2. Dada una constante natural N, implementá el TAD Lista de elementos de tipo T, usando un arreglo de tamaño N y un natural que indica cuántos elementos del arreglo son ocupados por elementos de la lista. ¿Esta implementación impone nuevas restricciones? ¿En qué función o procedimiento tenemos una nueva precondición?

```
implement List of T where

type List of T = tuple
                   elems: array[1..N] of T
                   size: nat
                 end tuple

constructors
    fun empty_list() ret l: List of T
        l.size := 0
    end fun

    {- PRE: l.size < N -}
    proc addl(in/out l: List of T, in e: T)
        for i := l.size downto 1 do
            l.elems[i+1] := l.elems[i]
        od
        l.elems[1] := e
        l.size := l.size + 1
    end proc

destroy
    proc destroy_list(in/out l: List of T)
        //skip
    end proc

operations
    fun is_empty_list(l: List of T) ret b: bool
        b := (l.size = 0)
    end fun

    {- PRE: not is_empty_list(l) -}
    fun head(l: List of T) ret e: T
        e := l.elems[1]
    end fun

    {- PRE: not is_empty_list(l) -}
    proc tail(in/out l: List of T)
        for i := 2 to l.size do
            l.elems[i-1] := l.elems[i]
        od
        l.size := l.size - 1
    end proc

    {- PRE: l.size < N -}
    proc addr(in/out l: List of T, e: T)
        l.size := l.size + 1
        l.elems[l.size] := e
    end proc

    fun length(l: List of T) ret n: nat
        n := l.size
    end fun
```

```
    {- PRE: l.size + l0.size < N -}
    proc concat(in/out l: List of T, in l0: List of T)
        for i := l.size+1 to l.size + l0.size do
            l.elems[i] := l0.elem[i-l.size]
        od
        l.size := l.size + l0.size
    end proc

    {- PRE: length(l) > n -}
    fun index(l: List of T, n: nat) ret e: T
        e := l.elems[n]
    end fun

    proc take(in/out l: List of T, in n: nat)
        l.size := n
    end proc

    {- PRE: l.size > n -}
    proc drop(in/out l: List of T, in n: nat)
        for i := n+1 to l.size do
            l.elems[i-n] := l.elems[i]
        od
        l.size := l.size - n
    end proc

    fun copy_list(l1: List of T) ret l2: List of T
        for i := 1 to l1.size do
            l2.elems[i] := l1.elems[i]
        od
        l2.size := l1.size
    end fun

end implement
```