



1.1	1.2	2.1	2.2	3	4.1	4.2
1	0.5	1.8	1.8	2	4	0.9

Apellido ANDRES ..... Nombre SEBASTIAN .....  
LU 1020122 ..... Turno NOCHE .....  
Cant. de hojas entregadas (sin contar ésta) 6

(9) SEBAS

El parcial se aprueba con 5 puntos. Entregar cada ejercicio en hoja separada. No se permite consultar ningún material durante el examen.

### Ejercicio 1. 2 puntos

#### 1. [1 punto]

Completar en las siguientes especificaciones nombres adecuados para el problema *a*, los parámetros *b* y *c*, y las etiquetas *x*, *y*, *z*, *u* y *w*.

```

problema a (in b: seq(Char × Char), in c: seq(Char) ) : seq(Char) {
    requiere x: { (forall i, j : Z)((0 ≤ i < |b| ∧ 0 ≤ j < |b| ∧ i ≠ j) → b[i]₀ ≠ b[j]₀)
    requiere y: { (forall i, j : Z)((0 ≤ i < |b| ∧ 0 ≤ j < |b| ∧ i ≠ j) → b[i]₁ ≠ b[j]₁)
    requiere z: { (forall i : Z)(0 ≤ i < |c| → (exists j : Z)(0 ≤ j < |b| ∧ b[j]₀ = c[i]))}
    asegura u: { |resultado| = |c| }
    asegura w: { (forall i : Z)(0 ≤ i < |c| → (exists j : Z)(0 ≤ j < |b| ∧ b[j]₀ = c[i] ∧ b[j]₁ = resultado[i])) }
}

```

#### 2. [1 punto]

Especificar el siguiente problema (se puede especificar de manera formal o semi-formal):  
Dados los inputs *b*: seq(Char × Char), *m*: seq(seq(Char)) y *n*: seq(seq(Char)), retornar verdadero si *n* es igual al resultado de aplicar el problema *a* (del punto 1.1) a cada elemento de la secuencia *m*.

### Ejercicio 2. 4 puntos

#### 1. [2 puntos]

Programar en Haskell una función que satisfaga la especificación del problema *a* del Ejercicio 1. Recordá escribir los tipos de los parámetros.

#### 2. [2 puntos]

Programar en Python una función que satisfaga la especificación del problema *a* del Ejercicio 1. Recordá escribir los tipos de los parámetros y variables que uses en tu implementación.

### Ejercicio 3. 2 puntos

Sea la siguiente especificación del problema aprobado y una posible implementación en lenguaje imperativo.

problema aprobado (in notas: seq(Z)) : Z {

```

    requiere: {notas > 0}
    requiere: {(forall i : Z)(0 ≤ i < |notas| → 0 ≤ notas[i] ≤ 10)}
    asegura: {result = 1 ↔ todos los elementos de notas son mayores o iguales a 4 y el promedio es mayor o igual a 7}
    asegura: {result = 2 ↔ todos los elementos de notas son mayores o iguales a 4 y el promedio está entre 4 (inclusive) y 7}
    asegura: {result = 3 ↔ alguno de los elementos de notas es menor a 4 o el promedio es menor a 4}
}

```

```

def aprobado(notas: list[int]) -> int:
L1:     suma_notas: int = 0
L2:     i: int = 0
L3:     while i < len(notas):
L4:         if notas[i] < 4:
L5:             return 3
L6:         suma_notas = suma_notas + notas[i]
L7:         i = i + 1
L8:     if suma_notas >= 7 * len(notas):
L9:         return 2
L10:    else:
L11:        if suma_notas > 4 * len(notas):
L12:            return 2
L13:        else:
L14:            return 3

```

1. Dar el diagrama de control de flujo (control-flow graph) del programa **aprobado**.
2. Escribir un test suite que ejecute todas las líneas del programa **aprobado**.
3. Escribir un test suite que tenga un cubrimiento de al menos el 50 por ciento de decisiones ("branches") del programa.
4. Explicar cuál/es es/son el/los error/es en la implementación. ¿Los test suites de los puntos anteriores detectan algún defecto en la implementación? De no ser así, modificarlos para que lo hagan.

#### Ejercicio 4. 2 puntos

1. [1 punto] Suponga las siguientes dos especificaciones de los problemas p1 y p2:

problema p1(x:Int)=res:Int {  
 requiere A;  
 asegura C;  
 }

problema p2(x:Int)=res:Int {  
 requiere B;  
 asegura C;  
 }

Si A es más fuerte que B, ¿Es cierto que todo algoritmo que satisface la especificación p1 también satisface la especificación p2? ¿Y al revés?, es decir, ¿Es cierto que todo algoritmo que satisface la especificación p2 también satisface la especificación p1? Justifique.

