[illegible]

Algoritmos y Estructuras de Datos

Primer Parcial — Sábado 7 de octubre de 2023

#Orden	Libreta	Apellido y Nombre	E1	E2	E3	E4	Nota Final
[redacted]		[redacted]	20	30	20	30	100

- Es posible tener una hoja (2 carillas) escrita a mano, con las anotaciones que se deseen
- Cada ejercicio debe entregarse en **hojas separadas**
- Incluir en cada hoja el número de orden asignado, número de libreta, número de hoja, apellido y nombre
- El parcial se aprueba con 60 puntos. Para promocionar es necesario tener al menos 70 y ningún ejercicio con 0 puntos (en ambos parciales).

E1. Especificación de problemas [20 pts]

Se desea especificar el problema *primosEnCero* que dada una secuencia s de enteros devuelve la secuencia pero con los valores que se encuentran en posiciones correspondientes a un número primo reemplazados por 0. Si lo desea puede ayudarse escribiendo predicados y funciones auxiliares.

Ejemplos

- $primosEnCero([0, 1, 2, 3, 4, 5, 6]) = [0, 1, 0, 0, 4, 0, 6]$
- $primosEnCero([5, 7, -2, 13, -9, 1]) = [5, 7, 0, 0, -9, 0]$

E2. Modelado con TADs [30 pts]

Queremos modelar el funcionamiento de un vivero. El vivero arranca su actividad sin ninguna planta y con un monto inicial de dinero. Las plantas las compramos en un mayorista que nos vende la cantidad que deseemos pero solamente de a una especie por vez. Como vivimos en un país con inflación, cada vez que vamos a comprar tenemos un precio diferente para la misma planta. Para poder comprar plantas tenemos que tener suficiente dinero disponible, ya que el mayorista no acepta fiarnos.

A cada planta le ponemos un precio de venta por unidad. Ese precio tenemos que poder cambiarlo todas las veces que necesitemos. Para simplificar el problema, asumimos que las plantas las vendemos de a un ejemplar (cada venta involucra un solo ejemplar de una única especie). Por supuesto que para poder hacer una venta tenemos que tener *stock* de esa planta y tenemos que haberle fijado un precio previamente. Además, se quiere saber en todo momento cuál es el balance de caja, es decir, el dinero que tiene disponible el vivero.

1. [10 pts] Indique las operaciones (procs) del TAD con todos sus parámetros.
2. [15 pts] Describa el TAD en forma completa, indicando sus observadores, los requiere y asegura de las operaciones. Puede agregar los predicados y funciones auxiliares que necesite, con su correspondiente definición.
3. [5 pts] ¿Qué cambiaría si supiéramos a priori que cada vez que compramos en el mayorista pagamos exactamente el 10% más que la vez anterior? Describa los cambios en palabras.

E3. Precondición más débil [20 pts]

Dado el siguiente condicional determinar la precondición más débil que permite hacer valer la poscondición (Q) propuesta. Simplifique la fórmula resultante tanto como sea posible.

```
if i mod 2 = 0 then
  | s[i] = 2 * s[i]
else
  | s[0] = 3;
end
```

$Q \equiv \{(\forall j : \mathbb{Z})(0 \le j < |s| \to_L s[j] \text{ mod } 2 = 0)\}$

E4. Correctitud del ciclo [30 pts]

Dado el siguiente ciclo, su precondición (P_c) y su postcondición (Q_c).

$P_c \equiv \{i = |s| - 1 \land res = 0\}$

```
while i ≥ 0 do
  | res := res + s[i] + 1;
  | i := i − 1;
end
```

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

a) [10 pts] Proponer un invariante (I) y una función variante (f_v) para el ciclo.

b) [20 pts] Demostrar los siguientes pasos de la demostración de correctitud del ciclo.

i) [5 pts] $P_c \to I$

ii) [10 pts] $(I \land \neg B) \to Q_c$

iii) [5 pts] $(I \land f_v \le 0) \to \neg B$

[redacted] [redacted] HOJA N° 1

FECHA [redacted]

1)

