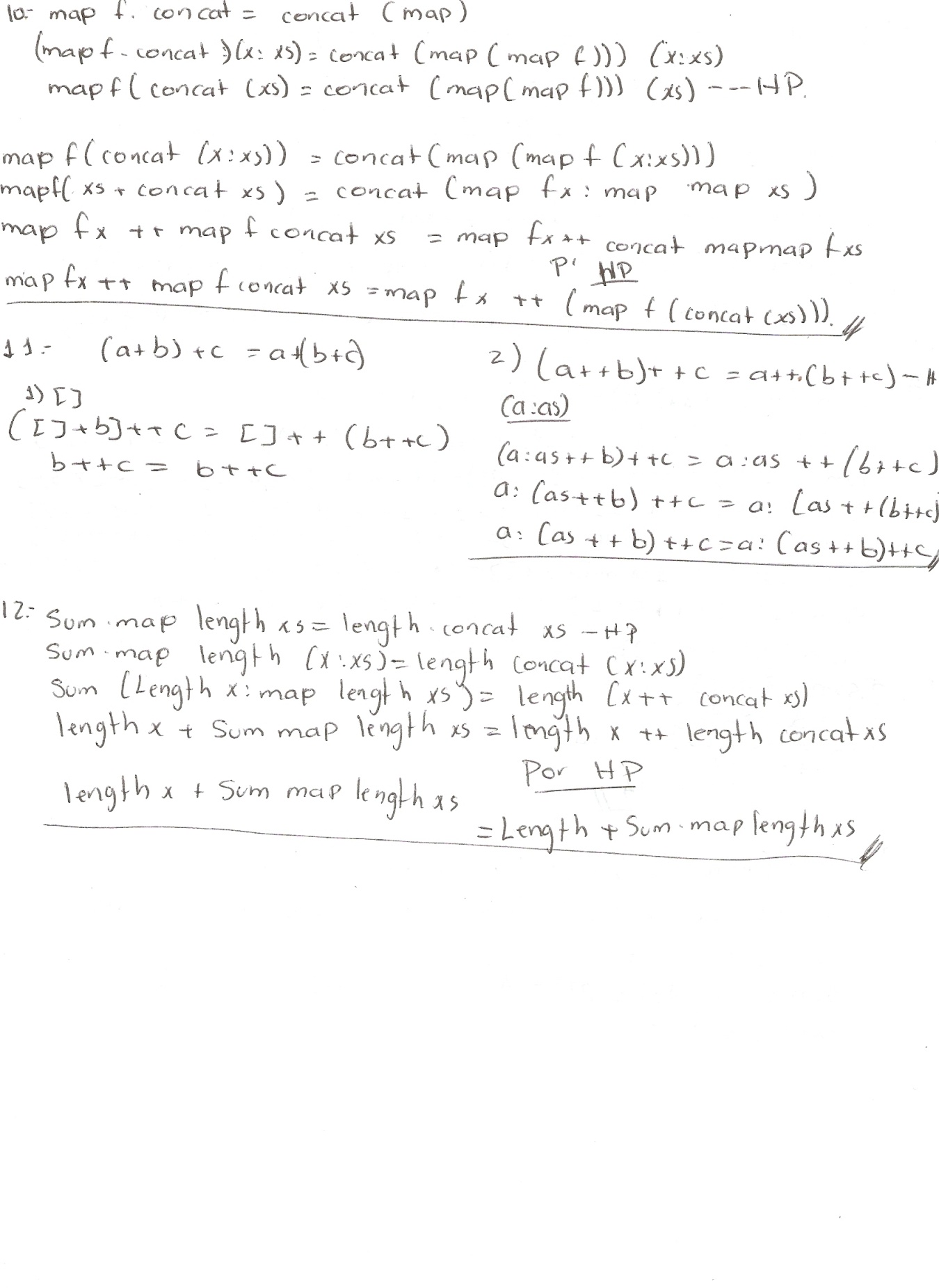
*sum* (*map* (1+) *xs*) = *length xs* + *sum xs.*

Explain in English what this theorem says. Using the definitions of the functions involved (*sum*, *length* and *map*), calculate the values of the left and right-hand sides of the equation using *xs* = [1*,* 2*,* 3*,* 4].

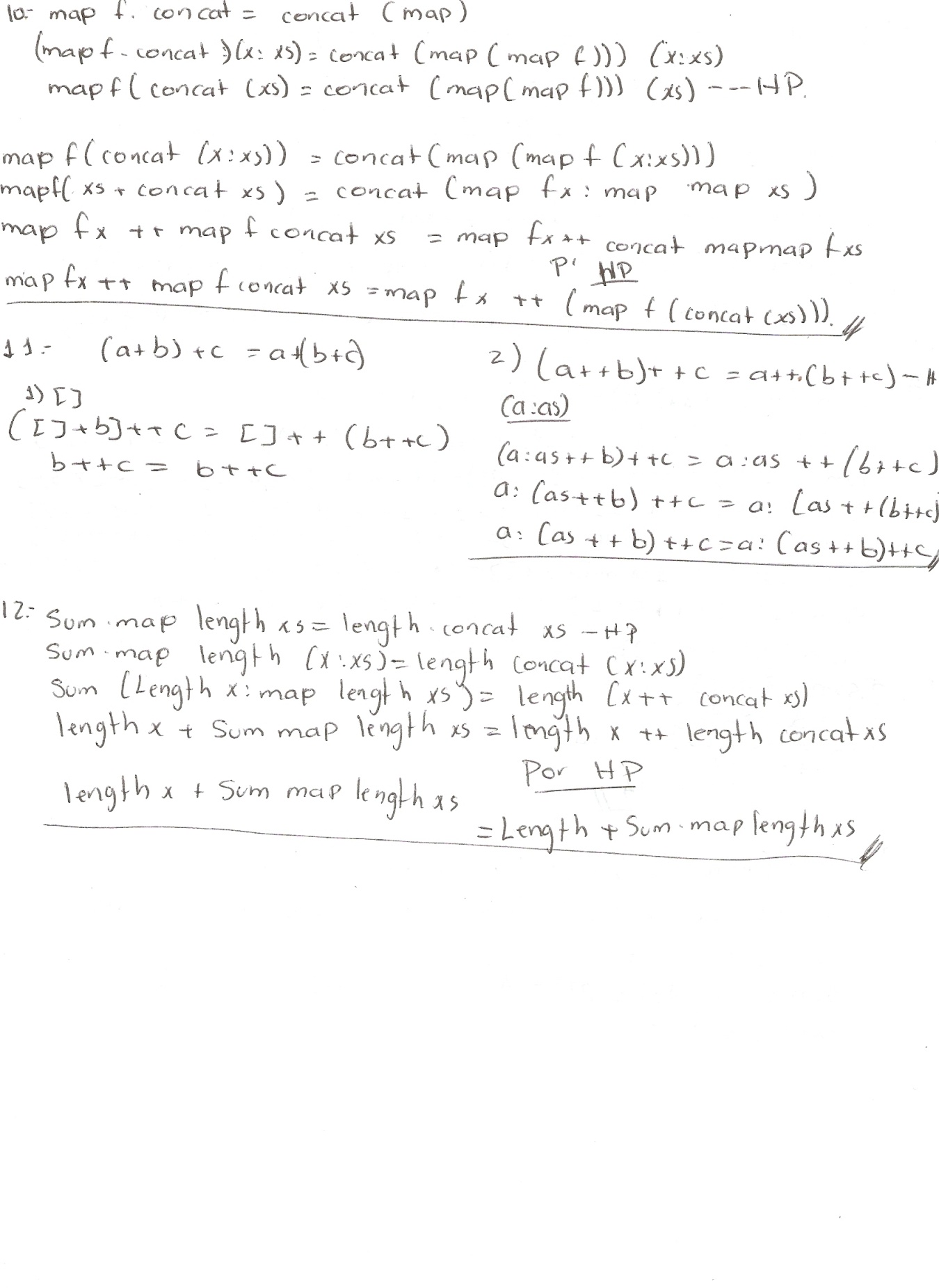
**Exercise 9.** Invent a new theorem similar to Theorem 20, where (1+) is replaced by (*k*+). Test it on one or two small examples. Then prove your theorem.



