1. Considere el problema de, dado un arreglo, calcular la suma del producto de todos los segmentos del arreglo, especificado de la siguiente manera:

Const N : Int, A : array[0, N) of Int;
Var r : Int;
{P : N ≥ 0}
S
{Q : r = ⟨ summ p, q : 0 ≤ p ≤ q ≤ N : prod.p.q ⟩
|[prod.p.q = ⟨ ∏ i : p ≤ i < q : A.i ⟩]|}

a) Calcular el resultado para A = [3, −2, 1]. Justificar, enumerando todos los elementos del rango.
Ayuda: El resultado es −8.

b) Derivar un programa imperativo que resuelva este problema.
Ayudas:
No intentar hacerlo con ciclos anidados.
Sale con un fortalecimiento.
Antes de fortalecer, cuidado con A.n

a) r = ⟨ summ p, q : 0 ≤ p ≤ q ≤ N : prod.p.q ⟩
≡{ N = 3
r = ⟨ summ p, q : 0 ≤ p ≤ q ≤ 3 : prod.p.q ⟩
≡{ enumeramos el rango
r = ⟨ summ p, q : p, q ∈ {(0,0),(0,1),(0,2),(0,3),(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)}: prod.p.q ⟩
≡{ aplicamos término
prod.0.0 + prod.0.1 + prod.0.2 +prod.0.3 + prod.1.1 + prod.1.2 + prod.1.3 + prod.2.2 + prod.2.3 + prod.3.3
≡{ def de prod
1 ≤ i < 3 A = [3, −2, 1]
1 + 3 + (-6) + (-6) + 1 + (-2) + (-2) + 1 + 1 + 1
≡{ aritmética
8-16 = -8

b)
Paso 1 (Invariante)
INV = r = ⟨ summ p, q : 0 ≤ p ≤ q ≤ n : prod.p.q ⟩ ^ 0 ≤ n ≤ N
B = n < N
Luego vale INV ^ ¬B → Q

Paso 2 (inicializamos)
forzamos rango vacío en prod con n = 0 (la sumatoria de 0 ≤ p ≤ q ≤ 0) y luego r = 1
r,n := 1,0

Paso 3 (Cota) Proponemos t = N - n ya que N es constante y podemos aumentar n para decrecer la cota en cada paso del ciclo, luego si la cota se hace 0 la guarda no vale, es decir, INV ^ B → t ≥ 0 se cumple ya que N - n ≥ 0 luego N ≥ n y es lo que asumimos en INV ^ B

Paso 4 (Cuerpo del ciclo) Encontremos E que preserve el INV mientras n crece

asumimos INV ^ B ≡ r = ⟨ summ p, q : 0 ≤ p ≤ q ≤ n : prod.p.q ⟩ ^ 0 ≤ n < N

wp.(r,n := E,n+1).(INV)

≡{ def de wp

E = ⟨ summ p, q : 0 ≤ p ≤ q ≤ n+1 : prod.p.q ⟩ ^ 0 ≤ n+1 ≤ N

≡{ lógica en el rango, lógica e hipótesis

E = ⟨ summ p, q : 0 ≤ p ≤ q ≤ n v q = n+1 : prod.p.q ⟩

≡{ partición de rango

E = ⟨ summ p, q : 0 ≤ p ≤ q ≤ n : prod.p.q ⟩ + ⟨ summ p, q : q = n+1 : prod.p.q ⟩

≡{ hipótesis, rango unitario

E = r + prod.p.n+1

≡{ def de prod

E = r + ⟨ ∏ i : p ≤ i < n+1 : A.i ⟩

≡{ lógica en el rango

E = r + ⟨ ∏ i : p ≤ i < n v i = n : A.i ⟩

≡{ partición de rango, rango unitario

E = r + (⟨ ∏ i : p ≤ i < n : A.i ⟩ * A.n)

Luego no podemos seguir programando, por lo que fortalecemos el invariante

INV' ≡ INV ^ r2 = ⟨ ∏ i : p ≤ i < n : A.i ⟩

Luego inicializamos nuevamente con n = 0, y por rango vacío r2 = 1

r,r2,n := 1,1,0

Ahora veamos el cuerpo del ciclo

wp.(r,r2,n := E,F,n+1).(INV')

≡{ mismos pasos salvo para r2

F = ⟨ ∏ i : p ≤ i < n+1 : A.i ⟩

≡{ lógica en el rango

F = ⟨ ∏ i : p ≤ i < n v i = n : A.i ⟩

≡{ partición de rango y rango unitario

F = r2 * A.n

Paso 5 (Probar que la cota decrece a medida que ejecutamos el ciclo)

veamos

{INV' ^ B ^ t = X} r,r2,n := r + (r2 * A.n), r2 * A.n, n+1 {t < X}

≡{ asumimos INV ^ B ^ N - n = X

wp.(r,r2,n := r + (r2 * A.n), r2 * A.n, n+1).(N-n < X)

≡{ def de wp

N - (n+1) < X

≡{ aritmética

N - n - 1 < X

≡{ hipótesis

X - 1 < X

≡{ lógica

true

Finalmente:
Const N : Int, A : array[0, N) of Int;
Var r,r2,n : Int;
{P : N ≥ 0}
r,r2,n := 1,1,0
<u>do</u> (n < N) →
r,r2,n := r + (r2 * A.n), r2 * A.n, n+1
<u>od</u>
$\{Q : r = \langle \text{summ } p, q : 0 \leq p \leq q \leq N : prod.p.q \rangle$
$|[prod.p.q = \langle \prod i : p \leq i < q : A.i \rangle]|\}$

2. Especificar con pre y post condición los siguientes problemas. Declarar constantes y variables necesarias para la especificación. No derivar.

a) Dado un arreglo A de N ≥ 0 elementos, calcular si los elementos forman una escalera ascendente de números.
Ejemplo:
Con A = [−3, −2, −1, 0, 1] la respuesta es positiva.
Con A = [11, 12, 12, 13] la respuesta es negativa.

b) Dado un arreglo A de N ≥ 0 elementos, calcular la suma de los elementos pares por un lado, y la suma de los elementos impares por otro.
Ejemplo:
Con A = [4, −8, 9, 12, −17], los elementos pares suman 8, y los impares suman −8.

a)
Const N : Int, A : array [0,N) of Int;
Var r : Int;
{N ≥ 0}
S
$\{r = \langle \forall i : 0 \leq i < N : A.i < A.(i+1) \rangle \}$

b)
Const N : Int, A : array [0,N) of Int;
Var par, impar : Int;
{N ≥ 0}
S
$\{ par = \langle \text{summ } i : 0 \leq i < N \wedge A.i \text{ mod } 2 == 0 : A.i \rangle$
$\wedge\ impar = \langle \text{summ } i : 0 \leq i < N \wedge \neg(A.i \text{ mod } 2 == 0) : A.i \rangle \}$

3. Escribir un programa imperativo que resuelva el problema del ejercicio 2b. No hace falta hacer la derivación.

```
Const N : Int, A : array [0,N) of Int;
Var par, impar : Int;
{N ≥ 0}
par, impar, n:= 0,0,0
do(n < N) →
   if (A.n % 2 == 0) →
      par, impar, n := A.n + par, impar, n+1
   else(¬(A.n % 2 == 0)) →
      par, impar, n := par, impar + A.n, n+1
od
{ par = ⟨ summ i : 0 ≤ i < N ^ A.i mod 2 == 0: A.i ⟩
^ impar = ⟨ summ i : 0 ≤ i < N ^ ¬(A.i mod 2 == 0) : A.i⟩}
```