~~proc primosEnCero (inout $s : seq\langle\mathbb{Z}\rangle$)~~

~~requiere : {True}~~

~~aseguro: { $|s| = |old(s)|$ $\wedge_L$~~

~~$(\forall j:\mathbb{Z})(0 \leq j < |s|$ $\wedge_L$ $\neg esPrimo(old(s)[j])$~~

~~$\rightarrow_L$ $s[j] = old(s)[j]$)~~

proc primosEnCero (inout $s : seq\langle\mathbb{Z}\rangle$)

requiere: {True}

aseguro: { $|s| = |old(s)|$ $\wedge_L$

$(\forall j:\mathbb{Z})(0 \leq j < |s| \wedge \neg esPrimo(j) \rightarrow_L s[j] = old(s)[j])$

$\wedge$ $(\forall i:\mathbb{Z})(0 \leq i < |s| \wedge esPrimo(i) \rightarrow_L s[i] = 0)$ }

pred esPrimo ($n:\mathbb{Z}$) {

$n \geq 2$ $\wedge$ $(\forall i:\mathbb{Z})(1 < i < n \rightarrow_L n \bmod i \neq 0)$ }

¿si $|s| = 0$? Sí, tiene sentido

NOTA

HOJA N° 6

[redacted] [redacted] [redacted]

Dinero es un float.

Planta es un String.

Cantidad es un int

TAD Vivero {

obs dineroActual : Dinero;

obs catalogo: Dict < Planta : ⟨ Dinero, Cantidad ⟩ >;

Proc abrirVivero (in d : Dinero) : Vivero

requiere : { d > 0 }

asegura: { res.dineroActual = d ∧

|res.catalogo| = 0 }

Proc dineroDisponible (in v: Vivero) : Dinero

asegura: { res = v.dineroActual }

Proc actualizarPrecio (in p: Planta, in d: Dinero, inout v: Vivero)

requiere : { d > 0 ∧ p in old(v).catalogo }

asegura: { v.dineroActual = old(v).dineroActual }

asegura: { v.catalogo = setKey(old(v).catalogo, p, ⟨ d, old(v).catalogo[p]$_1$ ⟩) }

NOTA

proc vender (in p: Planta, inout v: Vivero)

requiere: { p in old(v).catalogo $\wedge_L$

old(v).catalogo[p]$_1$ > 0 * ~~old(v).catalogo[p]$_0$ ≠ 0~~ }

asegura: { v.dineroActual = old(v).dineroActual

+ old(v).catalogo[p]$_0$ }

asegura: { v.catalogo = setKey (old(v).catalogo,

p, (old(v).catalogo[p]$_0$, old(v).catalogo[p]$_1$ - 1)) }

proc comprarPlanta (in p: Planta, in d: Dinero, in n: cantidad,

inout v: Vivero)

requiere: { d > 0 ^ n > 0 ^ old(v).dineroActual ≥ n·d }

asegura: { v.dineroActual = old(v).dineroActual - n·d }

asegura: { (¬ p in old(v).catalogo ^

v.catalogo = setKey (old(v).catalogo, p, (d*, n))) v

(p in old(v).catalogo $\wedge_L$ v.catalogo =

setKey (old(v).catalogo, p, (old(v).catalogo[p]$_0$,

old(v).catalogo[p]$_1$ + n))) }

* ~~El 0 representa que no tiene precio asignado~~

· Los puntos 1 y 2 fueron resueltos con el TAD.

Ignorar esto

* Esto es la justificación: cuando compro una

planta por primera vez le pongo por default

el precio que me costó. Total, lo puedo actualizar.

NOTA

3) Si supiera que cada vez que compro en el mayorista tengo que pagar exactamente 10% más que la vez anterior, significa que luego de la primera compra de un tipo de planta podria predecir el monto que pagaría la próxima vez. Luego, el parametro que representa el precio solo sería necesario para la primera vez. Quizás optaría por tener un observador que hable sobre el precio de compra de las plantas y que se actualice con cada nueva compra. Un diccionario me parece una buena opción.

NOTA

HOJA N° 2

[redacted] [redacted] FECHA [redacted]

3)

```
if i mod 2 = 0 then
        s[i] = 2 · s[i]      S1
else
        s[0] = 3             S2
end
```

$Q \equiv (\forall j:\mathbb{Z})(0 \leq j < |s| \rightarrow_L s[j] \bmod 2 = 0)$

$B \equiv i \bmod 2 = 0$

Quiero calcular la wp de:

$wp(\text{if } B \text{ then } S1 \text{ else } S2 \text{ fi}, Q) \equiv E$

Calcular dicha wp es equivalente por el axioma 4 a:

$E \equiv def(B) \wedge_L ((B \wedge wp(S1,Q)) \vee (\neg B \wedge wp(S2,Q)))$

$def(B) \equiv True$ (uso que $True \wedge A \equiv A$) → Esto es más adelante *

$\neg B \equiv i \bmod 2 \neq 0$

Voy a calcular las dos wp.