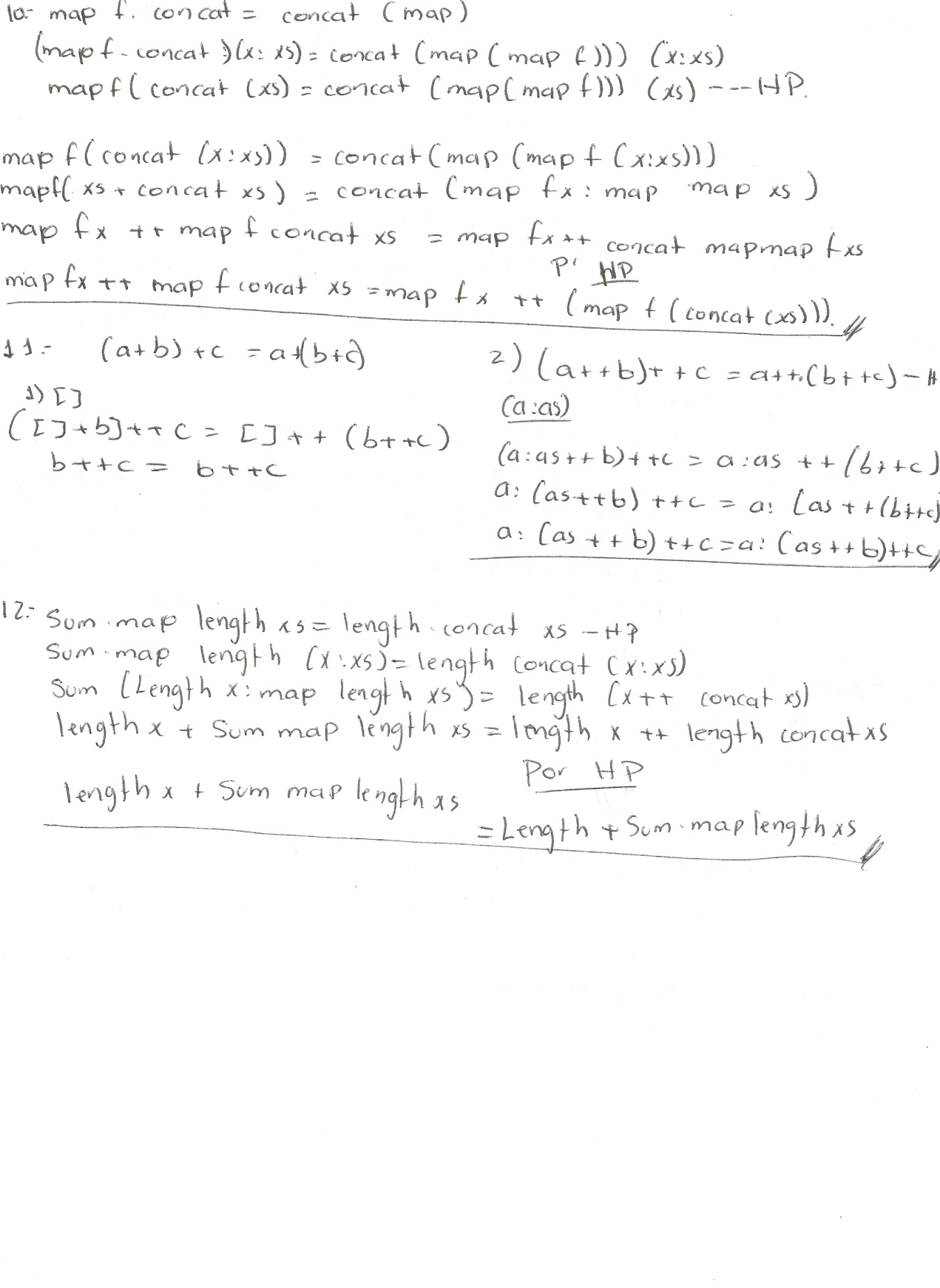
**Exercise 10.** Prove Theorem 25.



