1. (Algoritmo de la división) Dados dos números, hay que encontrar el cociente y el resto de la división entera entre ellos.

   Ayuda: Para enteros $x \geq 0$ e $y > 0$, el cociente q y el resto r de la división entera de x por y están caracterizados por $x = q * y + r \wedge 0 \leq r \wedge r < y$. Por lo tanto, debemos derivar un programa S que satisfaga

   Const $x, y : Int$;
   Var $q, r : Int$;
   $\{P : \quad x \geq 0 \wedge y > 0\}$ (precondición)
   S
   $\{Q : \quad x = q * y + r \wedge 0 \leq r \wedge r < y\}$ (postcondición)

Const x, y : Int;
Var q, r : Int;
{P : x ≥ 0 ∧ y > 0} (precondición)
S
{Q : x = q ∗ y + r ∧ 0 ≤ r ∧ r < y} (postcondition)

Como tenemos una conjunción en Q, podemos generar una invariante tomando una parte como invariante y otra como guarda tq se cumpla INV ^ ¬B → Q
Probemos con:
INV ≡ (x = q * y + r ^ 0 ≤ r)
B ≡ (r >= y)
¿Vale INV ^ ¬B → Q? Veamos.
(x = q * y + r ^ 0 ≤ r) ^ ¬(r >= y) → (x = q ∗ y + r ∧ 0 ≤ r ∧ r < y)
≡{ negación
(x = q * y + r ^ 0 ≤ r) ^ (r < y) → (x = q ∗ y + r ∧ 0 ≤ r ∧ r < y)
≡{ asumiendo la parte izquierda
true

Luego tenemos
Const x, y : Int;
Var q, r : Int;
{P : x ≥ 0 ∧ y > 0} (precondición)
S1
{INV : x = q * y + r ^ 0 ≤ r}
do (r >= y) →
    S2
od
{Q : x = q ∗ y + r ∧ 0 ≤ r ∧ r < y} (postcondition)

1)Veamos la inicialización
proponemos que sea una asignación:
S1 ≡ q,r := E,F

asumimos P : x ≥ 0 ∧ y > 0
wp.(q,r := E,F).(x = q * y + r ^ 0 ≤ r)
≡{ def de wp
x = E * y + F ^ 0 ≤ F
≡{ elegimos E = 0, aritmetica
x = F ^ 0 ≤ F
≡{ lógica, hipótesis
true

S1 ≡ q,r := 0,x

2) definimos una cota, nos interesa que r descrezca para que el ciclo termine.
vale [INV ^ B → r >= 0]
Veamos:

x = q * y + r ^ 0 ≤ r ^ r >= y → r >= 0
≡{ asumimos la primer parte
r >= 0
≡{ hipótesis
true

3) Cuerpo del ciclo, ¿Cómo decrece la cota? proponemos una asignación
S2 ≡ q,r := E, r-F

para eso demostremos que el invariante es invariante
{INV ^ B} S2 {INV}

asumimos INV : x = q * y + r ^ 0 ≤ r ^ r >= y
wp.(q,r := E, r-F).(x = q ∗ y + r ∧ 0 ≤ r)
≡{ def de wp
x = E ∗ y + r-F ∧ 0 ≤ r-F
≡{ x = q * y + r
q * y + r = E ∗ y + r - F ∧ 0 ≤ r-F
≡{aritmética
q * y + r = E ∗ y + (r - F) ∧ F ≤ r
≡{ con F = y por hipótesis
q * y + r = E ∗ y + (r - y)
≡{ aritmética
q * y = (E ∗ y)
≡{ aritmética
q * y - y = E * y
≡{ aritmética
y (q + 1) = E * y
≡{ aritmética
q + 1 = E
≡{ con E = q + 1
true

Finalmente
Const x, y : Int;
Var q, r : Int;
{P : x ≥ 0 ∧ y > 0} (precondición)
  q,r := 0,x
{INV : x = q * y + r ^ 0 ≤ r}
do (r >= y) →
  q,r := q+1, r-y
od
{Q : x = q ∗ y + r ∧ 0 ≤ r ∧ r < y} (postcondition)

2. Sea $N \geq 0$, especificar y derivar un programa que calcule el menor entero x que satisface $x^3 + x \geq N$.

   Ayuda: Especifique el problema (con pre y poscondición) de forma que en la poscondición queden conjunciones y asi poder utilizar la técnica "tomar término de la conjunción". Para ellos fijarse como se hace esto al especificar el Ejemplo 19.2 del libro.

#Tenemos f.x = x^3 + x >= N
#Como precondición tenemos {P: ⟨ ∃ n : 0 <= n : n^3 + n >= N ⟩ ^ N >= 0}
- existe un estado inicial que cumple con f.n
- N es natural
#Como postcondición tenemos {Q: 0 <= x ^ (x^3 + x >= N) ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ }
- x es natural
- se cumple f.x
- todo i natural menor que x, no satisface f.i. Por lo que, x resulta ser el mínimo natural i que cumpla f.i

#Entonces tenemos:
Const N: Int;
var x: Int;
{P: ⟨ ∃ n : 0 <= n : n^3 + n >= N ⟩ ^ N >= 0}
S
{Q: 0 <= x ^ (x^3 + x >= N) ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ }

(x^3 + x >= N) no puede ser invariante porque tendría que valer al inicio del ciclo.
Probemos con el invariante {INV: 0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ } y la guarda B ≡ ¬f.x ≡ (x^3 + x < N)
#¿Se cumple INV ^ ¬B → Q? veamos.
0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ ^ x^3 + x >= N → 0 <= x ^ (x^3 + x >= N) ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩
≡{ asumimos 0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ ^ x^3 + x >= N
0 <= x ^ (x^3 + x >= N) ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩
≡{ por hipótesis
true

#Inicialización la única variable que tenemos x := E para eso debemos demostrar que P → wp.(x := E).(INV) veamos:
asumimos P ≡ ⟨ ∃ n : 0 <= n : n^3 + n >= N ⟩ ^ N >= 0
wp.(x := E).(0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩)
≡{ def de wp
0 <= E ^ ⟨∀i: 0 <= i < E : i^3 + i < N⟩
≡{ observamos que esto es true si forzamos un rango vacío con E=0
0 <= 0 ^ ⟨∀i: 0 <= i < 0 : i^3 + i < N⟩
≡{ rango vacío y lógica
true

hasta ahora tenemos:

Const N: Int;
var x: Int;
{P: ⟨ ∃ n : 0 <= n : n^3 + n >= N⟩ ^ N >= 0}
x := 0
{INV: 0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ }
do (x^3 + x < N) →
    S2
od
{Q: 0 <= x ^ (x^3 + x >= N) ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ }

#Como lo único que sabemos de f.x es que vale para algún N, por lo que el candidato a cota es t.x = N - x
¿vale INV ^ B → x > 0? veamos.
0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ ^ x^3 + x < N → x > 0
≡{ asumiendo INV ^ B
true
#Ahora, incrementando x decrece la cota. Proponemos la asignación x := x + k y para ver cuál es k probemos si el invariante es invariante INV ∧ B ⇒ wp.(x := x + k).INV

asumimos INV ^ B ≡ 0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ ^ x^3 + x < N

wp.(x := x + k). 0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩
≡{ def de wp
0 <= x + k ^ ⟨∀i: 0 <= i < x + k : i^3 + i < N⟩
≡{ partición de rango
0 <= x + k ^ ⟨∀i: 0 <= i < x + k : i^3 + i < N⟩ ^ i^3 + i < N ^ ⟨∀i: x + 1 <= i < x + k : i^3 + i < N⟩
≡{ true por INV ^ B
0 <= x + k ^ ⟨∀i: x + 1 <= i < x + k : i^3 + i < N⟩
≡{ cualquier k >= 0 cumple con la primera parte, luego forzamos un rango vacío con K = 1
0 <= x + 1 ^ ⟨∀i: x + 1 <= i < x + 1 : i^3 + i < N⟩
≡{ rango vacío
0 <= x + 1
≡{ por hipótesis
true

Finalmente
Const N: Int;
var x: Int;
{P: ⟨ ∃ n : 0 <= n : n^3 + n >= N⟩ ^ N >= 0}
x := 0
{INV: 0 <= x ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ }
do (x^3 + x < N) →
x := x + 1
od
{Q: 0 <= x ^ (x^3 + x >= N) ^ ⟨∀i: 0 <= i < x : i^3 + i < N⟩ }

3. Sea $N \geq 0$, especificar y derivar un programa que calcule el mayor entero x que satisface $x^3 + x \leq N$.

Const N: Int;
var x: Int;
{P: N >= 0}
S
{Q: ⟨ ∀i : x < i < N : i^3 + i > N⟩ ^ x^3 + x <= N }

#P nos dice que N es natural
#Q nos dice que todo i entre x y N no satisface la función, por lo que, x resulta ser el máximo i que cumpla la función

Paso 1 (Invariante)
INV : ⟨ ∀i : x < i < N : i^3 + i > N⟩
B = x^3 + x > N
¿Se cumple INV ^ ¬B → Q? si, de hecho son equivalentes.

Tenemos
Const N: Int;
var x: Int;
{P: N >= 0}
x := E;
{INV : ⟨ ∀i : x < i < N : i^3 + i > N⟩}
do (x^3 + x > N) →
  {⟨ ∀i : x < i < N : i^3 + i > N⟩ ^ x^3 + x > N}
  x := F;
  {⟨ ∀i : x < i < N : i^3 + i > N⟩}
od
{Q: ⟨ ∀i : x < i < N : i^3 + i > N⟩ ^ x^3 + x <= N }

Paso 2 (Inicializar) La única variable que podemos inicializar es x
x := F veamos si vale P → wp.(x := F).INV

asumimos P : N >= 0
wp.(x := F).⟨ ∀i : x < i < N : i^3 + i > N⟩
≡{ def de wp
⟨ ∀i : F < i < N : i^3 + i > N⟩
≡{ elijo F=N
⟨ ∀i : N < i < N : i^3 + i > N⟩
≡{ rango vacío
true

Paso 3(Definir cota)
Como se cumple para algún N. Entonces nuestra candidata a cota será t.x = N + x
Luego, restando a x, la cota decrece y proponemos la asignación x := x - k. veamos si se cumple INV ∧ B ⇒ wp.(x := x - k).INV

asumimos INV ∧ B ≡
  ⟨ ∀i : x < i < N : i^3 + i > N⟩ ^ x^3 + x > N
  ≡{ rango unitario al revez
  ⟨ ∀i : x < i < N : i^3 + i > N⟩ ^ ⟨ ∀i : i = x : i^3 + i > N⟩
  ≡{ partición de rango al revez
  ⟨ ∀i : x = i v x < i < N : i^3 + i > N⟩
  ≡{ lógica
  ⟨ ∀i : x <= i < N : i^3 + i > N⟩
  ≡{ lógica
  ⟨ ∀i : x - 1 < i < N : i^3 + i > N⟩

wp.(x := x - k).⟨ ∀i : x < i < N : i^3 + i > N⟩
≡{ def de wp
⟨ ∀i : x-k < i < N : i^3 + i > N⟩
≡{ tomamos k = 1

⟨ ∀i : x-1 < i < N : i^3 + i > N⟩
≡{ hipótesis
true

Finalmente tenemos
Const N: Int;
var x: Int;
{P: N >= 0}
x := N;
do (x^3 + x > N) →
x := x - 1;
od
{Q: ⟨ ∀i : x < i < N : i^3 + i > N⟩ ^ x^3 + x <= N }

4. (Suma de los elementos de un arreglo) Dado un arreglo de enteros, especificar y derivar un programa que calcule la suma de todos los elementos del arreglo.

