Ejercicio 1. Consideremos el problema de sumar los elementos de un arreglo y la siguiente implementación en SmallLang, con el invariante del ciclo.

Especificación

proc sumar (in s: $array < \mathbb{Z} >$) : $\mathbb{Z}$
requiere $\{True\}$
asegura $\{res = \sum_{j=0}^{|s|-1} s[j]\}$

Implementación en SmallLang

```
res := 0;
i := 0;
while (i < s.size()) do
  res := res + s[i];
  i := i + 1
endwhile
```

Invariante de Ciclo

$$I \equiv 0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]$$

a) Escribir la precondición y la postcondición del ciclo.

b) ¿Qué punto falla en la demostración de corrección si el primer término del invariante se reemplaza por $0 \leq i < |s|$?

c) ¿Qué punto falla en la demostración de corrección si el límite superior de la sumatoria $(i-1)$ se reemplaza por i?

d) ¿Qué punto falla en la demostración de corrección si se invierte el orden de las dos instrucciones del cuerpo del ciclo?

e) Mostrar la corrección parcial del ciclo, usando los primeros puntos del teorema del invariante.

f) Proponer una función variante y mostrar la terminación del ciclo, utilizando la función variante.

a) La poscondicion del ciclo sera la misma que el *asegura* de la especificacion, $res = \sum_{j=0}^{i-1} s[j]$
La precondicion por otro lado no estoy seguro, sera que res=0 y i=0(las primeras dos lineas de S)?
Could be
b) de reemplazarse el primer termino del invariante este fallaria a la hora de mantenerse cierto al salir del ciclo
Esto pues el while aumenta i en uno |s| veces, donde la |s|-ava vez no entra al ciclo y el programa termina.
c) Deja de ser cierto durante toda la ejecucion del programa, basicamente te esta pidiendo que el termino que el programa esta por sumar ya este sumado a res
d) Al llegar a la ultima iteracion se intentaria acceder a un elemento del arreglo que no existe?¿? creo
f)

- $Pc \implies I$

  $res = 0 \wedge i = 0 \implies 0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]$

  $res = 0 \wedge i = 0 \implies 0 \leq 0 \leq |s| \wedge_L 0 = \sum_{j=0}^{1-1}$

  $res = 0 \wedge i = 0 \implies true$

- $\{I \wedge B\}S\{I\}$

  $\{0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j] \wedge i < |s|\} \equiv \{0 \leq i < |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]\}$

  res:= res + s[i]

  i:=i+1

  $\{0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]\}$

  Esto se traduce a comparar la primera parte de la tripla de Hoare con wp(res:=res+s[i]; i:=i+1, I)

  $\{0 \leq i < |s| \wedge_L 0 \leq i+1 \leq |s| \wedge_L res + s[i] = \sum_{j=0}^{i} s[j]\}$

  $\equiv \{0 \leq i < |s| \wedge_L -1 \leq i \leq |s| - 1 \wedge_L res = \sum_{j=0}^{i-1} s[j]\}$

  $\equiv \{0 \leq i < |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]\}$

  Y podemos ver que ambos $I \wedge B$ y la wp calculada son iguales, por lo tanto la tripla de Hoare es correcta

- $I \wedge \neg B \implies Q_c$

  $0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j] \wedge \neg(i < |s|)$

  $\equiv 0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j] \wedge i \geq |s|$

  $\equiv i = |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]$

  $\equiv res = \sum_{j=0}^{|s|-1} s[j]$

  Que es efectivamente nuestra postcondicion de la especificacion.

f) Propuesta 1: $fv = |s| - i + 1$

$\{I \wedge B \wedge v_0 = fv\} res := res + s[i]; i := i + 1 \{fv < v_0\} \equiv Pc \implies wp(s, fv < v_0)$

$wp(S, fv < v_0) \equiv wp(res := res + s[i], wp(i := i + 1, |s| - i + 1 < v_0))$
cosas...
$\quad \equiv |s| - (i + 1) + 1 < v_0 \equiv |s| - i < v_0$

retomando el principio:
$I \wedge B \wedge v_0 = fv \implies |s| - i < v_0$?
Si, pues $v_o = fv \implies |s| - i < |s| - i + 1$ que es trivialmente cierto.
Ahora, mi propuesta cumple con el segundo statement?
$I \wedge fv \leq 0 \implies \neg B$
$\alpha :\ |s| - i + 1 \leq 0 \implies |s| + 1 \leq i$
$\beta :\ \neg B \equiv i \geq |s| \equiv |s| \leq i$ $\longleftarrow$A lo que quiero llegar
$I \equiv 0 \leq i \leq |s| \wedge_L res = \sum_{j=0}^{i-1} s[j]$
Tengo informacion de sobra, $\alpha \implies |s| \leq i \equiv \neg B$
Seems a bit too easy pero es el primer ejercicio de la guia so...