**Exercise 11.** Prove that the ++ operator is associative.

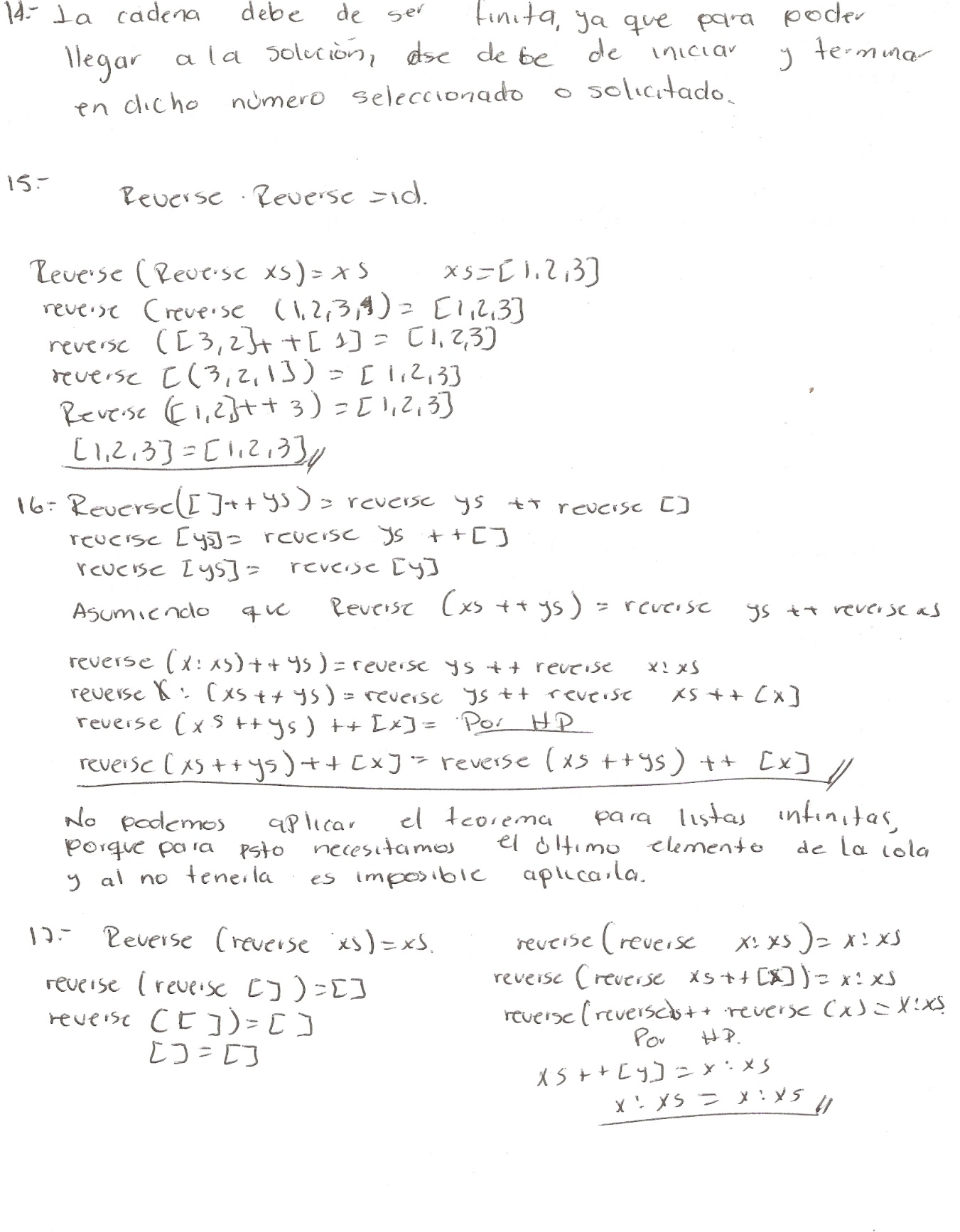
****

**Exercise 12.** Prove *sum . map length* = *length . concat*.

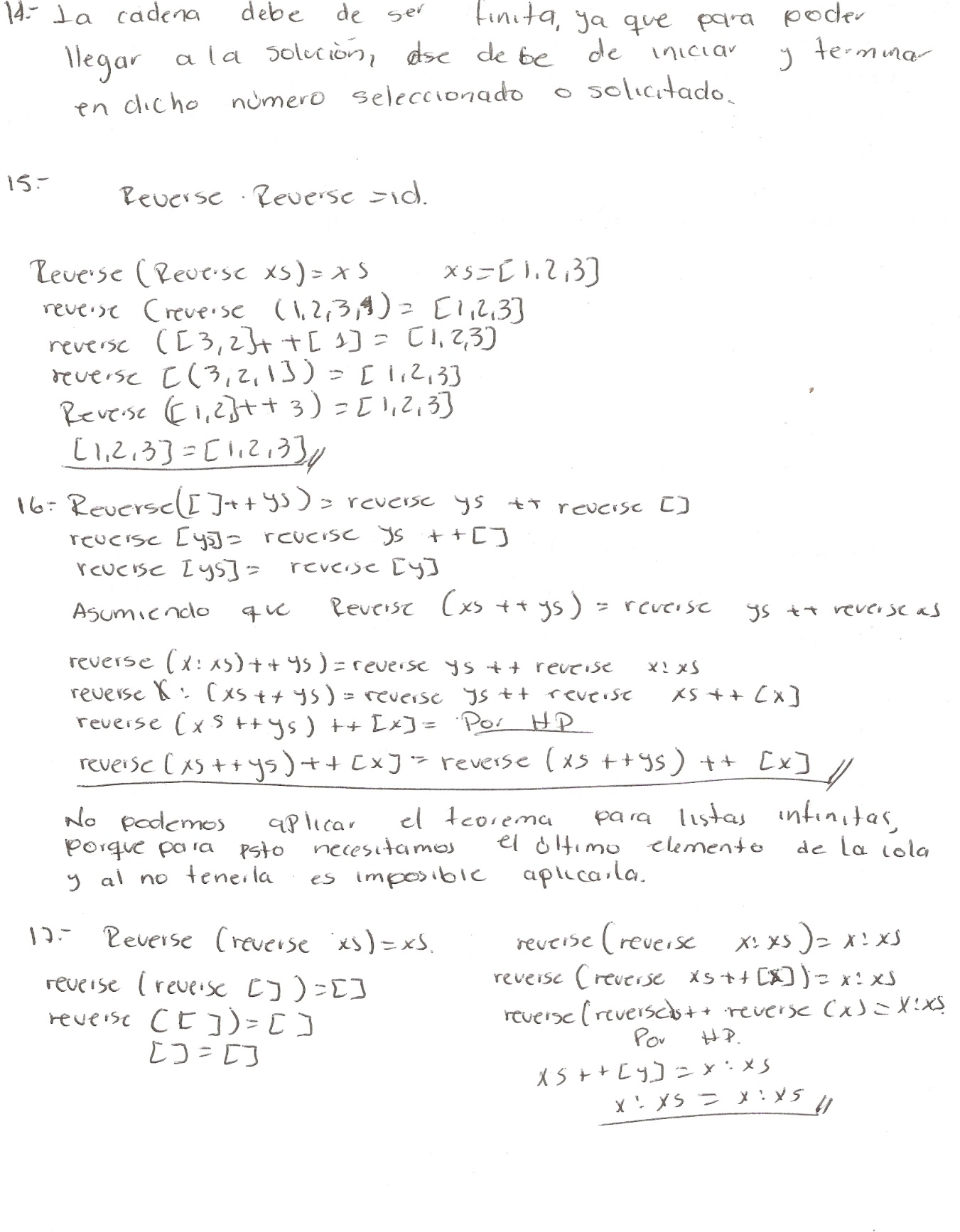


**Exercise 14.** State the requirements on finite length that the proof of *P* imposes on the arguments of concat, where *P* is defined as

*P*(*n*) *≡* concat xss = foldr (++) [] xss

****

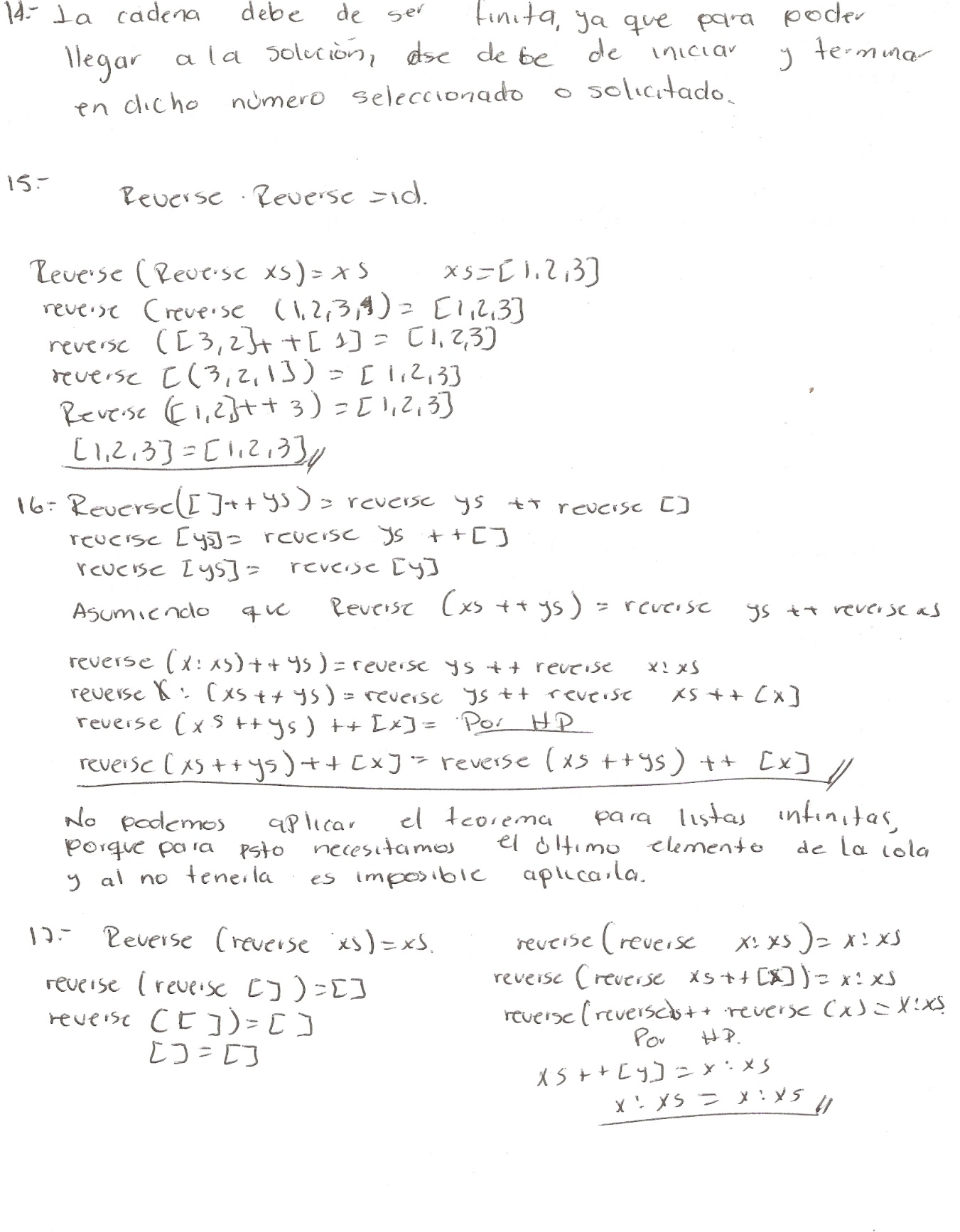
**Exercise 15.** Check that Theorem 27 holds for the argument [1*,* 2*,* 3].

