1) – 2)

a)

proc nombre (out a : array[1..n] of nat)

var i : nat

i := 1

do(i<=n)

a[i] := 0

i := i+1

od

\\\\\\\\\\\\\\\

for i:=1 to n do

a[i] := 0

od

end proc

b)

proc nombre (out a : array[1..n] of nat)

var i : nat

i := 1

do(i<=n)

a[i] := i

i := i+1

od

\\\\\\\\\\\\\\

for i:=1 to n do

a[i] := i

od

end proc

c)

proc nombre (out a : array[1..n] of nat)

var i : nat

i := 1

do(i<=n)

a[i] := (2\*i)-1

i := i+1

od

\\\\\\\\\\\\

for i:=1 to n do

a[i] := (2\*i)-1

od

end proc

d)

proc nombre (in/out a : array[1..n] of nat)

var i : nat

i := 1

do(i<=n)

a[i] := a[i]+1

i := i+2

od

\\\\\\\\\\\\

for i:=1 to n do

if(i mod 2 = 1) then a[i] := a[i]+1 fi

od

end proc

3)

3)

fun ord (in a : array[1..n]) ret res : bool

var upper : bool

var lower : bool

upper := true

lower := true

for i:=1 to (n-1) do

if(a[i] > a[i+1]) then lower := false

fi

if(a[i] < a[i+1]) then upper := false

fi

od

res := upper ∨ lower

end fun

El algoritmo recorre el arreglo desde 1 hasta n-1, y analiza si cada uno de los elementos es menor o igual al siguiente, o, si cada uno de los elementos es mayor o igual al siguiente.

Lo hace mediante el uso de dos guardas (booleanos) las cuales usando dos if, controla que no se de el caso contrario a las condiciones para que el arreglo esté ordenado, ya sea de mayor a menor o viceversa.

4) que paja ameo xd

5) a) n^6 ; b) n! \* 3

6) La función f devuelve la posición del mayor elemento del arreglo “array” desde 1 hasta i inclusive.

Sabiendo lo anterior, el procedimiento p ordena los elementos del arreglo array de menor a mayor, dejando el menor en la posición 1 y al mayor en la posición n.

Que paja reescribirlo bro xd

7) Mas paja jasdjsa