Especificación:

Const N: Int;
var r: Int;
var A:array[0,N) of Int;
{P: N >= 0}
S

{Q: r = ⟨sum i : 0 <= i < N : A.i ⟩}

Paso 1 (invariante)
cambio de variable
{INV : r = ⟨sum i : 0 <= i < n : A.i ⟩ ^ 0 <= n <= N}
B ≡ (n != N)
¿se cumple INV ^ ¬B → Q? veamos.
r = ⟨sum i : 0 <= i < n : A.i ⟩ ^ 0 <= n <= N ^ N = n → r = ⟨sum i : 0 <= i < N : A.i ⟩
≡{ asumimos INV ^ ¬B
r = ⟨sum i : 0 <= i < N : A.i ⟩
≡{ de asumir N = n
r = ⟨sum i : 0 <= i < n : A.i ⟩
≡{ hipótesis
true

Paso 2 (Inicializar) Solo tenemos dos variables para inicializar r y n, entonces tendríamos
n,r:=E.F veamos si vale P → wp.(n,r:=E,F).INV
asumimos P: N >= 0
wp.(n,r:=E,F).(r = ⟨sum i : 0 <= i < n : A.i ⟩ ^ 0 <= n <= N)
≡{ def de wp
F = ⟨sum i : 0 <= i < E : A.i ⟩ ^ 0 <= n <= N
≡{ forzamos rango vacío con E = 0, segunda parte true por hipótesis
F = 0
≡{ tomamos F = 0
true

Paso 3 (cota) proponemos N - n. vale INV ^ B → N - n >= 0. veamos.
asumimos INV ^ B ≡ r = ⟨sum i : 0 <= i < n : A.i ⟩ ^ 0 <= n <= N ^ n != N
≡ r = ⟨sum i : 0 <= i < n : A.i ⟩ ^ 0 <= n < N
N - n >= 0
≡{ sumamos n
N >= n
≡{ lógica
N > n v n = N
≡{hipótesis
true

¿cómo hago decrecer la cota? Agrandando n
entonces vamos INV ^ B → wp.(n,r:=n+k,F).INV
asumimos INV ^ B ≡ r = ⟨sum i : 0 <= i < n : A.i ⟩ ^ 0 <= n < N
wp.(n,r:=n+k,F).INV
≡{ def de wp
F = ⟨sum i : 0 <= i < n+K : A.i ⟩ ^ 0 <= n+K <= N
≡{ proponemos K = 1
F = ⟨sum i : 0 <= i < n+1 : A.i ⟩ ^ 0 <= n+1 <= N
≡{ sep término, lógica
F = ⟨sum i : 0 <= i < n : A.i ⟩ + A.n ^ 0 <= n+1 ^ n+1 <= N
≡{ hipótesis
F = r + A.n ^ n < N
≡{hipótesis
F = r + A.n
≡{ tomamos F = r + A.n
true

Finalmente tenemos
Const N: Int;
var r: Int;
var n: Int;

var A:array[0,N) of Int;
{P: N >= 0}
r,n := 0,0;
do (n != N) →
  n,r:= n+1 , r + A.n
od
{Q: r = ⟨sum i : 0 <= i < N : A.i ⟩}

5. Sea A un arreglo de enteros.

   a) Especificar y derivar un programa que determine si todos los elementos de A son mayores a 0.

   b) Especificar y derivar un programa que determine si algún elemento de A es mayor a 0.

a)
  Const N: Int;
  Var r: bool;
  {P: N >= 0}
  S
  {Q: r = ⟨∀i : 0 <= i < N : A.i > 0⟩}

  Paso 1 (invariante)
  #Cambio de variable:
  {INV : r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}
  B = n != N
  #¿Se cumple INV ^ ¬B → Q? veamos.
  asumimos INV ^ ¬B
  r = ⟨∀i : 0 <= i < N : A.i > 0⟩
  ≡{ de asumir 0 <= n <= N ≡ 0 <= n < N v n=N
  r = ⟨∀i : 0 <= i < n : A.i > 0⟩
  ≡{ hipótesis
  true

  Const N: Int;
  Var r: bool;
  Var n: Int;
  {P: N >= 0}
  S
  {INV : r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}
  do (n != N) →
  {INV ^ B}
  S1
  {INV}
  od
  {Q: r = ⟨∀i : 0 <= i < N : A.i > 0⟩}

Paso 2 (Inicializar)
#Tenemos r y n para inicializar. veamos P → wp.(r,n := E,F).INV
asumimos P ≡ (N >= 0)
wp.(r,n := E,F).r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N
≡{ def de wp
E = ⟨∀i : 0 <= i < F : A.i > 0⟩ ^ 0 <= F <= N
≡{ forzamos rango vacío con F = 0
E = ⟨∀i : 0 <= i < 0 : A.i > 0⟩ ^ 0 <= 0 <= N
≡{ rango vacío, lógica, idempotencia

E = true
≡{ E = true
true

Const N: Int;
Var r: bool;
Var n: Int;
{P: N >= 0}
r := true;
n := 0;
{INV : r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}
do (n != N) →
{INV ^ B}
S1
{INV}
od
{Q: r = ⟨∀i : 0 <= i < N : A.i > 0⟩}

Paso 3 (cota)
proponemos como cota t = N - n, la cual decrece si aumentamos n luego proponemos la asignación: (r,n:=E,n+K).
veamos {INV ^ B} S1 {INV}. asumimos r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N ^ n != N
≡ r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n < N
wp.(r,n:=E,n+K).(r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N)
≡{ def de wp
E = ⟨∀i : 0 <= i < n+K : A.i > 0⟩ ^ 0 <= n+K <= N
≡{ proponemos k = 1
E = ⟨∀i : 0 <= i < n+1 : A.i > 0⟩ ^ 0 <= n+1 <= N
≡{ separación de término, lógica
E = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ A.n > 0 ^ 0 <= n+1 ^ n + 1 <= N
≡{ lógica, 0 <= n + 1 ≡ 0 <= n
E = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ A.n > 0 ^ 0 <= n < N
≡{ hipótesis
E = r ^ A.n > 0
≡{ separamos por casos
caso A.n > 0 ≡ true
E = r
≡{ E = r
true

caso A.n > 0 ≡ false
E = r ^ false
≡{ lógica
E = false

¿vale INV ^ B → t >= 0?
Asumimos INV ^ B ≡ r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n < N
N - n >= 0
≡{ sumamos n
N >= n
≡{ hipótesis
true

¿Vale {INV ^ B ^ T = X} if…fi {T < X}?
asumimos r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n < N ^ N - n = X

wp.(if…fi).(N - n < X)
≡{ wp del if
((A.n >= 0) v (A.n < 0)) ^ (A.n >= 0) → wp.(r, n := r, n+1).(N - n < X) ^ (A.n < 0) → wp.(r, n := false, n + 1).(N - n < X)
≡{ lógica , def wp, tricotomía
(A.n >= 0) → (N - (n+1) < X) ^ (A.n < 0) → (N - (n+1) < X)
≡{ aritmética
(A.n >= 0) → (N - n -1) < X) ^ (A.n < 0) → (N - n - 1) < X)
≡{ hipótesis
(A.n >= 0) → (X -1) < X) ^ (A.n < 0) → (X - 1) < X)
≡{ lógica
(A.n >= 0) → true ^ (A.n < 0) → true
≡{ asumiendo (A.n >= 0) ^ (A.n < 0)
true

Const N: Int;
Var r: bool;
Var n: Int;
{P: N >= 0}
r := true;
n := 0;
{INV : r = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}
do (n != N) →
{INV ^ B}
if (A.n >= 0) →
r, n := r, n+1
- **(A.n < 0) →**

r, n := false, n + 1
{INV}
od
{Q: r = ⟨∀i : 0 <= i < N : A.i > 0⟩}

b)
Const N: Int;
Var r: bool;
{P: N >= 0}
S
{Q: r = ⟨ ∃ i : 0 <= i < N : A.i > 0⟩}

Paso 1 (invariante)
#Cambio de variable:
{INV : r = ⟨ ∃ i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}
B = n != N
#¿Se cumple INV ^ ¬B → Q? veamos.
asumimos INV ^ ¬B
r = ⟨∃i : 0 <= i < N : A.i > 0⟩
≡{ de asumir 0 <= n <= N ≡ 0 <= n < N v n=N
r = ⟨∃i : 0 <= i < n : A.i > 0⟩
≡{ hipótesis
true

Const N: Int;
Var r: bool;
Var n: Int;
{P: N >= 0}
S
{INV : r = ⟨∃i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}

do (n != N) →
{INV ^ B}
S1
{INV}
od
{Q: r = ⟨∃i : 0 <= i < N : A.i > 0⟩}

Paso 2 (Inicializar)
#Tenemos r y n para inicializar. veamos P → wp.(r,n := E,F).INV
asumimos P ≡ (N >= 0)
wp.(r,n := E,F).r = ⟨∃i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N
≡{ def de wp
E = ⟨∃i : 0 <= i < F : A.i > 0⟩ ^ 0 <= F <= N
≡{ forzamos rango vacío con F = 0
E = ⟨∃i : 0 <= i < 0 : A.i > 0⟩ ^ 0 <= 0 <= N
≡{ rango vacío, lógica, idempotencia
E = false
≡{ E = false
true