****

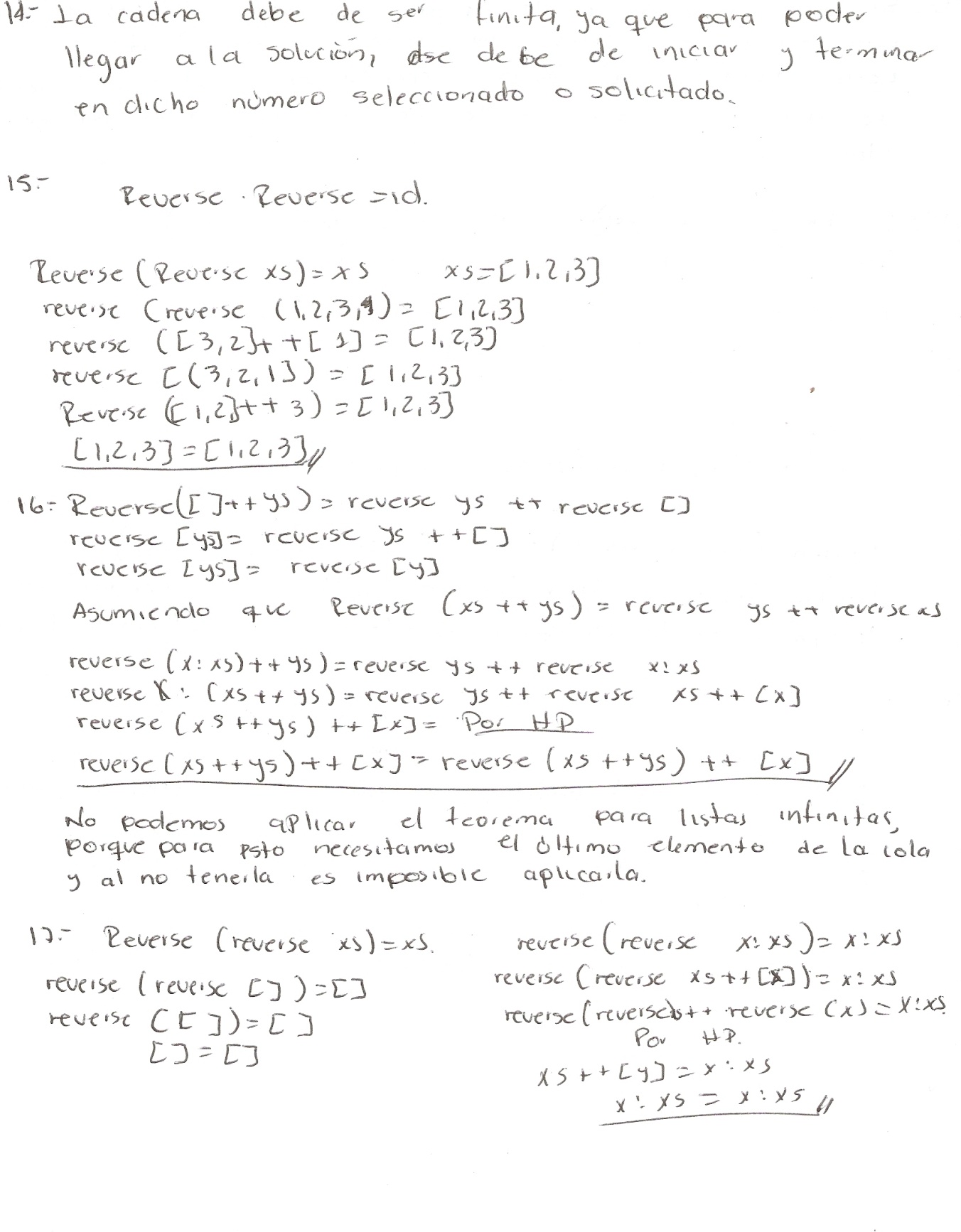
**Exercise 16.** Prove the following theorem, using induction:

*reverse* (*xs*++*ys*) = *reverse ys*++*reverse xs*

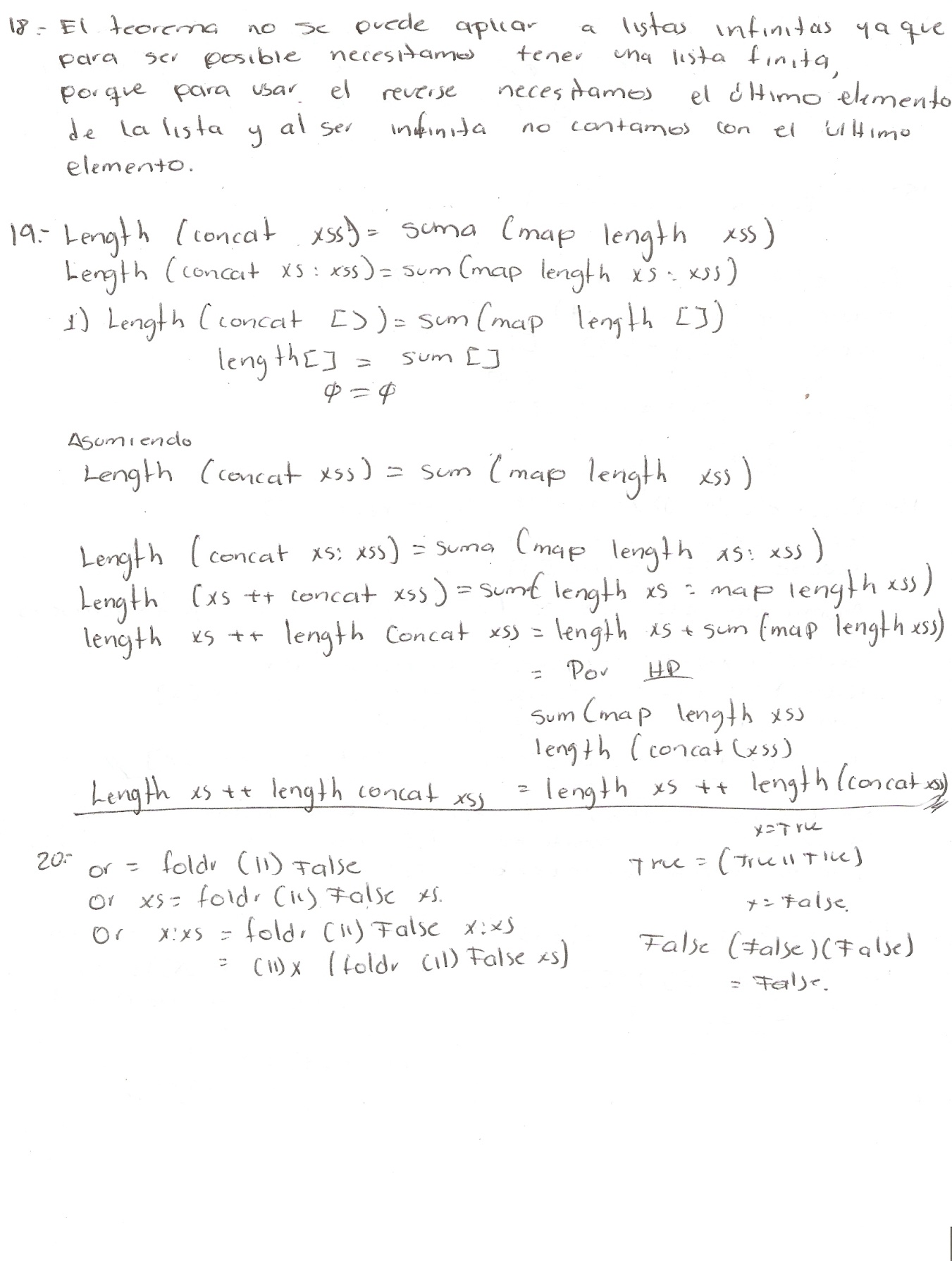
Then decide whether this theorem happens to be true for infinite lists like [1 *. .*]. Try to give a good argument for your conclusion, but you don’t have to prove it.