2. [1 punto] ¿Es posible que haya un test suite con 100 % de cubrimiento de nodos que todos los test pasen pero que igual el programa tenga un bug? Justifique.

Ej. 1

SEBASTIÁN ANDRÉS

1020/22  
Hoja 1

Ej. 1)

a) OBSERVACIONES:

- a recibe una secuencia de tuplas  $(\text{char} \times \text{char})$  (b) y una secuencia de CHAR (C), y devuelve otra secuencia de char.
- repuliere x: Dice que las tuplas en (B) son distintas entre sí en la primer componente.
- repuliere y: Dice que las tuplas en (B) son distintas entre sí en la segunda componente.
- repuliere z: Exige que todos los chars de (C) estén como primer componente de ALGUNA tupla en (B).
- Asegura u: la longitud del resultado y de (C) es la misma.
- Asegura w: Dice que resultado[i] es la segunda componente de la tupla en (B) que tiene como primer componente a c[i].
- ~ Negro, podemos reescribir la especificación de la sig forma...

~~Problema de la función LeerSiguienteComponente~~

•)

problema LeerSiguienteComponente (in L:  $\text{sep} < \text{char} \wedge \text{char} \rangle$ , in IDs:  $\text{sy} < \text{char} \rangle$ ):  $\text{sep} < \text{char} \rangle$  q  
Requiere diferentes componentes

$$(\forall i, j : \mathbb{Z}) ((0 \leq i < |L| \wedge 0 \leq j < |L| \wedge i \neq j) \rightarrow \&[i]_o \neq \&[j]_o)$$

}

Requiere diferentes componentes

$$(\forall i, j : \mathbb{Z}) ((0 \leq i < |L| \wedge 0 \leq j < |L| \wedge i \neq j) \rightarrow \&[i]_o \neq \&[j]_o)$$

}

Requiere todo Id de C en B

$$(\forall i : \mathbb{Z}) (0 \leq i < |\text{ids}| \rightarrow (\exists j : \mathbb{Z}) (0 \leq j < |L| \wedge \&[j]_o = \text{ids}[i]))$$

}

ASEGURA misma tamaño }  $|\text{resultado}| = |\text{ids}|$  }

ASEGURA devuelve 2 componentes }

$$(\forall i : \mathbb{Z}) (0 \leq i < |\text{ids}| \rightarrow (\exists j : \mathbb{Z}) (0 \leq j < |L| \wedge \&[j]_o = \text{ids}[i] \wedge \&[j]_i = \text{res}[i]))$$

}

}

• EN MI CASO, INTERPRETE' QUE (B) era --.

$$B = [(\text{id}_0, \text{valor}_0), \dots, (\text{id}_N, \text{valor}_N)]$$

✓

LEGO NAME A  $C = \text{ids}$  y la función (a) tambien  
PODRIA LLAMARSE "LEER VALORES".

B

PUNTO B



EJ 4

SEBASTIÁN ANDRÉS

1028/22  
HOJA 2

(B) ME A DE ESPECIFICAR:

b :  $\text{sep} < (\text{char} \times \text{char})$

m :  $\text{sep} < \text{sp} < \text{char}$

m :  $\text{sp} < \text{sp} < \text{char}$



true  $\Leftrightarrow$  n es el resultado de aplicar A a los elementos de m.

- PROBLEMA Son los Valores ( $b : \text{sep} < (\text{char} \times \text{char})$ ;  $m, m' : \text{sep} < \text{sp} < \text{char}$ ):  
Requiere  $\{ \text{true} \}$  no está teniendo en cuenta  
ASEGURA  $\{$  (o) requiere de LeerSegundaComponente  
 $\text{res} = \text{true} \Leftrightarrow |\text{m}| = |\text{m}'| \}$   $\wedge$  Son 2 Componente ( $\text{m} \neq \text{m}'$ ) ✓  
 $\}$
- pred son2Componente ( $m, m' : \text{sp} < \text{sp} < \text{char}$ )  $\{$   
 $(\forall i : \mathbb{Z}) (0 \leq i < |\text{m}| \wedge m[i] = \text{LeerSegundaComponente}(b, m'[i]))$  ✓  
 $\}$
- Donde LeerSegundaComponente ( $b, \text{ids}$ ) es el (a) especificado  
en el punto anterior. ✓ R