Const N: Int;
Var r: bool;
Var n: Int;
{P: N >= 0}
r := false;
n := 0;
{INV : r = ⟨∃i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}
do (n != N) →
{INV ^ B}
S1
{INV}
od
{Q: r = ⟨∃i : 0 <= i < N : A.i > 0⟩}

Paso 3 (cota)
proponemos como cota t = N - n, la cual decrece si aumentamos n luego proponemos la asignación: (r,n:=E,n+K).
veamos {INV ^ B} S1 {INV}. asumimos r = ⟨∃ i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N ^ n != N
≡ r = ⟨∃ i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n < N
wp.(r,n:=E,n+K).(r = ⟨∃i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N)
≡{ def de wp
E = ⟨∃i : 0 <= i < n+K : A.i > 0⟩ ^ 0 <= n+K <= N
≡{ proponemos k = 1
E = ⟨∃ i : 0 <= i < n+1 : A.i > 0⟩ ^ 0 <= n+1 <= N
≡{ separación de término, lógica
E = (⟨∃i : 0 <= i < n : A.i > 0⟩ v A.n > 0) ^ 0 <= n+1 ^ n + 1 <= N
≡{ lógica, 0 <= n + 1 ≡ 0 <= n
E = (⟨∀i : 0 <= i < n : A.i > 0⟩ v A.n > 0) ^ 0 <= n < N
≡{ distributiva
E = ⟨∀i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n < N v A.n > 0 ^ 0 <= n < N
≡{ hipótesis
E = r v A.n > 0
≡{ separamos por casos
caso A.n > 0 ≡ true
E = r v true
≡{ absorbente
E = true

caso A.n > 0 ≡ false
E = r v false
≡{ neutro
E = r

¿vale INV ^ B → t >= 0?
Asumimos INV ^ B ≡ r = ⟨ ∃ i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n < N
N - n >= 0
≡{ sumamos n
N >= n
≡{ hipótesis
true

¿Vale {INV ^ B ^ T = X} if…fi {T < X}?
asumimos r = ⟨ ∃i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n < N ^ N - n = X
wp.(if…fi).(N - n < X)
≡{ wp del if
((A.n >= 0) v (A.n < 0)) ^ (A.n >= 0) → wp.(r, n := r, n+1).(N - n < X) ^ (A.n < 0) → wp.(r, n := false, n + 1).(N - n < X)
≡{ lógica , def wp, tricotomía
(A.n >= 0) → (N - (n+1) < X) ^ (A.n < 0) → (N - (n+1) < X)
≡{ aritmética
(A.n >= 0) → (N - n -1) < X) ^ (A.n < 0) → (N - n - 1) < X)
≡{ hipótesis
(A.n >= 0) → (X -1) < X) ^ (A.n < 0) → (X - 1) < X)
≡{ lógica
(A.n >= 0) → true ^ (A.n < 0) → true
≡{ asumiendo (A.n >= 0) ^ (A.n < 0)
true

```
Const N: Int;
  Var r: bool;
  Var n: Int;
  {P: N >= 0}
    r := false;
    n := 0;
  {INV : r = ⟨∃i : 0 <= i < n : A.i > 0⟩ ^ 0 <= n <= N}
  do (n != N) →
  {INV ^ B}
    if (A.n >= 0) →
      r, n := true, n+1
    ●   (A.n < 0) →
      r, n := r, n + 1
  {INV}
  od
  {Q: r = ⟨∃i : 0 <= i < N : A.i > 0⟩}
```

6. Especificar y derivar un programa que calcule la suma de los elementos pares de un arreglo de enteros.

Especificación:

Const N : Int, A : Array[0,N) of Int;
var r: Int;
{P: N >= 0}
S

{Q: r = ⟨sum i : 0 <= i < N ^ (i mod 2 == 0) : A.i⟩ }

Paso 1 (invariante)
Proponemos: {INV : r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n <= N}
B : n != N
tq se cumpla INV ^ ¬B → Q (si se cumple, son equivalentes)

Const N : Int, A : Array[0,N) of Int;
var r, n: Int;
{P: N >= 0}
S1
{INV : r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n <= N}
do (n != N) →
{INV ^ B}
S2
{INV}
od
{Q: r = ⟨sum i : 0 <= i < N ^ (i mod 2 == 0) : A.i⟩ }

Paso 2 (inicializar)
Proponemos r,n:=E,F. asumimos P = N >= 0 y veamos P → wp.(S1).INV
wp.(r,n:=E,F).(r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n <= N)
≡{ def de wp
E = ⟨sum i : 0 <= i < F ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= F <= N
≡{ forzamos rango vacío con F = 0
E = 0 ^ 0 <= F <= N
≡{ lógica
E = 0 ^ 0 <= F v F <= N
≡{ F = N
E = 0 ^ N <= N
≡{ lógica
E = 0 ^ N = N v N < N
≡{ lógica
E = 0 ^ N = N
≡{ lógica
E = 0
≡{ E = 0
true

Const N : Int, A : Array[0,N) of Int;
var r, n: Int;
{P: N >= 0}
r,n := 0, N;
{INV : r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n <= N}
do (n != N) →
{INV ^ B}
S2
{INV}
od
{Q: r = ⟨sum i : 0 <= i < N ^ (i mod 2 == 0) : A.i⟩ }

Paso 3 (Cota)

Proponemos t = N - n **¿vale INV ^ B → t >= 0? ^ ¿Vale {INV ^ B ^ T = X} S2 {T < X}?**

¿vale INV ^ B → t >= 0? veamos

Asumimos r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n <= N ^ n != N

r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n < N

r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n < N

N - n >= 0

≡{ sumamos n

N >= n

≡{ hipótesis

true

para ver {INV ^ B ^ T = X} S2 {T < X} hacemos decrecer la cota agrandando n, luego proponemos

r,n:=E,n+K; asumimos INV ^ B ≡ r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n <= N ^ n != N

≡ r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n < N

y veamos wp.(r,n:=E,n+K).r = ⟨sum i : 0 <= i < N ^ (i mod 2 == 0) : A.i⟩

≡{ def de wp

E = ⟨sum i : 0 <= i < n+K ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n+K <= N

≡{ k = 1

E = ⟨sum i : 0 <= i < n+1 ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n+1 <= N

≡{ lógica

E = ⟨sum i : 0 <= i < n+1 ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n < N

≡{ hipótesis

E = ⟨sum i : 0 <= i < n+1 ^ (i mod 2 == 0) : A.i⟩

≡{ lógica

E = ⟨sum i : 0 <= i <= n ^ (i mod 2 == 0) : A.i⟩

≡{ lógica

E = ⟨sum i : i = n v 0 <= i < n ^ (i mod 2 == 0) : A.i⟩

≡{ partición de rango

E = ⟨sum i : i = n ^ (i mod 2 == 0) : A.i⟩ + ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩

≡{ hipótesis

E = ⟨sum i : i = n ^ (i mod 2 == 0) : A.i⟩ + r

≡{ rango unitario y condición

E = ((n mod 2 == 0) → A.n v ¬(n mod 2 == 0) → 0) + r

separamos por casos: caso (n mod 2 == 0) → A.n

E = A.n + r

caso ¬(n mod 2 == 0) → 0)

E = 0 + r

¿Vale {INV ^ B ^ T = X} S2 {T < X}?

Finalmente tenemos:

```
Const N : Int, A : Array[0,N) of Int;
var r, n: Int;
{P: N >= 0}
   r,n := 0, N;
{INV : r = ⟨sum i : 0 <= i < n ^ (i mod 2 == 0) : A.i⟩ ^ 0 <= n <= N}
do (n != N) →
      {INV ^ B}
   if(n % 2 == 0) →
      r,n = A.n + r, n + 1
   else if (n % 2 != 0) →
```

r,n = r, n + 1
{INV}
od
{Q: r = ⟨sum i : 0 <= i < N ^ (i mod 2 == 0) : A.i⟩ }

de enteros.

7. Especificar y derivar un programa que calcule el factorial de un número.

Especificación:
Const N: Int;
var r: Int;
{P : N >= 0}
S
{Q : res = N!}

Paso 1 (Invariante) INV ^ ¬B → Q
Proponemos
INV : res = n! ^ 0 <= n <= N
B = n != N

Paso 2 (inicializar) {P} S {INV}
tenemos res,n:=E,F para eso veamos P → wp.(res,n:=E,F).INV
asumimos P
wp.(res,n:=E,F).(res = n! ^ 0 <= n <= N)
≡{ def de wp
E = F! ^ 0 <= F <= N
≡{ sabemos que F comienza desde 0 hasta N así que probamos con el mas simple F = 0
E = 0! ^ 0 <= 0 <= N
≡{ logica, hipotesis y aritmética
E = 1
≡{ tomamos E = 1
true

Paso 3(cota) {INV ^ B} S1 {INV}
proponemos como cota t = N - n y esta decrece si n crece, luego proponemos la asignación res,n:=E,n+K
asumimos res = n! ^ 0 <= n <= N ^ n != N o equivalentemente res = n! ^ 0 <= n < N
wp.(res,n:=E,n+K).(res = n! ^ 0 <= n <= N)
≡{ def de wp
E = n+K! ^ 0 <= n+K <= N
≡{proponemos K=1
E = n+1! ^ 0 <= n+1 <= N
≡{def de n!, lógica
E = n+1*n! ^ 0 <= n+1 ^ n + 1 <= N
≡{ lógica
E = n+1*n! ^ 0 <= n+1 ^ n < N
≡{ hipótesis
E = n + 1 * res
≡{ tomamos E = n + 1 * res
true

Paso 4 (vale la cota)
i) INV ∧ B ⇒ t ≥ 0
ii) {INV ∧ B ∧ t = T} S {t < T}

i)
res = n! ^ 0 <= n <= N ^ n != N → N - n >= 0
≡{ equivalencia
res = n! ^ 0 <= n < N → N - n >= 0

≡{ asumimos
N - n >= 0
≡{ aritmética
N >= n
≡{ lógica
N > n v n = N
≡{ hipótesis
true v n = N
≡{ abs
true

ii)
asumimos
{INV ∧ B ∧ t = T} ≡ res = n! ^ 0 <= n <= N ^ n != N ^ N - n = T ≡ res = n! ^ 0 <= n < N ^ N - n = T
y veamos
wp.(res,n:=n+1*res,n+1).(N - n < T)
≡{ def de wp
N - n + 1 < T
≡{ hipótesis
T + 1 < T
≡{ decrece? xd
true