****

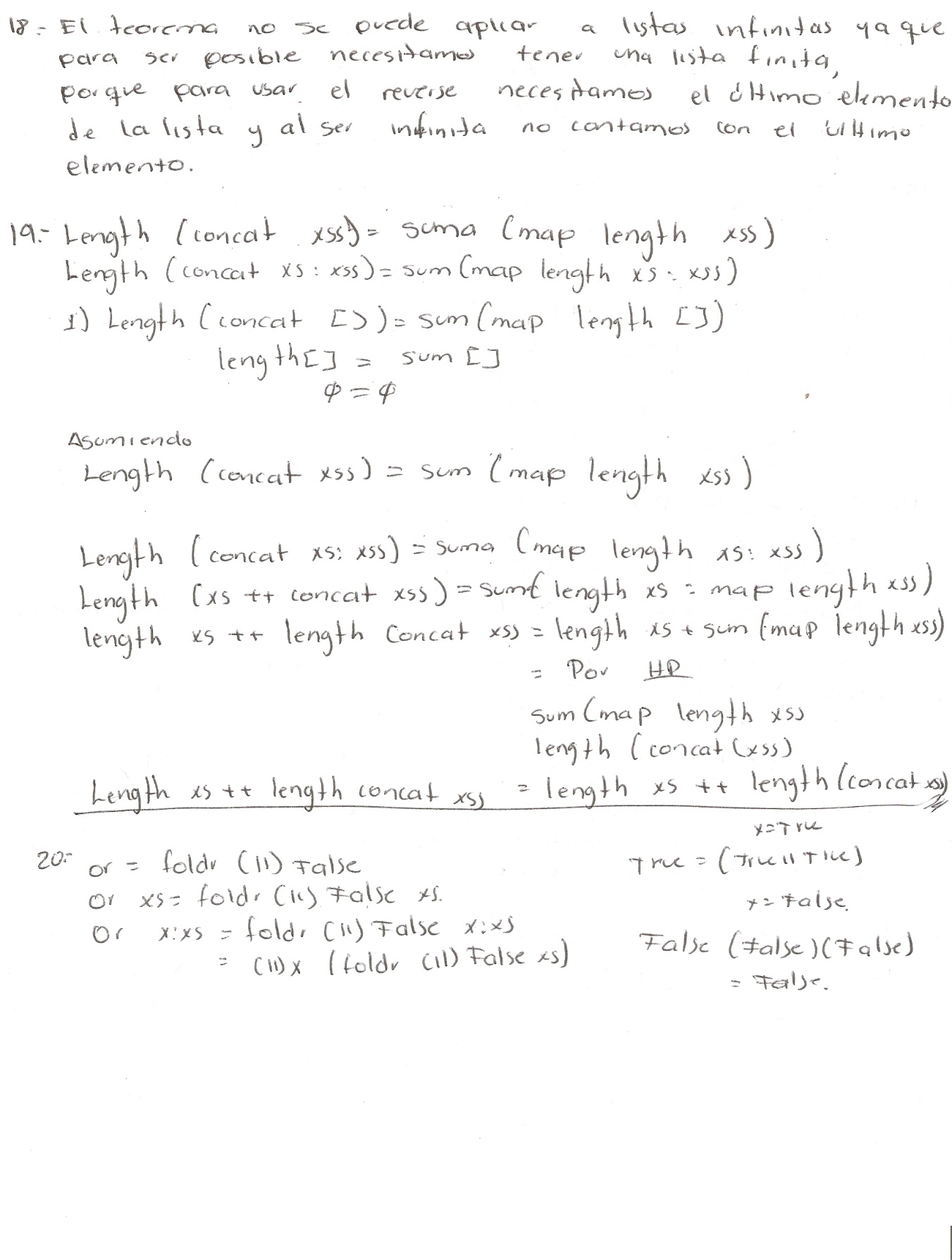
**Exercise 17.** Use induction to prove Theorem 27. *reverse* (*reverse xs*) = *xs*.

****

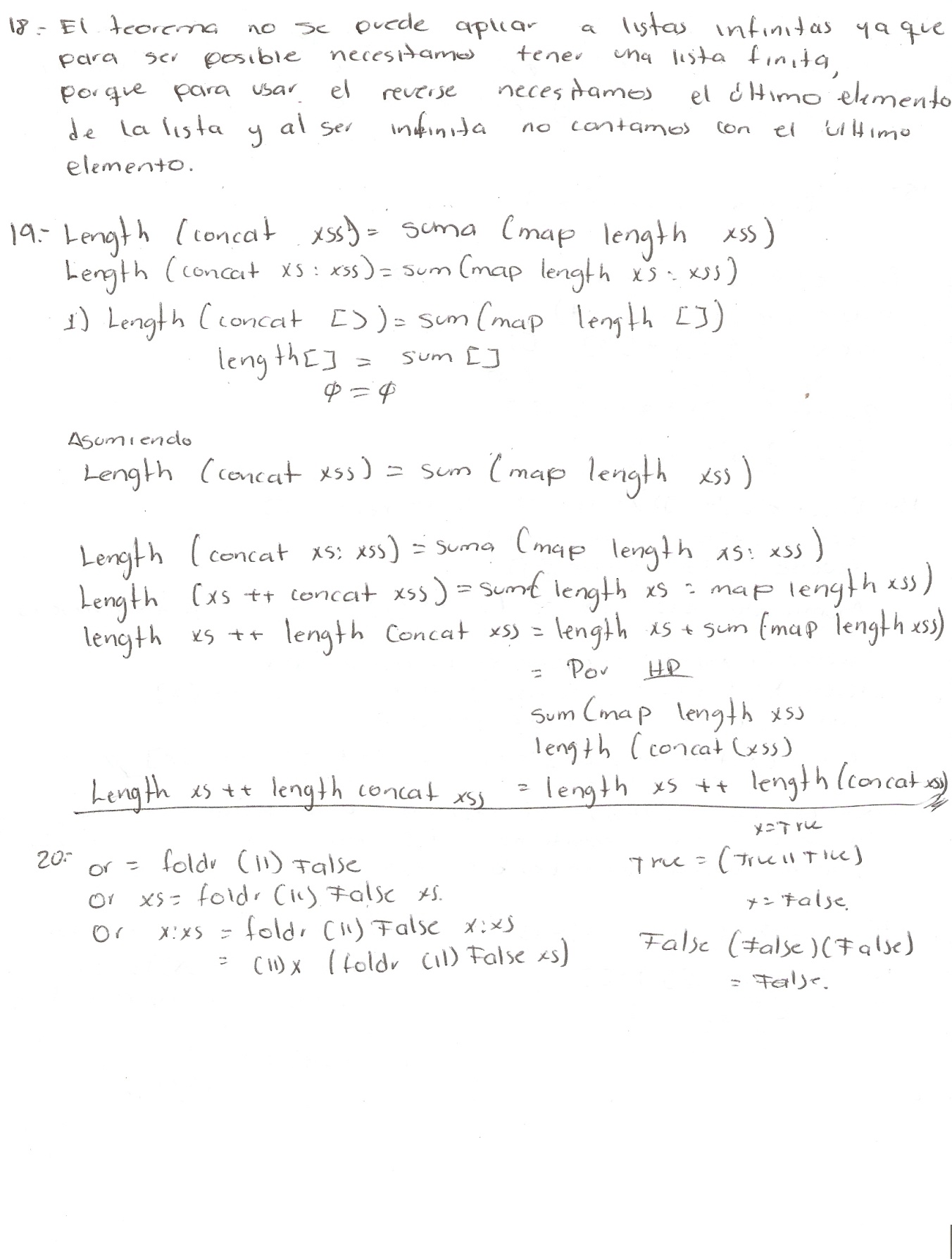
**Exercise 18.** Explain why Theorem 27 does not hold for infinite lists.