$wp(S1,Q) \equiv wp(s[i] := 2 \cdot s[i], Q)$

$\equiv wp(s := setAt(s,i,2 \cdot s[i]), Q)$

$\equiv def(setAt(s,i,2 \cdot s[i])) \wedge_L Q^{s}_{setAt(s,i,2 \cdot s[i])}$

$\equiv 0 \leq i < |s| \wedge_L (\forall j:\mathbb{Z})(0 \leq j < |setAt(s,i,2 \cdot s[i])| \rightarrow_L$
$setAt(s,i,2 \cdot s[i])[j] \bmod 2 = 0) \equiv E1$

Voy a usar que: (1)

$|s| = |setAt(s,i,2 \cdot s[i])|$

NOTA

- $setAt(s, i, 2 \cdot s[i])[j] = \begin{cases} 2 \cdot s[i] & \text{si } j = i \\ s[j] & \text{si } j \neq i \end{cases}$

Obs: j e i están en rango

· Con estas dos propiedades mencionadas voy a dividir el para todo en dos.

$E1 \equiv 0 \leq i < |s| \wedge_L (\forall j : \mathbb{Z})((0 \leq j < |s| \wedge j \neq i) \rightarrow_L s[j] \bmod 2 = 0)$
$\wedge (\forall k : \mathbb{Z})((0 \leq k < |s| \wedge k = i) \rightarrow_L setAt(s, i, 2 \cdot s[i])[k] \bmod 2 = 0)$

· Recuerdo que:

$setAt(s, i, 2 \cdot s[i])[i] = 2 \cdot s[i]$

$\cancel{E1 \equiv 0 \leq i < |s| \wedge_L (\forall j : \mathbb{Z})((0 \leq j < |s| \wedge j \neq i) \rightarrow_L s[j] \bmod 2 = 0)}$
$\cancel{\wedge (\forall k : \mathbb{Z})((0 \leq k < |s| \wedge k = i \rightarrow_L 2 \cdot s[i] \bmod 2 = 0)}$

· ~~Pude~~ puedo hacer el reemplazo porque si tengo que mi antecedente es falso, la implicación siempre es verdadero. Por lo tanto, me interesa analizar lo que pasa cuando este es verdadero. Y esto es equivalente a ver que ocurre con $2 \cdot s[i] \bmod 2 = 0$.

$E1 \equiv 0 \leq i < |s| \wedge_L (\forall j : \mathbb{Z})((0 \leq j < |s| \wedge j \neq i) \rightarrow_L s[j] \bmod 2 = 0)$
$\wedge \ 2 \cdot s[i] \bmod 2 = 0$

· Se puede notar que $2 \cdot s[i] \bmod 2 = 0$ siempre es verdadero ($s[i]$ está definido). Luego, me queda algo del estilo $A \wedge B \wedge True \equiv A \wedge B$

· $E1 \equiv 0 \leq i < |s| \wedge_L (\forall j : \mathbb{Z})((0 \leq j < |s| \wedge j \neq i) \rightarrow_L s[j] \bmod 2 = 0)$

· Ya calculé la primera wp. Ahora continuo con la otra.

NOTA

$WP(S2, Q) \equiv WP(s[0] := 3, Q) \equiv wp(s := setAt(s,0,3), Q)$

$\equiv def(setAt(s,0,3)) \wedge_L Q^{s}_{setAt(s,0,3)}$

$\equiv 0 \le 0 < |s| \wedge_L (\forall j:\mathbb{Z})(0 \le j < |setAt(s,0,3)| \to_L$

$setAt(s,0,3)[j] \bmod 2 = 0) \equiv E2$

Vuelvo a usar las ideas de (1) para reemplazar y dividir en casos.