8. Dado un arreglo $A : array[0, N)\, of\, Num$ con $N \geq 0$, contar cuántas veces coinciden dos elementos:

Const $N : Int, A : array\ [0, N)\, of\, Int;$
Var $r : Int;$
$\{P : N \geq 0\}$
S
$\{Q :\ r = \langle N\, i, j\ : 0 \leq i < j < N :\ A.i = A.j\,\rangle\}$

Paso 1 (Invariante) cambio de variable
{INV : r = ⟨N i , j : 0 <= i < j < n : A.i = A .j ⟩ ^ 0 <= n <= N}
B ≡ n != N

¿Se cumple INV ^ ¬B → Q? veamos
asumimos r = ⟨N i , j : 0 <= i < j < n : A.i = A .j ⟩ ^ 0 <= n <= N ^ n = N
≡ r = ⟨N i , j : 0 <= i < j < N : A.i = A .j ⟩ ^ 0 <= N
r = ⟨N i , j : 0 <= i < j < N : A.i = A .j ⟩
≡{ hipótesis
true

Replantear el ciclo:
Const N : Int, A : array[0, N) of Int ;
Var r, n : Int ;
{ P: N ≥ 0 }
S1
{ INV }
do n ≠ N →
{ INV ∧ B }
S2
{ INV }
od
{ Q: r =⟨ N i, j : 0 ≤ i < j < N : A.i = A.j ⟩}

Paso 2 (inicialización S1) n,r := E,F
asumimos P : N >= 0
wp.(n,r := E,F).(r = ⟨N i , j : 0 <= i < j < n : A.i = A .j ⟩ ^ 0 <= n <= N)
≡{ def de wp
F = ⟨N i , j : 0 <= i < j < E : A.i = A .j ⟩ ^ 0 <= E <= N
≡{ forzamos rango vacío con E=0, hipótesis
F = 0
≡{ F = 0
true
S1 = n,r := 0,0

Paso 3(cota y cuerpo del ciclo)
proponemos como cota t = N - n la cual decrece si aumentamos n. Luego proponemos la asignación
S2 = n,r := n + K, F y veamos {INV ^ B} S2 {INV}
asumimos r = ⟨N i , j : 0 <= i < j < n : A.i = A .j ⟩ ^ 0 <= n <= N ^ n ≠ N
o equivalentemente r = ⟨N i , j : 0 <= i < j < n : A.i = A .j ⟩ ^ 0 <= n < N
ahora veamos
wp.(n,r := n + K, F).(r = ⟨N i , j : 0 <= i < j < n : A.i = A .j ⟩ ^ 0 <= n <= N)
≡{ def de wp
F = ⟨N i , j : 0 <= i < j < n + K : A.i = A .j ⟩ ^ 0 <= n+K <= N
≡{ proponemos k = 1
F = ⟨N i , j : 0 <= i < j < n + 1 : A.i = A .j ⟩ ^ <u>0 <= n+1 <= N</u>
≡{ lógica 0 <= n+1 <= N ≡ 0 <= n ^ n < N y por hipótesis true
F = ⟨N i , j : 0 <= i < j < n + 1 : A.i = A .j ⟩
≡{ por lógica
0 <= i < j < n + 1
es lo mismo que
0 <= i < j ^ j < n + 1
equivalentemente
0 <= i < j ^ (j < n ^ j = n)
distributiva
(j < n ^ 0 <= i < j) v (j = n ^ 0 <= i < j)
≡{
F = ⟨N i , j : (j < n ^ 0 <= i < j) v (j = n ^ 0 <= i < j) : A.i = A .j ⟩
≡{ partición de rango
F = ⟨N i , j : (j < n ^ 0 <= i < j) : A.i = A .j ⟩ + ⟨N i , j : (j = n ^ 0 <= i < j) : A.i = A .j ⟩
≡{ eliminación de variable, lógica
F = ⟨N i , j : 0 <= i < j < n : A.i = A .j ⟩ + ⟨N i , j : 0 <= i < n : A.i = A.n ⟩

≡{ hipótesis
$F = r + \langle N\ i , j : 0 <= i < n : A.i = A.n \rangle$
Luego no podemos seguir derivando algo programable, por ende proponemos un ciclo anidado tq
$r2 = \langle N\ i , j : 0 <= i < n : A.i = A.n \rangle$
Replanteemos S2 como
$\{ INV \land B \}$
    S3 ;
    $\{ INV \land B \land\ r2 = \langle N\ i : 0 \leq i < n : A.i = A.n \rangle \}$
    $r , n := r + r2 , n + 1$
    $\{ INV \}$

Derivemos el nuevo ciclo S3
$\{ P' : INV \land B \}$
  S3;
$\{ Q' : INV \land B \land\ r2 = \langle N\ i : 0 \leq i < n : A.i = A.n \rangle \}$

Paso 1 (Invariante) notemos que n y r para Q son “constantes” luego usamos reemplazo de constante por variable m por n.
observamos que solo nos interesa reemplazar n por m en el rango.
$\{INV' : INV \land B \land\ r2 = \langle N\ i : 0 \leq i < m : A.i = A.n \rangle \wedge 0 <= m <= n \}$
$B \equiv m != n$
Luego vale $INV' \wedge \neg B \rightarrow Q'$

Replanteemos el ciclo S3:
$\{ P' : INV \land B \}$
  S1’;
{INV’}
do (m != n) →
{INV’ ^ B’}
  S2’;
{INV’}
od
$\{ Q' : INV \land B \land\ r2 = \langle N\ i : 0 \leq i < n : A.i = A.n \rangle \}$

Paso 2 (Inicialización S1’) inicializamos las variables m,r2:=E,F para ello asumimos P’ y veamos
wp.(m,r2:=E,F).INV’
≡{ def de wp
$INV \land B \land\ F = \langle N\ i : 0 \leq i < E : A.i = A.n \rangle \wedge 0 <= E <= n$
≡{ no nos interesa la primer parte
$F = \langle N\ i : 0 \leq i < E : A.i = A.n \rangle \wedge 0 <= E <= n$
≡{ forzamos rango vacío con E = 0
$F = \langle N\ i : 0 \leq i < 0 : A.i = A.n \rangle \wedge 0 <= 0 <= n$
≡{ rango vacío, hipótesis
$F = 0$
≡{ F = 0
true

S1’ = m,r2 := 0,0

Paso 3 (cota y cuerpo del ciclo) Proponemos cota t = n - m la cual decrece al aumentar m, luego proponemos la siguiente asignación S2’ = m,r2 := m + K, F y veamos {INV’ ^ B’} S2’ {INV’}
asumimos $INV' \wedge B' \equiv INV \land B \land\ r2 = \langle N\ i : 0 \leq i < m : A.i = A.n \rangle \wedge 0 <= m <= n \wedge m != n$ equivalentemente
$INV \land B \land\ r2 = \langle N\ i : 0 \leq i < m : A.i = A.n \rangle \wedge 0 <= m < n$
luego veamos
wp.(m,r2 := m + K, F).($INV \land B \land\ r2 = \langle N\ i : 0 \leq i < m : A.i = A.n \rangle \wedge 0 <= m <= n$)
≡{ def de wp
$F = \langle N\ i : 0 \leq i < m + K : A.i = A.n \rangle \wedge 0 <= m + K <= n$
≡{ proponemos k = 1

F =⟨ N i : 0 ≤ i < m + 1 : A.i = A.n ⟩ ^ 0 <= m + 1 <= n
≡{ por lógica
0 <= m + 1 <= n ≡ 0 <= m + 1 ^ m + 1 <= n ≡ 0 <= m < n
≡{ hipótesis
F =⟨ N i : 0 ≤ i < m + 1 : A.i = A.n ⟩ ^ true
≡{ por lógica
F =⟨ N i : 0 ≤ i <= m : A.i = A.n ⟩
≡{ lógica
F =⟨ N i : 0 ≤ i < m v i = m : A.i = A.n ⟩
≡{ partición de rango
F =⟨ N i : 0 ≤ i < m : A.i = A.n ⟩ + ⟨ N i : i = m : A.i = A.n ⟩
≡{ hipótesis
F = r2 + ⟨ N i : i = m : A.i = A.n ⟩
≡{ rango unitario y condición
F = r2 + A.m = A.n → 1
A.m != A.n → 0

Luego separamos por casos en el ciclo y finalmente tenemos

```
Const N : Int, A : array[0, N) of Int ;
Var r, n : Int ;
{ P: N ≥ 0 }
   n,r := 0,0
do n ≠ N →
   m,r2 := 0,0
   do (m != n) →
      if (A.m = A.n) →
         m,r2 := m + 1, r2 + 1
      else (A.m != A.n) →
         m,r2 := m + 1
   od
   r , n :=  r + r2 , n + 1
od
{ Q: r =⟨ N i, j : 0 ≤ i < j < N : A.i = A.j ⟩}
```

9. Dado un arreglo $A : array[0, N)\, of\, Num$ con $N \geq 0$, determinar si hay dos elementos que suman 8:

Const $N : Int, A : array\ [0, N)\, of\, Int;$
Var $r : Bool;$
$\{P : N \geq 0\}$
S
$\{Q :\ r = \langle \exists i, j\ : 0 \leq i < j < N :\ A.i + A.j = 8\,\rangle\}$

Paso 1 (invariante) Por cambio de constante por variable.
{INV : r = ⟨ ∃ i,j : 0 ≤ i < j < n : A.i + A.j = 8 ⟩ } ^ 0 ≤ n ≤ N
B = n != N
Luego INV ^ ¬B → Q

Paso 2 (Inicializamos)
asumimos P : N ≥ 0 y veamos
wp.(n,r := E,F).(INV)
≡{ def de wp

$F = \langle \exists i,j : 0 \leq i < j < E : A.i + A.j = 8 \rangle \wedge 0 \leq E \leq N$
≡{ forzamos rango vacío con E = 0, hipótesis
$F = false \wedge true$
≡{ lógica
$F = false$

Paso 3 (cota) Proponemos $t = N - n$ la cual decrece si aumentamos n ; Luego veamos
$INV \wedge B \rightarrow t \geq 0$
Asumimos $INV \wedge B$
$N - n \geq 0$
≡{ aritmética
$N \geq n$
≡{ hipótesis
true

