Prove by Structural Induction that

#input = #output for the following program

```
k [ ] = [ ]
 k (e:es) = e*e : k es
```

Base case: #input => #[ ] => 0

#output => #(k [ ]) => #[ ] => 0

Base case proved

Inductive Step

```
   Assume the hypothesis  #input = #output.

That is #a = #(k a)

 Show:   #(e:a) = #(k (e:a))

 LHS #(e:a) => 1 + #a          (by def. of #)

 RHS #(k (e:a)) => #(e*e:(k a) (by def.of k)

                => 1 + #(k a)  (by def. of #)

                => 1 + #a     (by hypothesis)
   Since LHS = RHS, the theorem is proven.
```

2.
```
p   as     [ ]   =  [ ]
p  [ ]      bs   =  bs
p (a:as) (b:bs) =  a:p as bs, if a > b
                 =  b:p as bs, otherwise
```

Prove by Structural Induction (on the length of the list given as second input) that the length of the second input equals the length of the output from p. That is prove that

```
#(sec_list) = #(p x sec_list), for all x.
```

Base case: second_list = [ ]

Show: # [] = #(p x [])

LHS = #[ ] => 0 (by defn of #)

RHS = #(p x [ ])
=> #[ ] => 0 (by defn. of p and #)

Since LHS = RHS The Base case is proven

```
Inductive Step

Assume #second_list = #output_list
                   for some second list k

    that is #k = #(p  first_list  k)


Show:  #(e:k) = #(p  first_list (e:k) )

Begin with LHS = #(e:k)
               => 1 + #k  ( by defn. of #)
```

Now we look at the three cases for the RHS

Case 1: length of first list = [ ]

```
 RHS =  #(p [ ]  (e:k) )

     => #(e:k)     (by def. of p)

     => 1 + #k     (by def. of #)

     =  LHS Since RHS = LHS, case 1 is proven
```

<u>Case 2: (y:ys) is not empty and y > e</u>

```
RHS = #(p  (y:ys)  (e:k) )
=> #(y : p ys  k)    (by def. of p)

=> 1 + #(p  ys k)    (by defn of #)

=> 1 + #k            (by hypothesis)

= LHS   Since RHS = LHS, case 2 is proven
```

Case 3: y:ys is not empty and ~(y > e)
(the otherwise case)

```
RHS = #(p  (y:ys)  (e:k) )

=> #(y : p ys  k)   (by def. of p)

=> 1 + #( p ys  k)  (by defn of #)

=> 1 + #k           (by hypothesis)

= LHS  Since RHS = LHS case 3 is proven
```

Since all cases are proven for the inductive step, the proof is complete.

3. Show by structural induction that (++) is associative, i.e. that

(a++b)++c = a++(b++c)

Base case: a = [ ]

LHS = ([ ]++b)++c = b++c

RHS = [ ] ++(b++c) = b++c LHS = RHS; base case proved

Inductive step

Assume the hypothesis (x ++ b) ++ c = x ++ (b ++ c)

Show: ((e:x) ++ b) ++ c = (e:x) ++ (b ++ c)

How LHS = ((e:x) ++ b) ++c (def. of ++)

=> (e : (x ++ b)) ++ c (def. of ++)

=> e : ((x ++ b) ++ c) (by hypothesis)

=> e : (x ++ (b ++ c))

=> (e : x) ++ (b ++c)

= RHS Therefore, the theorem is proved.

4.Prove by structural induction that the number of elements in the input is less than the number of elements of the output from f:

f [ ] = [3]
f (e:es) = e * 8 : f es

Base case: input is [ ] therefore #input = 0
output is f [ ]
#(f [ ]) => #[3] from definition
=> 1
Therefore #input < #output; **base case proved**

Inductive Step

Assume

Hypothesis is #as < #(f as) where as is the input

Show : #(e:as) < #(f (e:as))

```
LHS = #(e:as)
    = 1 + # as

RHS = # (f (e:as))            (by def. of f)
    => # (e *8 : f as)        (by def. of #)
    => 1 + # (f as)           (by hypothesis)
    > 1 + # as
    > LHS
```

Since LHS < RHS, the theorem is proved.