Algoritmos y Estructuras de Datos

Primer Parcial – Sábado 7 de octubre de 2023

Nicolás

#Orden	Libreta	Apellido y Nombre	E1	E2	E3	E4	Nota Final
301	1081/22	Fuentes Urfeig Pedro	20	30	15	30	95

- 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
```

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

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] $P_c \rightarrow I$
   II) [10 pts] $(I \land \neg B) \rightarrow Q_c$
   III) [5 pts] $(I \land f_v \leq 0) \rightarrow \neg B$

Pedro Fuentes Urfeig	Computación 1088/22	Hoja 1

20

1) proc PrimosEnCero (inout $S: seq\langle \mathbb{Z} \rangle$) {

requiere $\{S = S_0 \wedge \overline{|S| > 0}\}$ → NO, PUEDE DEVOLVER [ ]

~~asegura~~ asegura $\{|S| = |S_0|\}$ ✓

asegura $\{(\forall j:\mathbb{Z})(0 \leq j < |S| \wedge esPrimo(j) \Rightarrow_L S[j] = 0)\}$ ✓

asegura $\{(\forall j:\mathbb{Z})(0 \leq j < |S| \wedge \neg esPrimo(j) \Rightarrow_L S[j] = S_0[j])\}$. ✓

}

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

$\neg(\exists i:\mathbb{Z})(2 \leq i < j \wedge j \mod i = 0)$ ✓

}

Pedro Fuentes Urfelig	Computación 1088/22	Hoja 2

2) a) TAD Vivero { // Especie es String, Precio es $\mathbb{N}$, Cantidad es $\mathbb{N}$

obs stock: dict<Especie, $\mathbb{Z}$>

obs precios: dict<Especie, Precio>.

obs dinero: $\mathbb{Z}$

proc nuevoVivero(in d: $\mathbb{Z}$): Vivero. {

req {$d \geq 0$}

asegura {res.stock = {} ∧ res.precios = {} ∧ res.dinero = d}

}

proc comprar(inout α: Vivero, in e: Especie, in c: Cantidad, in p: Precio) {

requiere {$c * p \leq \alpha$.dinero} ∧ $p>0$. ESTÁ BIEN.

asegura {((e in α_0.stock $\wedge_L$ α.stock = setKey(α_0.stock, e, α_0.stock[e] + c) ∧
α.dinero = α_0.dinero − $c*p$) ∨

(¬(e in α_0.stock) $\wedge_L$ α.stock = setKey(α_0.stock, e, c) ∧ α.dinero = α_0.dinero − $c*p$))}

asegura {α.precios = α_0.precios}.

}

proc FijarPrecio(inout α: Vivero, in e: Especie, in p: Precio) {

requiere {e in α.stock}.

asegura {α.dinero = α_0.dinero ∧ α.stock = α_0.stock}.

~~asegura {(e in α_0.precios $\wedge_L$ α.precios = setKey(α_0.precios, e, α_0.precios[e]~~

asegura {α.precios = setKey(α_0.precios, e, p)}.

proc Vender(inout d: Vivero, in e: Especie) {

requiere $\{(e \text{ in } d.stock \land_L d.stock[e] \geq 1) \land e \text{ in } d.precios\}$.

asegura $\{d.precios = d_0.precios\}$.

asegura $\{d.dinero = d_0.dinero + d_0.precios[e]\}$.

asegura $\{d.stock = setKey(d_0.stock, e, d_0.stock[e] - 1)\}$.

}.

proc BalanceDeCaja(in d: Vivero): $\mathbb{Z}$ {

asegura $\{res = d.dinero\}$.

}.

}

2) 3) Si el precio aumentase en 10% cada vez que compro, cambiaría mi proc comprar. Antes no le pasaría el nuevo precio, pero cambiaría mi obs stock. En vez de guardar la cantidad, guardaría una tupla que determine la cantidad actual y el último precio pagado. ~~También~~

En el proc comprar, tendría que cambiar en el requiere que mi dinero actual sea mayor o igual a la cantidad multiplicada por 1.1, multiplicada por el último precio pagado.

Pedro Fuentes Urfeig | Computación 1088/22 | Hoja 3.

3) Tomo S como todo el código y $D \equiv i \bmod 2 = 0$.

$$wp(S, Q) \equiv wp(\text{if } D \text{ then } S[i] := 2 * S[i] \text{ else } S[0] := 3,\ Q) \equiv$$

$$\equiv \left(D \wedge_L 0 \leq i < |s| \wedge_L Q^{S}_{setAt(S, i, 2*S[i])}\right) \vee \left(\neg D \wedge_L 0 \leq i < |s| \wedge_L Q^{S}_{setAt(S, 0, 3)}\right).$$

Analizo el primer caso, donde vale D.

$$D \wedge_L 0 \leq i < |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow_L (i = j \wedge_L \underbrace{2 * S[i] \bmod 2 = 0}_{\text{True}}) \vee (i \neq j \wedge_L S[j] \bmod 2 = 0))$$

Descarto el caso $i = j$ junto que da True

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

Analizo el caso donde vale $\neg D$.

$$\neg D \wedge_L 0 \leq i < |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |s| \rightarrow (j = 0 \wedge_L \underbrace{3 \bmod 2 = 0}_{\text{False}}) \vee (j \neq 0 \wedge_L S[j] \bmod 2 = 0)).$$

Dado que el caso $j = 0$ es Falso, necesito que $j \neq 0$ sea True para que valga el consecuente. Pero no no puede ser True, ya que $j = 0$ es un valor válido para j según el rango del $\forall$. Por ende, el término del $\forall$ es Falso, haciendo que el caso de $\neg D$ sea falso.

Luego, vuelvo al la wp original

$$wp(S, Q) \equiv i \bmod 2 = 0 \wedge 0 \leq i < |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |s| \wedge i \neq j \rightarrow_L S[j] \bmod 2 = 0).$$

ESTÁ BIEN PERO TE SALTEASTE LOS PASOS DEL SETAT

Pedro Fuentes Urfeig | Computación 18/8/22. | Hoja 4.

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

$fv = i+1$

b) i) $P_c \Rightarrow I$. Asumo que el precedente es cierto y reemplazo.

$i = |s|-1 \wedge res = 0 \Rightarrow \underbrace{-1 \leq |s|-1 \leq |s|-1}_{True.} \wedge 0 = \sum_{j=|s|}^{|s|-1} (s[j]+1).$

Puesto que la sumatoria está fuera de rango, vale automáticamente cero. Luego, vale la implicación.

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

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

$\underbrace{i = -1 \wedge res = \sum_{j=0}^{|s|-1} (s[j]+1)}_{} \Rightarrow \underbrace{res = \sum_{j=0}^{|s|-1} (s[j]+1)}_{}.$ ⊛

$I \wedge \neg B \Rightarrow True.$

⊛ Puedo hacer esta reescritura ya que entre 0 y |s|-1 hay exactamente |s| números. Luego, es lo mismo escribir +|s| fuera de la sumatoria que escribir un +1 adentro de esta.

iii) $I \wedge i+1 \leq 0 \Rightarrow i < 0$

$I \wedge i \leq -1 \Rightarrow i < 0.$

$I \wedge i \leq -1 \Rightarrow True.$ (puesto que $i \in \mathbb{Z}$)

BIEN!

Scanned with CamScanner