Paso 4 (cuerpo del ciclo)
asumimos $INV \wedge B$ y veamos
$wp.(n,r:=n+K,F).(r = \langle \exists i,j : 0 \leq i < j < n : A.i + A.j = 8 \rangle \wedge 0 \leq n \leq N)$
≡{ def de wp
$F = \langle \exists i,j : 0 \leq i < j < n+K : A.i + A.j = 8 \rangle \wedge 0 \leq n+K \leq N$
≡{ proponemos aumentar n en 1 k = 1
$F = \langle \exists i,j : 0 \leq i < j < n+1 : A.i + A.j = 8 \rangle \wedge 0 \leq n+1 \leq N$
≡{ lógica
$0 \leq n+1 \leq N \equiv 0 \leq n+1 \wedge n+1 \leq N \equiv 0 \leq n \wedge n < N \equiv 0 \leq n < N$
≡{ hipótesis
$F = \langle \exists i,j : 0 \leq i < j < n+1 : A.i + A.j = 8 \rangle \wedge true$
≡{ neutro ^
$F = \langle \exists i,j : 0 \leq i < j < n+1 : A.i + A.j = 8 \rangle$
≡{ lógica
$0 \leq i < j < n+1$
equivale a
$0 \leq i < j \wedge j < n+1$
equivale a
$0 \leq i < j \wedge j = n \vee j < n$
distributiva
$(0 \leq i < j \wedge j = n) \vee (j < n \wedge 0 \leq i < j)$
$F = \langle \exists i,j : (0 \leq i < j \wedge j = n) \vee (j < n \wedge 0 \leq i < j) : A.i + A.j = 8 \rangle$
≡{ partición de rango
$F = \langle \exists i,j : (0 \leq i < j \wedge j = n) : A.i + A.j = 8 \rangle \vee \langle \exists i,j : (j < n \wedge 0 \leq i < j) : A.i + A.j = 8 \rangle$
≡{ eliminación de variable, hipótesis
$F = r \vee \langle \exists i : (0 \leq i < n : A.i + A.n = 8 \rangle$

No podemos seguir programando, proponemos un ciclo anidado
Replanteemos S2 como
{ $INV \wedge B$ }
S3 ;
{ $INV \wedge B \wedge r2 = \langle \exists i : 0 \leq i < n : A.i + A.n = 8 \rangle$ }
r , n := r v r2 , n + 1
{ INV }

Luego derivamos el nuevo ciclo S3
{P' : INV ^ B}
S3

{Q' : INV ^ B ^ r2 =⟨ ∃ i : 0 ≤ i < n : A.i + A.n = 8 ⟩ }

Paso 1(invariante)
cambio constante por variable
INV' : INV ^ B ^ r2 = ⟨ ∃ i : 0 ≤ i < m : A.i + A.n = 8 ⟩^ 0 ≤ m ≤ n
B' = m != n
Luego vale INV' ^ B' → Q'

Paso 2(inicializar) r2,m:=E,F
asumimos P'
wp.(r2,m:=E,F).(INV ^ B ^ r2 = ⟨ ∃ i : 0 ≤ i < m : A.i + A.n = 8 ⟩^ 0 ≤ m ≤ n)
≡{ def de wp
E = ⟨ ∃ i : 0 ≤ i < F : A.i + A.n = 8 ⟩^ 0 ≤ F ≤ n
≡{ Rango vacío con F=0 y hipótesis
E = false ^ true
≡{ abs
E = false

luego S1' = r2,m:= false , 0

Paso 3 (cota) proponemos t' = n - m la cual decrece si aumentamos m. luego,
INV' ^ B' → t ≥ 0
asumimos INV' ^ B
n - m ≥ 0
≡{ aritmética
n ≥ m
≡{ hipótesis
true

Paso 4 (cuerpo del ciclo) r2,m := E,m+k
asumimos
INV ^ B ^ r2 = ⟨ ∃ i : 0 ≤ i < m : A.i + A.n = 8 ⟩^ 0 ≤ m ≤ n ^ B'
≡{ equivalentemente
INV ^ B ^ r2 = ⟨ ∃ i : 0 ≤ i < m : A.i + A.n = 8 ⟩^ 0 ≤ m < n
luego,
wp.(r2,m := E,m+k).(INV ^ B ^ r2 = ⟨ ∃ i : 0 ≤ i < m : A.i + A.n = 8 ⟩^ 0 ≤ m ≤ n)
≡{ def de wp
E = ⟨ ∃ i : 0 ≤ i < m+k : A.i + A.n = 8 ⟩^ 0 ≤ m+k ≤ n)
≡{ proponemos k = 1
E = ⟨ ∃ i : 0 ≤ i < m+1 : A.i + A.n = 8 ⟩^ 0 ≤ m+1 ≤ n)
≡{ hipótesis
E = ⟨ ∃ i : 0 ≤ i < m+1 : A.i + A.n = 8 ⟩
≡{ lógica
0 ≤ i < m+1 ≡ 0 ≤ i ≤ m ≡ 0 ≤ i < m v i = m
≡{ equivalentemente
E = ⟨ ∃ i : 0 ≤ i < m v i = m : A.i + A.n = 8 ⟩
≡{ partición de rango
E = ⟨ ∃ i : 0 ≤ i < m : A.i + A.n = 8 ⟩v E = ⟨ ∃ i : i = m : A.i + A.n = 8 ⟩
≡{ rango unitario, hipótesis
E = r2 v A.m + A.n = 8
≡{ E = r2 v A.m + A.n = 8
true

Finalmente el programa quedaría como

Const N :
Int, A : array [0, N) of Int;
Var r,r2 : Bool;
Var n,m : Int;
{P : N ≥ 0}
r,n := false, 0
<u>do</u> (n != N) →
  r2, m := false, 0
  <u>do</u> (m != n) →
    r2, m := r2 v A.m + A.n = 8 ,m + 1
  <u>od</u>
  r , n := r v r2 , n + 1
<u>od</u>
{Q : r = ⟨ ∃ i, j : 0 ≤ i < j < N : A.i + A.j = 8 ⟩}

10. Especificar y derivar: Dado un arreglo $a : array[0, N)\, of\, Num$ con $N \geq 0$ determinar si alguno de sus elementos es igual a la suma de los anteriores. Usar fortalecimiento de invariante.

Const N: Int, A : array [0, N) of Int;
Var r : Bool;
{N ≥ 0}
S
{ r = ⟨ ∃ i : 0 ≤ i < N : A.i = ⟨sum j : 0 ≤ j < i : A.j ⟩ }

abreviamos ⟨sum j : 0 ≤ j < i : A.j ⟩ ≡ sum.i
la función equivale a
summ.0 = 0
summ.i+1 = summ.i + A.i

Paso 1 (Invariante) cambio de constante por variable
INV : r = ⟨ ∃ i : 0 ≤ i < n : A.i = sum.i ⟩ ^ 0 ≤ n ≤ N
B = n != N
luego, vale INV ^ ¬B → Q

Paso 2 (Inicializar) n,r := E,F
asumimos P
wp.(n,r := E,F).(r = ⟨ ∃ i : 0 ≤ i < n : A.i = sum.i ⟩ ^ 0 ≤ n ≤ N)
≡{ def de wp
F = ⟨ ∃ i : 0 ≤ i < E : A.i = sum.i ⟩ ^ 0 ≤ E ≤ N
≡{ rango vacío con E = 0, hipótesis
F = false ^ true
≡{ abs
F = false
≡{ F = false
true

S1 = n,r := 0, false

Paso 3 (cota) Proponemos t = N - n la cual decrece si aumentamos n, veamos INV ^ B → t ≥ 0
asumimos INV ^ B
N - n ≥ 0
≡{ aritmética
N ≥ n
≡{ hipótesis
true

Paso 4 (Cuerpo del ciclo) proponemos n,r := n + K, F
asumimos INV ^ B
wp.(n,r := n + K, F).(r = ⟨ ∃ i : 0 ≤ i < n : A.i = sum.i ⟩ ^ 0 ≤ n < N)
≡{ def de wp
F = ⟨ ∃ i : 0 ≤ i < n+K : A.i = sum.i ⟩ ^ 0 ≤ n+K < N
≡{ proponemos K = 1, hipótesis
F = ⟨ ∃ i : 0 ≤ i < n + 1 : A.i = sum.i ⟩
≡{ lógica
0 ≤ i ≤ n
0 ≤ i < n v i = n
≡{ lógica
F = ⟨ ∃ i : (0 ≤ i < n) v (i = n) : A.i = sum.i ⟩
≡{ partición de rango
F = ⟨ ∃ i : (0 ≤ i < n) : A.i = sum.i ⟩ v ⟨ ∃ i, j : i = n : A.i = sum.i ⟩
≡{ lógica, eliminación de variable, hipótesis
F = r v A.n = summ.n

Fortalecemos invariante
INV' =INV ^ r2 = ⟨sum j : 0 ≤ j < n : A.j ⟩

Inicializamos nuevamente
asumimos P

wp.(n,r,r2 := E,F,C).(r = ⟨ ∃ i : 0 ≤ i < n : A.i = sum.i ⟩ ^ 0 ≤ n ≤ N ^ r2 = ⟨sum j : 0 ≤ j < n : A.j ⟩)
≡{ def de wp
F = ⟨ ∃ i, j : 0 ≤ j < i < E : A.i = sum.i ⟩ ^ 0 ≤ E ≤ N ^ C = ⟨sum j : 0 ≤ j < n : A.j ⟩
≡{ Forzamos rango vacío con E = 0
F = false ^ C = false

Luego **S1 = n,r,r2 := 0,false,false**

Cuerpo del ciclo nuevamente
asumimos INV' ^ B
wp.(n,r,r2 := n+1,F,C).(r = ⟨ ∃ i : 0 ≤ i < n : A.i = sum.i ⟩ ^ 0 ≤ n ≤ N ^ r2 = ⟨sum j : 0 ≤ j < n : A.j ⟩
≡{ def de wp
C = ⟨sum j : 0 ≤ j < n+1 : A.j ⟩
≡{ lógica, partición de rango
C = ⟨sum j : 0 ≤ j < n : A.j ⟩ + ⟨sum j : j = n : A.j ⟩
≡{ rango unitario, hipótesis
C = r2 + A.n

Luego **S3 = n,r,r2 := n+1, r v A.n = r2, r2 + A.n**

<u>Finalmente</u>

Const N: Int, A : array [0, N) of Int;
Var r : Bool;
$\{N \geq 0\}$
n,r,r2 := 0, false, 0
do →
    n,r,r2 := n+1, r v (A.n = r2), r2 + A.n
od
$\{ r = \langle \exists i : 0 \leq i < N : A.i = \langle sum\ j : 0 \leq j < i : A.j \rangle \rangle \wedge r2 = \langle sum\ j : 0 \leq j < n : A.j \rangle \}$

11. Especificar y derivar: Dado un arreglo $a : array[0, N)\ of\ Num$ con $N \geq 0$ determinar si sus elementos son iguales al factorial de la posición. Usar fortalecimiento de invariante.