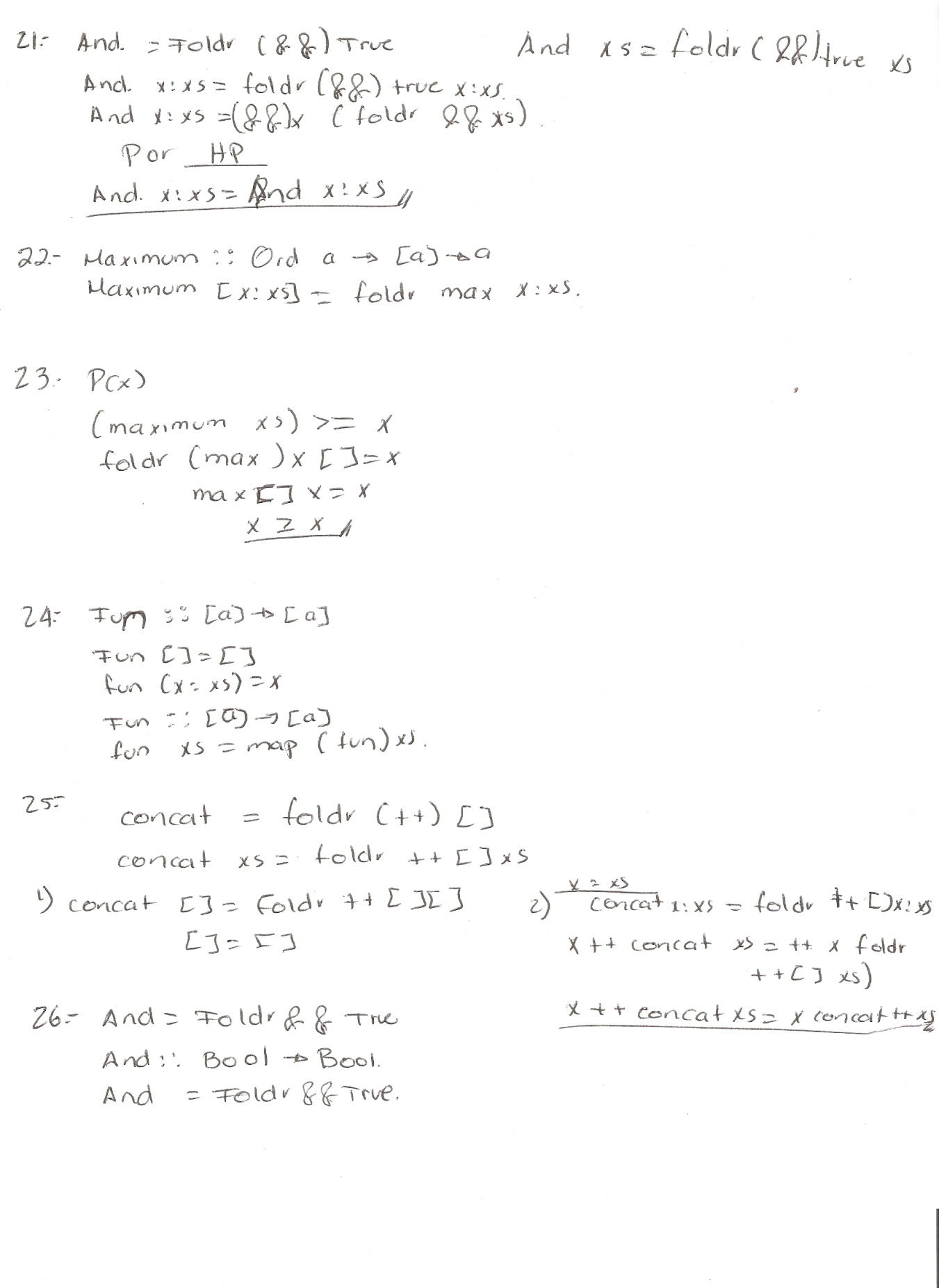
**Exercise 19.** Assume that xss is a finite list of type [[a]], that it is of length *n*, and that xs is a finite list and an arbitrary element of xss. Prove that length (concat xss) = sum (map length xss).



**Exercise 20.** Prove that or defined over an argument that has an arbitrary number of elements delivers the value True if True occurs as one of the elements of its argument.



**Exercise 21.** Prove that and defined over an argument that has an arbitrary number of elements delivers the value True if all of the elements in its argument are True.