$E2 \equiv 0 < |s| \wedge_L (\forall j:\mathbb{Z})((0 \le j < |s| \wedge j \neq 0) \to_L s[j] \bmod 2 = 0)$

$\wedge (\forall i:\mathbb{Z})((0 \le i < |s| \wedge i = 0) \to_L setAt(s,0,3)[i] \bmod 2 = 0)$

Observar que ~~devuelta~~ de vuelta nos interesa analizar en el segundo para todo que ocurre cuando el antecedente es verdadero. Esto es equivalente a ver si

$setAt(s,0,3)[0] \bmod 2 = 0 \equiv 3 \bmod 2 = 0 \equiv False$

Luego, mi conjunción queda de la forma:

$A \wedge B \wedge False \equiv False$ (A ∧ B no se indefine)

$E2 \equiv False$

Ahora tengo que mi WP es equivalente a:

ACÁ * $E \equiv True \wedge ((B \wedge E1) \vee (\neg B \wedge E2))$

$\equiv (B \wedge E_1) \vee (\neg B \wedge False)$

$\equiv (B \wedge E1) \vee False \equiv B \wedge E1$

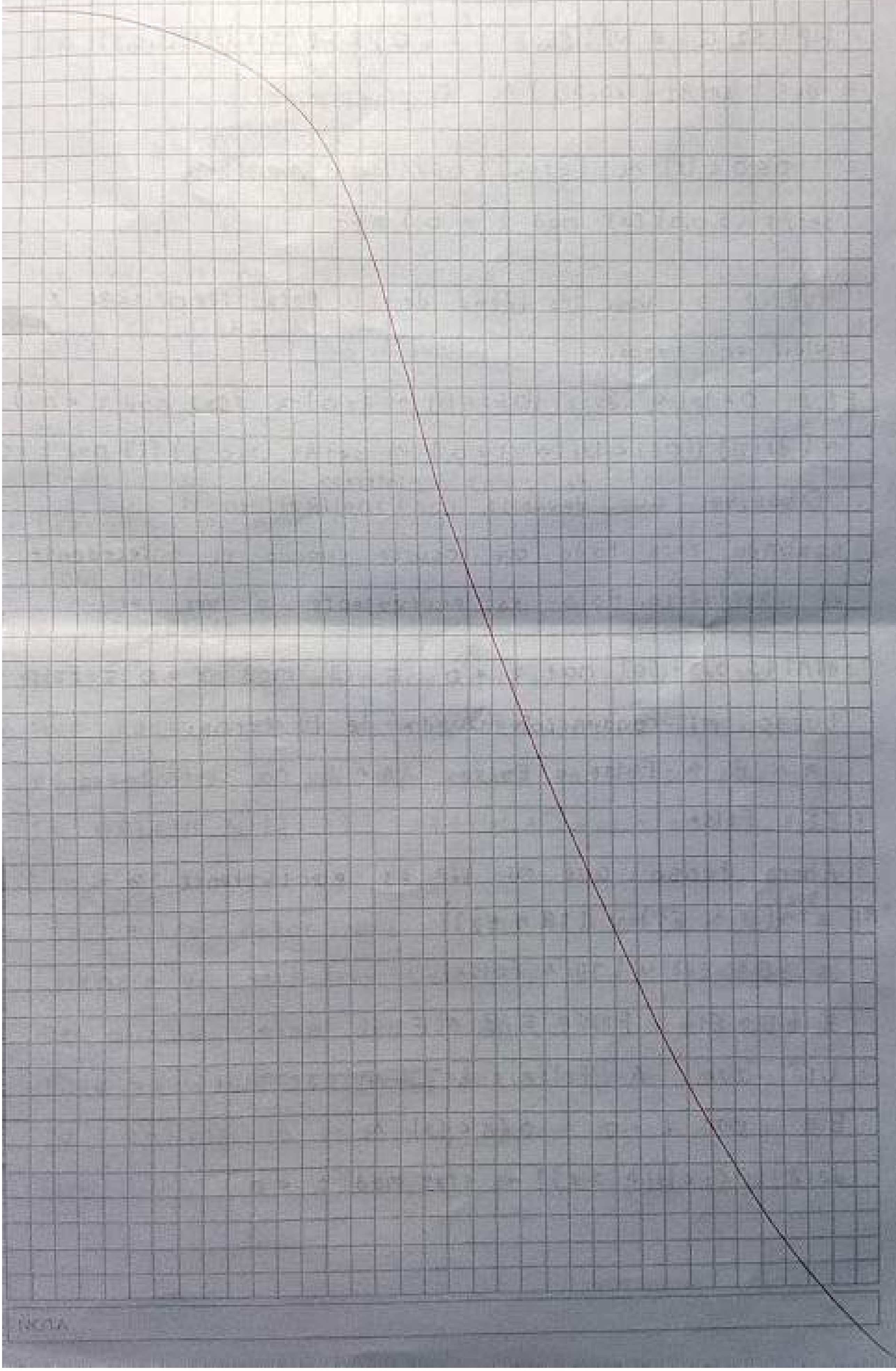
Usé que $A \vee False \equiv A$ ~~(A no se indefi~~