Const N: Int, A : array [0, N) of Int;
Var r : Bool;
$\{P: N \geq 0\}$
S
$\{Q: r = \langle \forall i : 0 \leq i < N : A.i = i! \rangle\}$

Paso 1 (Invariante)
INV : $r = \langle \forall i : 0 \leq i < n : A.i = i! \rangle \wedge 0 \leq n \leq N$
B = n != N
Luego, se cumple INV ^ ¬B → Q

Paso 2(Inicializamos) r,n := E,F
asumimos P
$wp.(r,n := E,F).(r = \langle \forall i : 0 \leq i < n : A.i = i! \rangle \wedge 0 \leq n \leq N)$
≡{ def de wp
$E = \langle \forall i : 0 \leq i < F : A.i = i! \rangle \wedge 0 \leq F \leq N$
≡{forzamos rango vacío con F=0, luego rango vacío, hipótesis, neutro ^
E = true

S1 = r,n := true, 0

Paso 3(cota) Proponemos t = N - n la cual decrece si aumentamos n.
Luego, INV ^ B → t ≥ 0
asumimos INV ^ B ≡ $r = \langle \forall i : 0 \leq i < n : A.i = i! \rangle \wedge 0 \leq n < N$
N - n ≥ 0
≡{ aritmética, hipótesis
true

Paso 4 (Cuerpo del ciclo) proponemos la asignación S2 = r,n := F, n + 1
asumimos INV ^ B ≡ $r = \langle \forall i : 0 \leq i < n : A.i = i! \rangle \wedge 0 \leq n < N$
$wp.(r,n := F, n + 1).(r = \langle \forall i : 0 \leq i < n : A.i = i! \rangle \wedge 0 \leq n \leq N)$
≡{ def de wp
$F = \langle \forall i : 0 \leq i < n+1 : A.i = i! \rangle \wedge 0 \leq n+1 \leq N$
≡{ lógica
$F = \langle \forall i : 0 \leq i \leq n : A.i = i! \rangle \wedge 0 \leq n+1 \wedge n + 1 \leq N$

≡{ lógica
F = ⟨ ∀ i : 0 ≤ i < n v n = i : A.i = i! ⟩ ^ 0 ≤ n+1 ^ n < N
≡{ lógica, partición de rango
F = ⟨ ∀ i : 0 ≤ i < n: A.i = i! ⟩ ^ ⟨ ∀ i : n = i : A.i = i! ⟩ ^ 0 ≤ n < N
≡{ hipótesis, rango unitario
F = r ^ A.n = n!

<u>Fortalecemos la invariante fact.n = A.n = n!</u>

INV' : r = ⟨ ∀ i : 0 ≤ i < n : A.i = i! ⟩ ^ 0 ≤ n ≤ N ^ fact.n = n!

Inicializamos
asumimos P
wp.(r,n,fact := E,F,C).(r = ⟨ ∀ i : 0 ≤ i < n : A.i = i! ⟩ ^ 0 ≤ n ≤ N ^ fact.n = n!)
≡{ def de wp
E = ⟨ ∀ i : 0 ≤ i < F : A.i = i! ⟩ ^ 0 ≤ F ≤ N ^ fact.F = F!
≡{ rango vacío con F=0, mismos pasos, 0! = 1
E = true ^ fact.F = 1

S1' = r,n,fact := true, 0, 1

Cuerpo del ciclo
asumimos INV' ^ B ≡ r = ⟨ ∀ i : 0 ≤ i < n : A.i = i! ⟩ ^ 0 ≤ n < N ^ fact.n = n!)
wp.(r,n,fact := E, n + 1, F).(r = ⟨ ∀ i : 0 ≤ i < n : A.i = i! ⟩ ^ 0 ≤ n ≤ N ^ fact = n!)
≡{ def de wp
E = ⟨ ∀ i : 0 ≤ i < n + 1 : A.i = i! ⟩ ^ 0 ≤ n + 1 ≤ N ^ F = (n+1)!
≡{ mismos pasos
E = r ^ A.n = n! ^ F = (n+1)!
≡{ elijo E = r ^ (A.n = fact)
≡{ def de n!
F = (n+1)!
≡{ def !
F = (n+1)*fact
Finalmente
Const N: Int, A : array [0, N) of Int;
Var r : Bool;
{P: N ≥ 0}
r,n,fact := true, 0, 1
do →
    r,fact,n := r ^ (A.n = fact), (n+1)*fact, n + 1
od
{Q: r = ⟨ ∀ i : 0 ≤ i < N : A.i = i! ⟩ ^ fact = n! }

12. Especificar y derivar un programa imperativo que calcule Fibonacci de un número dado. Usar fortalecimiento de invariante.

Const N : Int, A : array [0,N) of Int;
Var f : Int;
{P : N ≥ 0}
S
{Q : f = fib.N}
con
fib.0 = 0

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

Paso 1 (invariante)
Con cambio de variable por constante proponemos
INV : f = fib.n ^ 0 ≤ n ≤ N
B = n != N
luego vale INV ^ ¬B → Q

Paso 2 (inicializar) f,n := E,F
asumimos P
wp.(f,n := E,F).(f = fib.n ^ 0 ≤ n ≤ N)
≡{ def de wp
E = fib.F ^ 0 ≤ F ≤ N
≡{ proponemos F = 0
E = fib.0 ^ 0 ≤ 0 ≤ N
≡{ hipótesis, def de fib
E = 0
≡{ elijo E = 0
true
luego vale {P} S1 {INV}

Paso 3 (cota) Proponemos t = N - n la cual decrece si aumentamos n
¿Se cumple INV ^ B → t ≥ 0? veamos.
asumiendo INV ^ B ≡ f = fib.n ^ 0 ≤ n < N
N - n ≥ 0
≡{ aritmética, hipótesis
true
luego vale INV ^ B → t ≥ 0

Paso 4 (Cuerpo del ciclo) Proponemos f,n := F,n+K
asumimos INV ^ B ≡ f = fib.n ^ 0 ≤ n < N
wp.(f,n := F,n+K).(f = fib.n ^ 0 ≤ n ≤ N)
≡{ def de wp
F = fib.n+K ^ 0 ≤ n+K ≤ N
≡{ proponemos aumentar n en 1 k = 1
F = fib.n+1 ^ 0 ≤ n+1 ≤ N
≡{ hipótesis
F = fib.n+1

Fortalecemos invariante
INV' = f = fib.n ^ 0 ≤ n ≤ N ^ f2 = fib.(n+1)

Inicializamos de nuevo f,f2,n := F,E,K
asumimos P
wp.(f,f2,n := F,E,K).(f = fib.n ^ 0 ≤ n ≤ N ^ f2 = fib.(n+1))
≡{ def de wp
F = fib.K ^ 0 ≤ K ≤ N ^ E = fib.(K+1)
≡{ como antes proponemos K = 0, F = 0,
F = 0 ^ E = fib.(1)

≡{ def de fib
E = 1

luego S1' = f,f2,n := 0,1,0

Cuerpo del ciclo nuevamente Proponemos f,f2,n := F,E,n+K
Asumimos INV' ^ B ≡ f = fib.n ^ 0 ≤ n < N ^ f2 = fib.(n+1)
wp.(f,f2,n := F,E,n+K).(f = fib.n ^ 0 ≤ n ≤ N ^ f2 = fib.(n+1))
≡{ def de wp
F = fib.n+K ^ 0 ≤ n+K ≤ N ^ E = fib.(n+K+1)
≡{ mismos pasos
F = f2 ^ E = fib.(2)
≡{ def de fib
F = f2 ^ E = fib(n) + fib(n+1)
≡{ hipótesis
F = f2 ^ E = f + f2

Luego f,f2,n := f2,f+f2,n+1

Finalmente
Const N : Int, A : array [0,N) of Int;
Var f : Int;
{P : N ≥ 0}
f,f2,n := 0,1,0
do(n != N) →
  f,f2,n := f2,f+f2,n+1
od
{Q : f = fib.N}
con
fib.0 = 0
fib.1 = 1
fib(n+2) = fib(n) + fib(n+1)

13. (Máxima diferencia) Dado un arreglo de enteros, calcular la máxima diferencia entre dos de sus elemento (en orden, el primero menos el segundo).

La especificación del programa es:

Const $N : Int;$
Var $a : array[0, N)\ of\ Int; r : Int;$
$\{P : N \geq 2\}$
S
$\{Q : r = \langle \text{Max}\ p, q\ : 0 \leq p < q < N :\ a.p - a.q\ \rangle\}$

Paso 1 (Invariante)
INV = r = ⟨ Max p , q : 0 ≤ p < q < n : a.p - a.q ⟩ ^ 2 ≤ n ≤ N
B = n < N
Luego vale INV ^ ¬B → Q

Paso 2 (inicializamos)
asumimos P y luego
wp.(r,n:=F,E).(INV)
≡{ def de wp
F = ⟨ Max p , q : 0 ≤ p < q < E : a.p - a.q ⟩ ^ 2 ≤ E ≤ N
≡{ como Max no tiene rango vacío, proponemos E = 2 ya que es el caso base

$F = \langle \text{Max } p , q : 0 \leq p < q < 2 : a.p - a.q \rangle \wedge \underline{2 \leq 2 \leq N}$
≡{ logica, hipotesis
$F = \langle \text{Max } p , q : 0 \leq p < q < 2 : a.p - a.q \rangle$
≡{ lógica
$0 \leq p < q < 2$
$\equiv 0 \leq p \wedge p < q \wedge q \leq 1$ por transitividad
$\equiv 0 \leq p \wedge p < q \wedge 0 < q \leq 1$
$\equiv 0 \leq p \wedge p < q \wedge 1 \leq q \leq 1$
$\equiv 0 \leq p \wedge p < q \wedge q = 1$ luego $0 \leq p < 1$
$\equiv p = 0 \wedge q = 1$
$F = \langle \text{Max } p , q : p = 0 \wedge q = 1 : a.p - a.q \rangle$
≡{ rango unitario
$F = A.0 - A.1$

<u>luego s1 = r,n := A.0 - A.1, 0</u>

Paso 3 (función de cota) proponemos $t = N - n$ la cual decrece si aumentamos n en 1 por cada iteración del ciclo, luego
$INV \wedge B \rightarrow t \geq 0$
asumimos $INV \wedge B$
$N - n \geq 0$
≡{ aritmética
$N \geq n$
≡{ hipótesis
true