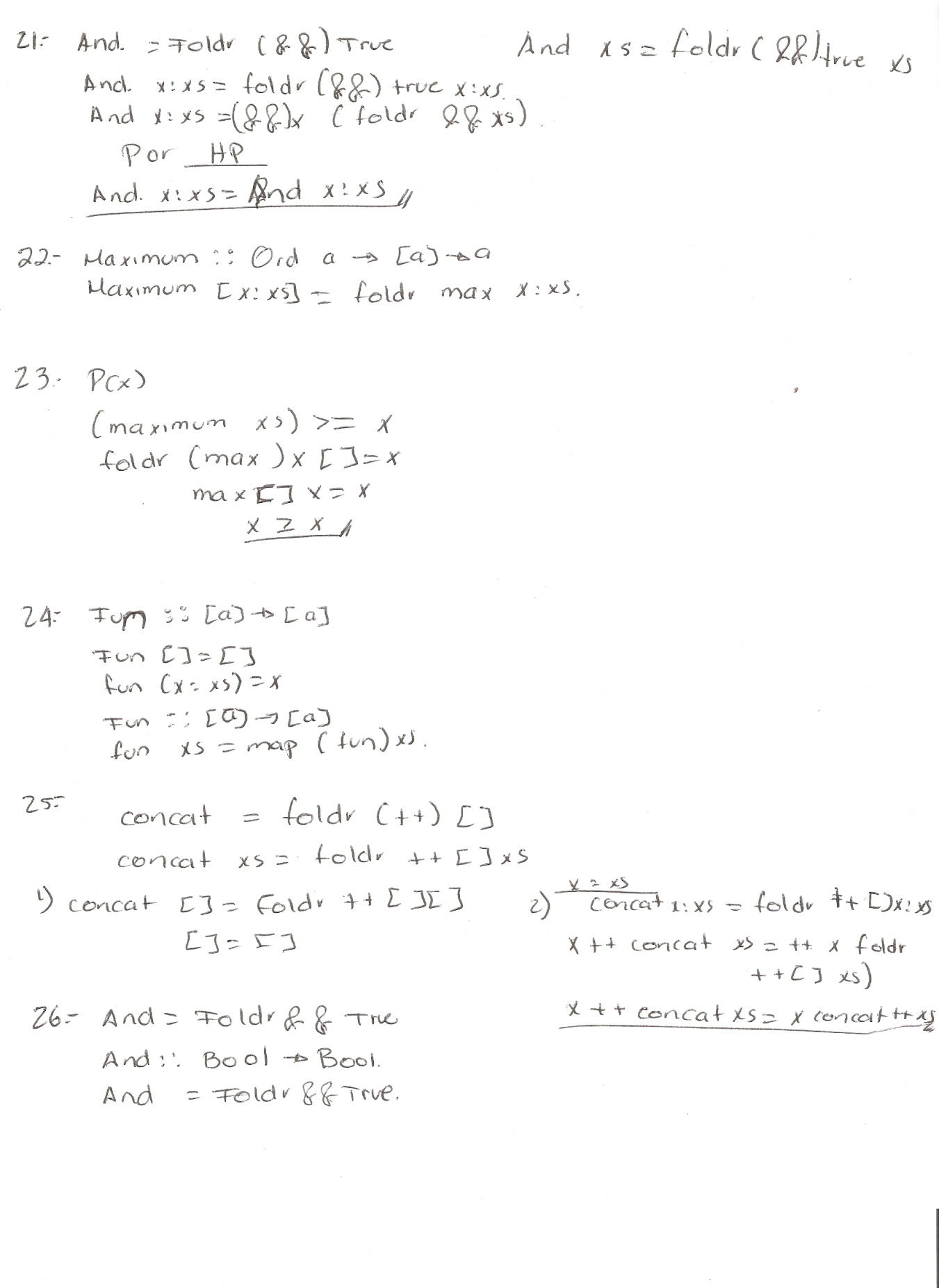
**Exercise 22.** Assume there is a function called max that delivers the larger of its two arguments.

max x y = x *if* x *>*= y

*and*

max x y = y *if* y *>*= x

Write a function maximum that, given a non-empty sequence of values whose sizes can be compared (that is, values from a type of class Ord), delivers the largest value in the sequence.