$E \equiv i \bmod 2 = 0 \wedge 0 \le i < |s| \wedge_L$

$(\forall j:\mathbb{Z})((0 \le j < |s| \wedge j \neq i) \to_L s[j] \bmod 2 = 0)$

NOTA

Finalmente, calculé la wp que estaba buscando.



NOTA

4) While $i \geq 0$ do

$\quad res := res + s[i] + 1$

$\quad i := i - 1$

end

$P_c \equiv i = |s| - 1 \wedge res = 0$

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

a) $I \equiv -1 \leq i \leq |s| - 1 \wedge res = \sum_{j=i+1}^{|s|-2-\boxed{i}} (s[j] + 1)$

$f_v \equiv i + 1$

Ya propuse mi invariante y mi función variante.

b) Voy a demostrar los pasos indicados para la correctitud de un ciclo.

1) $P_c \Rightarrow I$

Asumo que P_c es verdadera y pruebo que I también lo tiene que ser.

- $-1 \leq i \leq |s| - 1$: esto es verdadero ya que $i = |s| - 1$
- $res = \sum_{j=i+1}^{|s|-2-i} (s[j] + 1)$ uso que $res = 0$ y que ↓

$$0 = \sum_{j=|s|}^{|s|-2-|s|+1} (s[j]+1) \equiv 0 = \sum_{j=|s|}^{-1} (s[j] + 1)$$

y como $0 \leq |s|$, la sumatoria tiene índices fuera de rango y eso significa que da 0. Luego, la igualdad de res también es verdadera.

NOTA

· Luego, probé que $P_c \Rightarrow I$

ii) $I \wedge \neg B \Rightarrow Q_c$

· $I \wedge \neg B$ es mi hipótesis (supongo que es verdadera) y pruebo que Q_c también lo es.

· $\neg B \equiv i \leq -1$

$$I \wedge \neg B \equiv -1 \leq i \leq |s|-1 \wedge i \leq -1 \wedge res = \sum_{j=i+1}^{|s|-2-i} (s[j]+1)$$

$$\equiv i = -1 \wedge res = \sum_{j=i+1}^{|s|-2-i} (s[j]+1)$$

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

Como $I \wedge \neg B$ es verdadero puedo saber que:

$$res = \sum_{j=-1+1}^{|s|-2+1} (s[j]+1) \equiv res = \sum_{j=0}^{|s|-1} (s[j]+1)$$

· Reemplazo a este res en (1)

$$\sum_{j=0}^{|s|-1} (s[j]+1) = |s| + \sum_{j=0}^{|s|-1} s[j]$$

· Puedo dividir a la primera sumatoria:

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

· Uso que: $\sum_{j=0}^{|s|-1} 1 = \sum_{j=1}^{|s|} 1 = |s|$

NOTA

CamScanner

y con este resultado llego a que:

$|s| + \sum_{j=0}^{|s|-1} s[j] = |s| + \sum_{j=0}^{|s|-1} s[j]$ lo cual

es siempre verdadero

Luego, probé que $I \wedge \neg B \Rightarrow Q_c$

III) $I \wedge f_v \leq 0 \Rightarrow \neg B$

Supongo que $I \wedge f_v \leq 0$ es verdadero y pruebo que $\neg B$ también lo es.

$I \wedge f_v \leq 0 \equiv -1 \leq i \leq |s|-1 \wedge res = \sum_{j=i+1}^{|s|-2-i} (s[j]+1)$

$\wedge \; i+1 \leq 0 \equiv -1 \leq i \leq |s|-1 \wedge i \leq -1 \wedge res = \sum_{j=i+1}^{|s|-2-i} (s[j]+1)$

$\equiv i = -1 \wedge res = \sum_{j=i+1}^{|s|-2-i} (s[j]+1)$

$\neg B \equiv i \leq -1$

$i \leq -1$: como $i = -1$ es verdadero tengo que $-1 \leq -1$ lo cual es siempre verdadero.

Luego, probé que $I \wedge f_v \leq 0 \Rightarrow \neg B$

NOTA