Paso 4 (cuerpo del ciclo)
asumimos $INV \wedge B \equiv r = \langle \text{Max } p , q : 0 \leq p < q < n : a.p - a.q \rangle \wedge 2 \leq n < N$
wp.(r,n := E,n+1).(INV)
≡{ def de wp
$E = \langle \text{Max } p , q : 0 \leq p < q < n+1 : a.p - a.q \rangle \wedge \underline{2 \leq n+1 \leq N}$
≡{ lógica
$E = \langle \text{Max } p , q : 0 \leq p < q < n+1 : a.p - a.q \rangle \wedge \underline{2 \leq n+1 \wedge n + 1 \leq N}$
≡{ lógica
$E = \langle \text{Max } p , q : 0 \leq p < q < n+1 : a.p - a.q \rangle \wedge \underline{2 \leq n \wedge n < N}$
≡{ hipótesis y lógica en $0 \leq p < q < n+1$
$\equiv 0 \leq p < q \wedge q < n+1$
$\equiv 0 \leq p < q \wedge (q=n \vee q < n)$
$\equiv (0 \leq p < q \wedge q = n) \vee (0 \leq p < q \wedge q < n)$
$E = \langle \text{Max } p , q : (0 \leq p < q \wedge q = n) \vee (0 \leq p < q \wedge q < n) : a.p - a.q \rangle$
≡{ partición de rango
$E = \langle \text{Max } p , q : (0 \leq p < q \wedge q = n) : a.p - a.q \rangle \text{ max } \langle \text{Max } p , q : (0 \leq p < q \wedge q < n) : a.p - a.q \rangle$
≡{ eliminación de variable, lógica
$E = \langle \text{Max } p : 0 \leq p < n : a.p - a.n \rangle \text{ max } \langle \text{Max } p , q : (0 \leq p < q < n) : a.p - a.q \rangle$
≡{ hipótesis distributiva con max
$E = r \text{ max } ((\langle \text{Max } p : 0 \leq p < n : a.p \rangle) - a.n)$

Luego no podemos seguir derivando por lo que fortalecemos invariante
$r2 = (\langle \text{Max } p : 0 \leq p < n : a.p \rangle)$
y **S2 = r,n := r max (r2 - a.n), n + 1**

$INV' \equiv INV \wedge r2 = \langle \text{Max } p : 0 \leq p < n : a.p \rangle$

Inicializamos de nuevo:
r,r2,n := E,F,N asumimos P y luego
wp.(r,r2,n := E,F,N).(INV ^ r2 = ⟨ Max p : 0 ≤ p < n : a.p ⟩)
≡{ mismos pasos salvo en r2
E = A.0 - A.1 ^ F = ⟨ Max p : 0 ≤ p < 2 : a.p ⟩
≡{ elegimos E = A.0 - A.1
F = ⟨ Max p : 0 ≤ p < 2 : a.p ⟩
≡{ lógica
F = ⟨ Max p : 0 ≤ p ≤ 1 : a.p ⟩
≡{ partición de rango, rango unitario
F = a.0 máx a.1

luego **S1´ = r,r2,n := A.0 - A.1, r2 max a.n, 0**
Aunque debemos plantear un if
if (A.0 ≥ A.1) →
   r,r2,n := A.0 - A.1, A.0, 0
else (A.0 < A.1) →
   r,r2,n := A.0 - A.1, A.1, 0

Veamos el cuerpo del ciclo nuevamente
INV' ^ B ≡ r = ⟨ Max p , q : 0 ≤ p < q < n : a.p - a.q ⟩ ^ 2 ≤ n < N ^ r2 = ⟨ Max p : 0 ≤ p < n : a.p ⟩
wp.(r,r2,n := E,F,n+1).(INV')
≡{ def de wp, mismos pasos salvo para r2
E = r max (r2 - a.n) ^ F = ⟨ Max p : 0 ≤ p < n+1 : a.p ⟩
≡{ elegimos E = r max (r2 - a.n)
F = ⟨ Max p : 0 ≤ p < n+1 : a.p ⟩
≡{ mismos pasos que inicialización
F = r2 max a.n
Luego **S2' =**
if (r2 ≥ a.n && r ≥ (r2 - a.n)) →
   r,r2,n := r,r2,n+1
else(r2 < a.n && r < (r2 - a.n)) →
   r,r2,n := r2 - a.n, a.n ,n+1
fi

Por lo que, finalmente
Const N : Int;
Var a : array[0, N) of Int;
r,r2,n : Int;
{P : N ≥ 2}
if (A.0 ≥ A.1) →
   r,r2,n := A.0 - A.1, A.0, 2
else (A.0 < A.1) →
   r,r2,n := A.0 - A.1, A.1, 2
fi
do(n < N) →
   if (r ≥ (r2 - a.n) && r2 ≥ a.n)→
   r,r2,n := r,r2,n+1
else (r ≥ (r2 - a.n) && r2 < a.n) →
   r,r2,n := r, a.n, n+1

else (r < (r2 - a.n) && r2 ≥ a.n) →
    r,r2,n := r2 - a.n, r2, n+1
else (r < (r2 - a.n) && r2 < a.n) →
    r,r2,n := r2 - a.n, a.n , n+1
fi
od
{Q : r = ⟨Max p, q : 0 ≤ p < q < N : a.p − a.q ⟩}

[4,2,1,6,7]
0 ≤ p < q < 5 ≡ 0 ≤ p < q ≤ 4
p ∈ (0,1,2,3)
q ∈ (1,2,3,4)
luego p,q ∈ { (0,1),(0,2),(0,3),(0,4),(1,2),(1,3),(1,4),(2,3),(2,4),(3,4) }

4 - 2 max 4 - 1 max 4 - 6 max 4 - 7 max 2 - 1 max 2 - 6 max 2 - 7 max 1 - 6 max 1 - 7 max 6 - 7
2 max 3 max -2 max -3 max 1 max -4 max -5 max -5 max -6 max -1 = **3**

Ahora veamos con el programa

r	r2	n	estado
2	4	0	S1
2	4	1	S1
2	4	2	S1
3	4	3	S1
3	6	4	S1
3	7	5	S2

14. (Segmento de suma máxima) Dado un arreglo de enteros, calcular la suma del segmento de suma máxima del arreglo.

La especificación del programa es:

Const $N : Int$;
Var $a : array[0, N)\ of\ Int; r : Int$;
$\{P : N \geq 0\}$
S
$\{Q : r = \langle \text{Max } p, q : 0 \leq p \leq q \leq N : sum.p.q \rangle$
$[\![sum.p.q = \langle \sum i : p \leq i < q : a.i \rangle]\!]\}$

sum.n.n
≡{ equivalentemente
⟨ summ i : n ≤ i < n : a.i ⟩
≡{ rango vacío
0

summ.p.q+1
≡{ equivalentemente
⟨ summ i : n ≤ i < n+1 : a.i ⟩
≡{ n ≤ i < n+1 ≡ n ≤ i < n v i = n, partition de rango, rango unitario

summm.p.q + a.q

Paso 1 (Invariante) Cambio de variable
INV ≡ r = ⟨Max p, q : 0 ≤ p ≤ q ≤ n : sum.p.q ⟩ ^ 0 ≤ n ≤ N
B ≡ n != N
Luego vale INV ^ ¬B → Q

Paso 2 (Inicializamos) proponemos r,n := E,F
asumimos P
wp.(r,n := E,F).(INV)
≡{ def de wp
E = ⟨Max p, q : 0 ≤ p ≤ q ≤ F : sum.p.q ⟩ ^ 0 ≤ F ≤ N
≡{ proponemos F = 0
E = ⟨Max p, q : 0 ≤ p ≤ q ≤ 0 : sum.p.q ⟩ ^ 0 ≤ 0 ≤ N
≡{ logica, hipotesis
E = ⟨Max p, q : p = 0 ^ q = 0 : sum.p.q ⟩
≡{ rango unitario
E = summ.0.0
≡{ def de summ
E = 0

Luego S1 = r,n := 0,0

Paso 3 (Función de cota) Proponemos t = N - n la cual decrece si aumentamos n en 1 unidad, luego vale INV ^ B → t ≥ 0

Paso 4 (Cuerpo del ciclo) proponemos r,n := E,n+1
asumimos INV ^ B ≡ r = ⟨Max p, q : 0 ≤ p ≤ q ≤ n : sum.p.q ⟩ ^ 0 ≤ n < N
wp.(r,n := E,n+1).(r = ⟨Max p, q : 0 ≤ p ≤ q ≤ n : sum.p.q ⟩ ^ 0 ≤ n ≤ N)
≡{ def de wp
E = ⟨Max p, q : 0 ≤ p ≤ q ≤ n+1 : sum.p.q ⟩ ^ 0 ≤ n+1 ≤ N
≡{ por lógica, hipótesis 0 ≤ n+1 ≤ N ≡ 0 ≤ n + 1 ^ n+1 ≤ N ≡ 0 ≤ n ^ n < N ≡ 0 ≤ n < N
E = ⟨Max p, q : 0 ≤ p ≤ q ≤ n+1 : sum.p.q ⟩
≡{ por lógica
0 ≤ p ≤ q ≤ n+1
0 ≤ p ≤ q ^ q ≤ n + 1
0 ≤ p ≤ q ^ (q = n + 1 v q < n+1)
distributiva
(0 ≤ p ≤ q ^ q < n+1) v (0 ≤ p ≤ q v q = n + 1)
(0 ≤ p ≤ q ≤ n) v (0 ≤ p ≤ q v q = n + 1)
≡{
E = ⟨Max p, q : (0 ≤ p ≤ q ≤ n) v (0 ≤ p ≤ q v q = n + 1) : sum.p.q ⟩
≡{ partición de rango
E = ⟨Max p, q : (0 ≤ p ≤ q ≤ n) : sum.p.q ⟩ max ⟨Max p, q : (0 ≤ p ≤ q v q = n + 1) : sum.p.q ⟩
≡{ eliminación de variable, hipótesis
E = r max ⟨Max p, q : 0 ≤ p ≤ n + 1 : sum.p.(n+1) ⟩
≡{ por lógica
0 ≤ p ≤ n + 1
0 ≤ p < n + 1 v p = n + 1
(0 ≤ p ≤ n) v (p = n + 1)
≡{
E = r max ⟨Max p, q : (0 ≤ p ≤ n) v (p = n + 1) : sum.p.(n+1) ⟩