Ej 2

SEBASTIAN AVONÉS

10/20/22.  
no, A 3

Ej 2) i- Programar en Haskell (a)

ii- Programar en Python (a) :

i)

$a :: [(char, char)] \rightarrow [char] \rightarrow [char]$

$a \underset{\text{Valores}}{[ } [ ] = [ ]$

B-

$a \text{ VALORES } (x : xs) = \text{segComp } x \text{ VALORES} : a \text{ VALORES } xs$

$\text{segComp} :: \text{char} \rightarrow [(char, char)] \rightarrow \text{char}$

$\text{segComp } n [ ] = \text{undefined} \quad \text{-- ASUMO POR ESPECIFICACIÓN QUE NUNCA SUCDE}$

$\text{segComp } n ((x:r) : xs) \quad | \quad n == x = v$

| otherwise =  $\text{segComp } n xs$

ii) DEF  $a(b:List[(char, char)], c: List[char]) \rightarrow List[char] ::$

# ASUMO EL REFERENCIAL ( $\exists i. id \in C \rightarrow (f t : b) (t_i == id)$ )

out =  $List[char] = list()$

for id in C:

for  $(i, value)$  in b:

if  $i == id$ :

out.append(value)

break

B-

PASS X

RETURN OUT

## OBSERVACIONES

[(char, char)]

i) sapComp :: char → ~~char~~ → char es una auxiliar que devuelve la segunda componente de la tupla en la lista de tuplas dada, pero tiene como primer componente al char dado.

Ej:

$$\text{sapComp } 1 [(2,3), (1,0)] \rightarrow 0$$

ii) En python aproveche la descomposición de tuplas iterando en la lista de tuplas pero tranquilamente podría haber usado un while ( $i < \text{len}(c)$ ) --.  $i + i$  o con un for  $i$  in range( $\text{len}(c)$ ). ✓

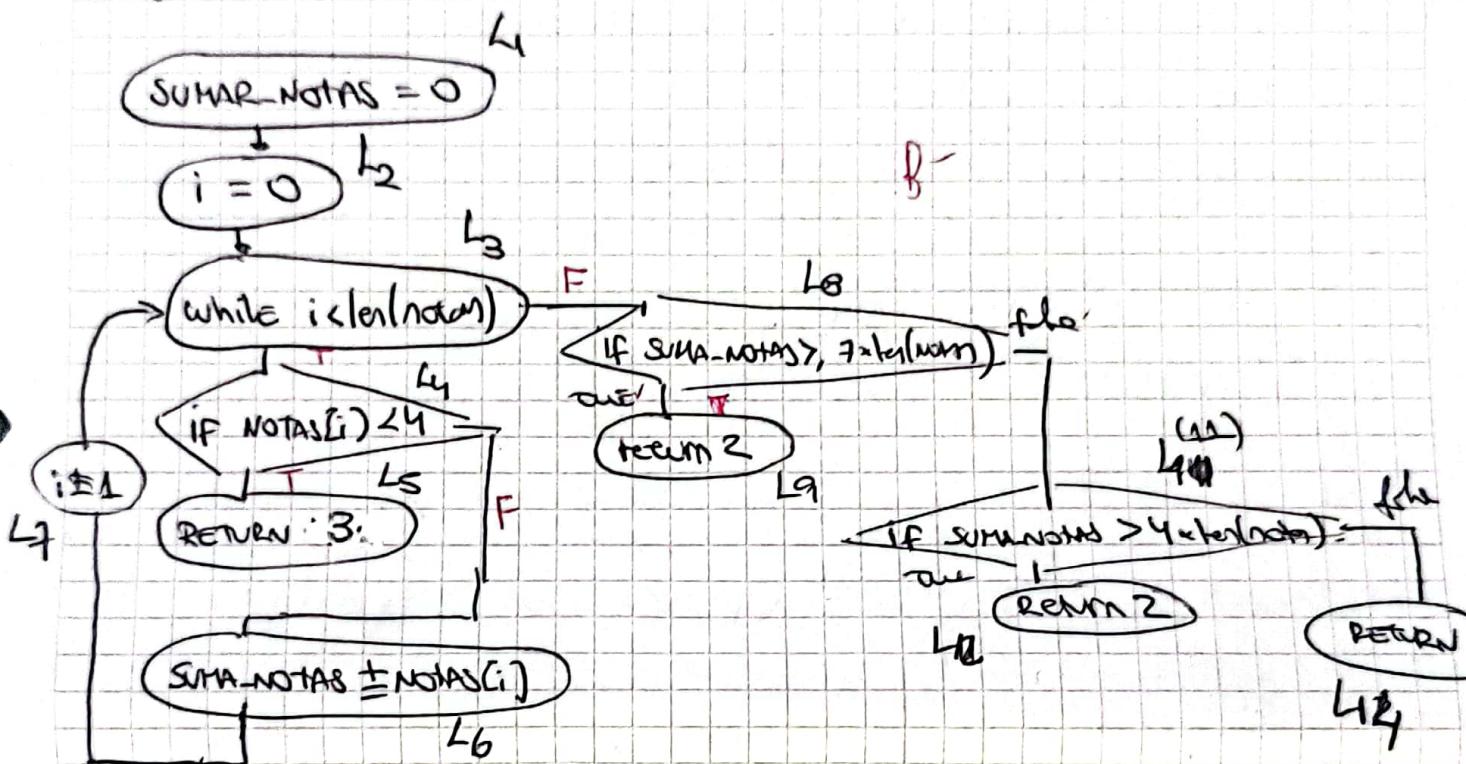
Otra observación es que Python admite escribir list en vez de list[inf] de la librería Typing o hasta no escribirlo.