****

**Exercise 23.** Assume that the list xs is of type Ord a =*>* [a], and that x is

an arbitrary element of xs. Given the definition of maximum, defined as

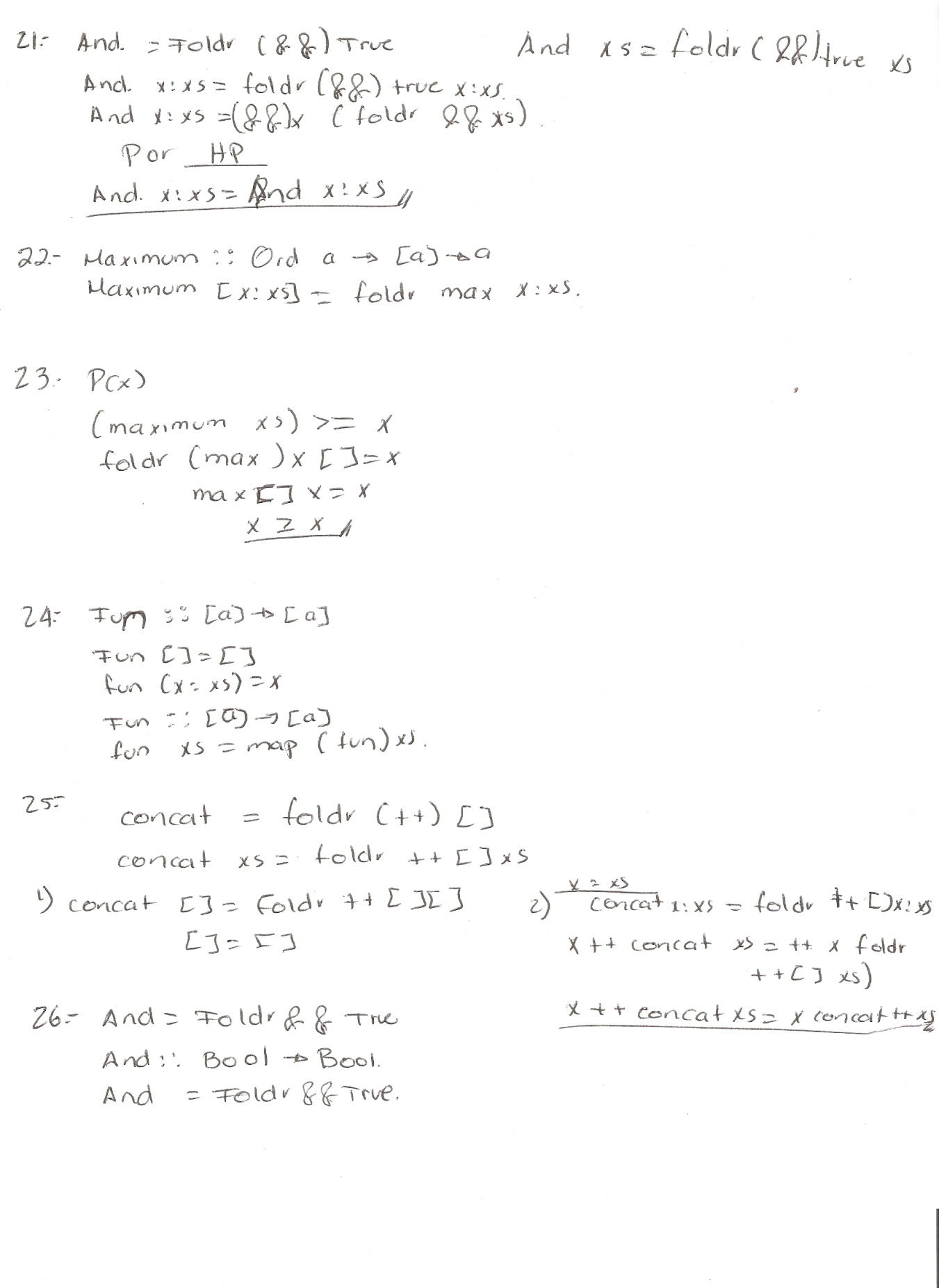
maximum :: [Ord] -> Ord

maximum xs = foldr (max) y ys

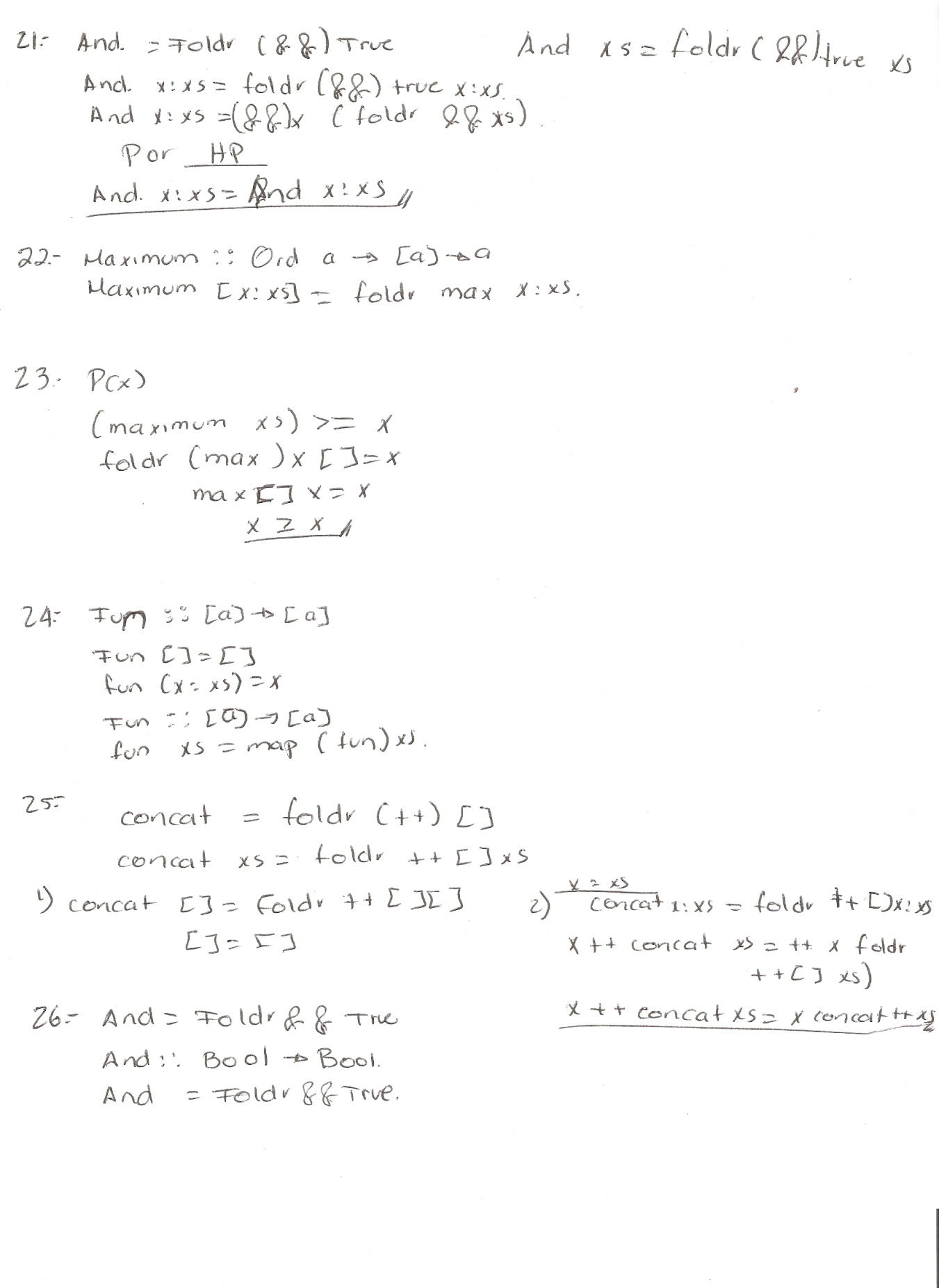
where xs = y:ys

prove that maximum has the following property:

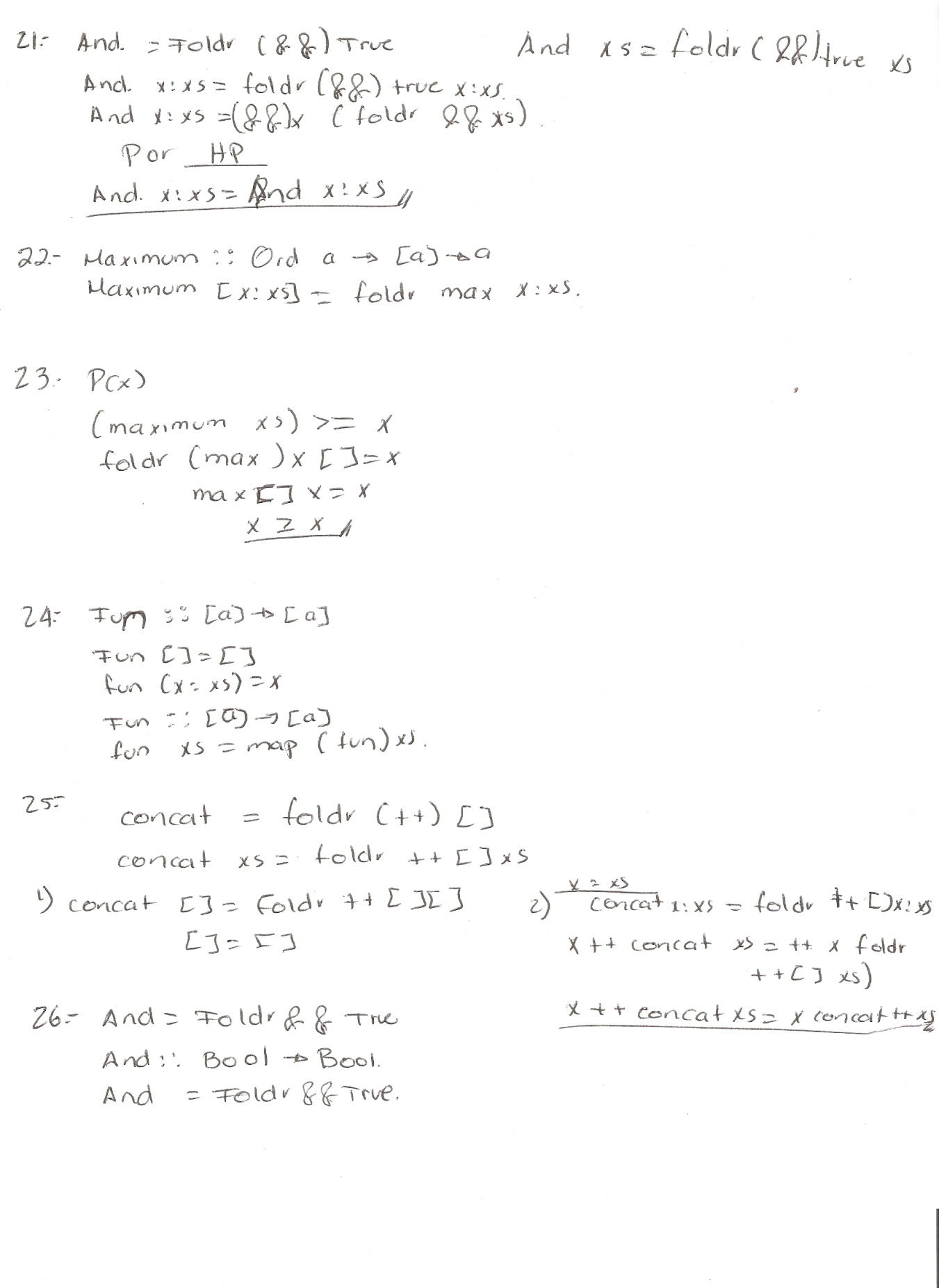
(maximum xs) *>*= x



**Exercise 24.** Write a function that, given a sequence containing only nonempty sequences, delivers the sequence made up of the first elements of each of those non-empty sequences.



**Exercise 25.** Prove the equation concat = foldr (++) []. Assume that the lists are finite, so that list induction can be used.



**Exercise 26.** Define an and operator using && and foldr.

**Exercise 27.** Given a list xs of type Bool, prove that and ([False] ++ xs) = False*.*