≡{ partición de rango
E = r max ⟨Max p, q : 0 ≤ p ≤ n : sum.p.(n+1) ⟩ máx ⟨Máx p, q : p = n + 1 : sum.p.(n+1) ⟩
≡{ eliminación de variable
E = r max ⟨Max p : 0 ≤ p ≤ n : <u>sum.p.(n+1)</u> ⟩ máx <u>sum.(n+1).(n+1)</u>
≡{ def de summ
E = r max ⟨Max p : 0 ≤ p ≤ n : summ.p.n + a.n ⟩ máx 0
≡{ distributiva
E = r max (⟨Max p : 0 ≤ p ≤ n : summ.p.n⟩ + a.n) máx 0
≡{ **fortalecemos**
E = r max (t + a.n) máx 0
con t = ⟨Max p : 0 ≤ p ≤ n : summ.p.n⟩

INV' ≡ INV ^ t = ⟨Max p : 0 ≤ p ≤ n : summ.p.n⟩

Paso 2 (inicializamos) DE NUEVO
asumimos P
wp.(r,t,n := E,F,C).(INV')
≡{ mismos pasos E=0 ^ C=0
F = ⟨Max p : 0 ≤ p ≤ 0 : summ.p.0⟩
≡{ p = 0, def de summ
F = 0
Luego S1' ≡ r,t,n := 0,0,0

Paso 4 (cuerpo del ciclo) DE NUEVO
asumimos INV' ^ B ≡ INV ^ B ^ t = ⟨Max p : 0 ≤ p ≤ n : summ.p.n⟩
wp.(r,t,n := E,F,n+1).(INV')
≡{ mismos pasos para r y n
t = ⟨Max p : 0 ≤ p ≤ n+1 : summ.p.n+1⟩
≡{ partición de rango
t = ⟨Max p : 0 ≤ p ≤ n : summ.p.n+1⟩ max ⟨Max p : p = n+1 : summ.p.n+1⟩
≡{ rango unitario
t = ⟨Max p : 0 ≤ p ≤ n : summ.p.n+1⟩ max summ.n+1.n+1
≡{ def de summ
t = ⟨Max p : 0 ≤ p ≤ n : summ.p.n+1⟩ max 0
≡{ def de summ, distributiva
t = (⟨Max p : 0 ≤ p ≤ n : summ.p.n⟩ + a.n) max 0
≡{ hipotesis
t = (t + a.n) max 0

S2' ≡ r,t,n := r max (t + a.n) máx 0, (t + a.n) max 0, n+1
≡ r,t,n := r max (t + a.n), (t + a.n) max 0, n+1

Finalmente
Const N : Int;
Var a : array[0, N) of Int;
r,t,n : Int;
{P : N ≥ 0}
r,t,n := 0,0,0
<u>do</u>(n != N) →
r,t,n := r max (t + a.n), (t + a.n) max 0, n+1
<u>od</u>
{Q : r = ⟨Max p, q : 0 ≤ p ≤ q ≤ N : sum.p.q ⟩

$|[sum.p.q = \langle P\ i : p \leq i < q : a.i \rangle]|\}$

15. Dada la siguiente especificación:

Const $M : Int, A : array[0, M)\ of\ Int$;
Var $r : Int$;
$\{P : M \geq 0\}$
S
$\{Q : r = \langle N\ p, q\ : 0 \leq p < q < M :\ A.p * A.q \geq 0 \rangle\}$
Decir en palabras qué hace el programa y derivarlo.

En orden, calcula las veces en las cuales un producto entre dos elementos es positivo

Paso 1 (Invariante) cambio de constante por variable
$INV \equiv r = \langle N\ p, q : 0 \leq p < q < m : A.p * A.q \geq 0 \rangle \wedge 0 \leq m \leq M$
$B = (m\ != M)$
Luego vale $INV \wedge \neg B \rightarrow Q$

Paso 2 (Inicializar)
asumimos P
$wp.(r,m:=E,F).(INV)$
$\equiv\{$ def de inv
$E = \langle N\ p, q : 0 \leq p < q < F : A.p * A.q \geq 0 \rangle \wedge 0 \leq F \leq M$
$\equiv\{$ rango vacío con F=0, hipótesis
$E = 0$
$\equiv\{ E = 0$
true

S1 = r,m := 0,0

Paso 3 (cota) Proponemos $t = N - n$ la cual decrece si aumentamos n luego vale
$INV \wedge B \rightarrow t \geq 0$
Paso 4(cuerpo del ciclo)
asumimos $INV \wedge B \equiv r = \langle N\ p, q : 0 \leq p < q < m : A.p * A.q \geq 0 \rangle \wedge 0 \leq m < M$
$wp.(r,m:=E,m+1).(INV)$
$\equiv\{$ def de wp
$E = \langle N\ p, q : 0 \leq p < q < m+1 : A.p * A.q \geq 0 \rangle \wedge \underline{0 \leq m+1 < M}$
$\equiv\{$ logica, hipotesis
$E = \langle N\ p, q : 0 \leq p < q < m+1 : A.p * A.q \geq 0 \rangle$
$\equiv\{$ por lógica
$0 \leq p < q < m+1 \equiv 0 \leq p < q \wedge q < m+1 \equiv (0 \leq p < q) \wedge (q = m \vee q < m)$
por distributiva $(0 \leq p < q \wedge q = m) \vee (0 \leq p < q \wedge q < m)$
$\equiv\{$
$E = \langle N\ p, q : (0 \leq p < q \wedge q = m) \vee (0 \leq p < q \wedge q < m) : A.p * A.q \geq 0 \rangle$
$\equiv\{$ partición de rango
$E = \langle N\ p, q : (0 \leq p < q \wedge q = m) : A.p * A.q \geq 0 \rangle + \langle N\ p, q : (0 \leq p < q \wedge q < m) : A.p * A.q \geq 0 \rangle$
$\equiv\{$ eliminación de variable y hipotesis
$E = r + \langle N\ p, q : 0 \leq p < m : A.p * A.m \geq 0 \rangle$
$\equiv\{$ definición de conteo
$E = r + \langle summ\ p, q : 0 \leq p < m \wedge A.p * A.m \geq 0 : 1 \rangle$
$\equiv\{$ regla de signos

$E = r + \langle \text{summ } p, q : 0 \le p < m \wedge (A.p \ge 0 \wedge A.m \ge 0) \vee (A.p < 0 \wedge A.m < 0) : 1 \rangle$

Caso 1 ≡ A.m ≥ 0

$E = r + \langle \text{summ } p, q : 0 \le p < m \wedge (A.p \ge 0 \wedge true) \vee (A.p < 0 \wedge false): 1 \rangle$
≡{ lógica
$E = r + \langle \text{summ } p, q : 0 \le p < m \wedge A.p \ge 0 : 1 \rangle$
≡{ def de conteo
$E = r + \langle N\ p, q : 0 \le p < m : A.p \ge 0\rangle$

Fortalecemos el caso A.m ≥ 0
E = r + rpos
con rpos = ⟨N p, q : 0 ≤ p < m : A.p ≥ 0⟩

Caso 2 ≡ A.m < 0

$E = r + \langle \text{summ } p, q : 0 \le p < m \wedge (A.p \ge 0 \wedge false) \vee (A.p < 0 \wedge true) : 1 \rangle$
≡{ lógica
$E = r + \langle \text{summ } p, q : 0 \le p < m \wedge A.p < 0 : 1 \rangle$
≡{ def de conteo
$E = r + \langle N\ p, q : 0 \le p < m : A.p < 0\rangle$

Fortalecemos el caso A.m < 0
E = r + rneg
con rneg = ⟨N p, q : 0 ≤ p < m : A.p < 0 ⟩

Luego nuestro nuevo invariante:
$INV' \equiv INV \wedge rpos = \langle N\ p, q : 0 \le p < m : A.p \ge 0\rangle$
$\qquad\qquad rneg = \langle N\ p, q : 0 \le p < m : A.p < 0\rangle$

Inicializamos
Como m = 0 en la inicialización, tenemos rango vacío de conteo = 0 y nos queda
r,rpos,rneg,m := 0,0,0,0

Cuerpo del ciclo
Asumimos $INV' \wedge B \equiv INV \wedge B \wedge rpos = \langle N\ p, q : 0 \le p < m : A.p \ge 0\rangle$
$\qquad\qquad\qquad\qquad \wedge\ rneg = \langle N\ p, q : 0 \le p < m : A.p < 0\rangle$
Luego
wp.(r,rpos,rneg,m:=E,F,G,m+1).(INV')
≡{ def de wp, mismos pasos salvo para rpos y rneg
$F = \langle N\ p, q : 0 \le p < m+1 : A.p \ge 0\rangle \wedge G = \langle N\ p, q : 0 \le p < m+1 : A.p < 0\rangle$
≡{ lógica
$0 \le p < m+1$
$0 \le p \le m$
$(0 \le p < m) \vee (p = m)$
$F = \langle N\ p, q : (0 \le p < m) \vee (p = m) : A.p \ge 0\rangle \wedge G = \langle N\ p, q : (0 \le p < m) \vee (p = m) : A.p < 0\rangle$
={ partición de rango
$F = \langle N\ p, q : (0 \le p < m) : A.p \ge 0\rangle + \langle N\ p, q : p = m : A.p \ge 0\rangle \wedge G = \langle N\ p, q : (0 \le p < m) : A.p < 0\rangle +$
$\langle N\ p, q : p = m : A.p < 0\rangle$
≡{ hipótesis
$F = rpos + \langle N\ p, q : p = m : A.p \ge 0\rangle \wedge G = rneg + \langle N\ p, q : p = m : A.p < 0\rangle$
≡{ rango unitario
$(A.m \ge 0) \rightarrow F = rpos + 1$

¬(A.m ≥ 0) → F = rpos + 0
^
(A.m < 0) → G = rneg + 1
¬(A.m < 0) → G = rneg

Finalmente

Const M : Int, A : array[0, M) of Int;
Var r,rpos,rneg,m : Int;
{P : M ≥ 0}
r,rpos,rneg,m := 0,0,0,0
<u>do</u>(m != M) →
    if(A.m ≥ 0) →
        r,rpos,rneg,m := r + rpos, rpos + 1, rneg, m+1
    else(A.m < 0) →
        r,rpos,rneg,n := r + rneg, rpos, rneg + 1, m+1
<u>od</u>
{Q : r = ⟨N p, q : 0 ≤ p < q < M : A.p ∗ A.q ≥ 0 ⟩}

Testeo
[-1,2,-3,4]

r	rpos	rneg	m	estado
0	0	0	0	S1
0	0	1	1	s1
0	1	1	2	s1
1	1	2	3	s1
2	2	2	4	s2