Ejercicio 2. Dadas la especificación y la implementación del problema `sumarParesHastaN`, escribir la precondición y la postcondición del ciclo, y mostrar su corrección a través del teorema del invariante.

Especificación

proc `sumarParesHastaN` (in n: $\mathbb{Z}$) : $\mathbb{Z}$
    requiere $\{n \geq 0\}$
    asegura $\{res = \sum_{j=0}^{n-1}(\text{if } j \ mod \ 2 = 0 \text{ then } j \text{ else } 0 \text{ fi})\}$

Implementación en SmallLang

```
res := 0;
i := 0;
while (i < n) do
  res := res + i;
  i := i + 2
endwhile
```

Invariante de ciclo

$$I \equiv 0 \leq i \leq n+1 \land i \ mod \ 2 \ = \ 0 \land res = \sum_{j=0}^{i-1}(\text{if } j \ mod \ 2 = 0 \text{ then } j \text{ else } 0 \text{ fi})$$

La precondicion del ciclo seria ....
La poscondicion es $res = \sum_{j=0}^{n-1} if \ \ j \ \ mod2 = 0 \ \ then \ \ j \ \ else \ \ 0$
$P \implies I$ es trivialmente cierto

$2^{da})\{I \land B\}res := res + i; i := i + 2\{I\}$

$I \equiv 0 \leq i \leq n+1 \land i\%2 = 0 \land res = \sum_{j=0}^{i-1} if \ \ j\%2 = 0 \ \ then \ \ j \ \ else \ \ 0$ ←Por las dudas: Uso % en lugar de mod por cuestiones de spacing
$B \equiv i < n$
$wp(S, I) \equiv 0 \leq i \leq n+1 \land i\%2 = 0 \land res + i = \sum_{j=0}^{i+1} if \ \ j\%2 = 0 \ \ then \ \ j \ \ else \ \ 0$
Analizando un poco nos damos cuenta de que i siempre sera par, por lo que i+1 es impar, entonces el termino i+1 de la sumatoria siempre sumara 0 (por el condicional), asi que podemos obviarlo. A su vez, podemos restar a ambos lados el termino i-esimo, que este sera par y por lo tanto si contara en la sumatoria, simplificando la expresion.
$wp(S, I) \equiv 0 \leq i \leq n+1 \land i\%2 = 0 \land res = \sum_{j=0}^{i-1} if \ \ j\%2 = 0 \ \ then \ \ j \ \ else \ \ 0$
Entonces, $I \land B \equiv 0 \leq i < n \land i\%2 = 0 \land res = \sum_{j=0}^{i-1} if \ \ j\%2 = 0 \ \ then \ \ j \ \ else \ \ 0$
Que efectivamente implica mi wp, pues el segundo termino es identico y en el primero, $0 \leq i < n \implies 0 \leq i \leq n+1$

$3^{ra})\ I \land \neg B \implies Q_c$
$I \equiv 0 \leq i \leq n+1 \land i\%2 = 0 \land res = \sum_{j=0}^{i-1} if \ \ j\%2 = 0 \ \ then \ \ j \ \ else \ \ 0$
$\neg B \equiv i \geq n$
$n \leq i \land i \leq n+1 \equiv n \leq i \leq n+1$
Pero i tiene que ser par, por lo que: $n \leq i \leq n+1 \equiv n = i = n \equiv i = n$
$i = n \land i\%2 = 0 \land res = \sum_{j=0}^{i-1} if \ \ j\%2 = 0 \ \ then \ \ j \ \ else \ \ 0 \equiv res = \sum_{j=0}^{n-1} if \ \ j\%2 = 0 \ \ then \ \ j \ \ else \ \ 0 \equiv Q_c$
Posible invariante: $n - i + 1$
$\{I \land B \land v_0 = fv\}S\{fv < v_0\}$
$wp(i := i+2, n - i + 1 < v_0) \equiv n - i - 1 < v_0$
entonces $v_0 = fv \implies n - i - 1 < n - i + 1$
$5^{ta})\ I \land fv \leq 0 \implies \neg B$
Quiero entonces llegar a $\neg B \equiv i \geq n$
$fv \leq 0 \implies n + 1 \leq i$
$I \implies 0 \leq i \leq n+1$
Estas dos cosas implican que $i = n+1$ que a su vez implica $i \geq n$
De forma un poco confusa y poco organizada, pero queda entonces demostrada la correctitud del ciclo.