Ej 3

SEBASTIÁN AVILA

10/28/22  
Nota 4

Ej 3) i) Armar CFG:



ii) Escribir test suite que pase por todos los líneas

TEST SUITE:

- ASSERT APROBADO ([3]) == 3; || Líneas: {L1, L2, L3, L4, L5} ✓
- ASSERT APROBADO ([10, 10]) == 1; || Líneas: {L1, L2, L3, L4, L5, L6, L7, L8, L9} ✓
- ASSERT APROBADO ([6, 6]) == 2; || Líneas: {L1, L2, L3, L4, L5, L6, L7, L8, L9, L10, L11} ✓
- ASSERT APROBADO ([4, 4]) == 2; || Líneas: {L1, L2, L3, L4, L5, L6, L7, L8, L9, L10, L11, L12} ✓

~ Luego, estos 4 test case pasan por todos los líneas

iii) Escribir un test suite que cubra el 100% de las Branches.

- En particular, el test suite dado en (ii) cubre el 100% de las Branches.

$$\text{Cobertura} = \frac{\#\{\text{Branches cubiertas}\}}{2\#\{\text{Branches en el programa}\}}$$

B

TEST SUITE
assert Aprobado ([3]) == 3
assert Aprobado ([2,10]) == 1
assert Aprobado ([6,6]) == 2
assert Aprobado ([4,4]) == 2

iv) Explicar el error en la implementación:

- LA LÍNEA 9 DEBERÍA DEVOLVER 1 (RETURN 1)
- LA LÍNEA 11 DEBERÍA SER (if SUMA-NOTAS >= 4\*len(notas)) .

Si, el test suite del punto A detecta ambos errores ... B

- Assert Aprobado ([10,10]) == 1 falla y devuelve 2,
- Assert Aprobado ([4,4]) == 2 falla y devuelve 3,

Ej 4

SEMANA ANDRÉS

10/28/22  
M2A S

Ej 4)

i) Dados  $P_1, P_2 \dots$

- Si  $A \subset B$  pues que  $B$ , luego ( $A \Rightarrow B$ ) es tautología, por lo que cumplir el requerimiento  $P_1$  implica cumplir el requerimiento de  $P_2 \dots$

$$\begin{array}{c} \text{B} \quad \text{(A)} \\ \text{Ej: } A: \{x \in \mathbb{N} \mid x = 1\} \\ B = \{x \in \mathbb{N} \mid x \neq 1\} \end{array}$$

- No obstante, cualquier algoritmo que satisface  $P_1$  no cumple necesariamente  $P_2$ . Porque sólo es válido la postcondición C para el "dominio" de  $P_1$  (A) que es un subconjunto del "dominio" de  $P_2$  (B).

→ En mi ejemplo:

$$\begin{array}{ccc} \text{• Problema } P_1(x: \text{int}): \exists x \{ & \text{• Problema } P_2(x: \text{int}): \exists x \{ & B \\ \text{ty } \{x = 1\} & \text{ty } \{x \neq 1\} & \\ \text{obj } \{x = 1\} & \text{obj } \{x \neq 1\} & \\ \} & \} & \end{array}$$

→ Algoritmos que satisfagan  $P_1$  no tienen ninguna garantía para  $(\forall x \in \mathbb{N})(x \neq 1)$ .

- El caso contrario ...



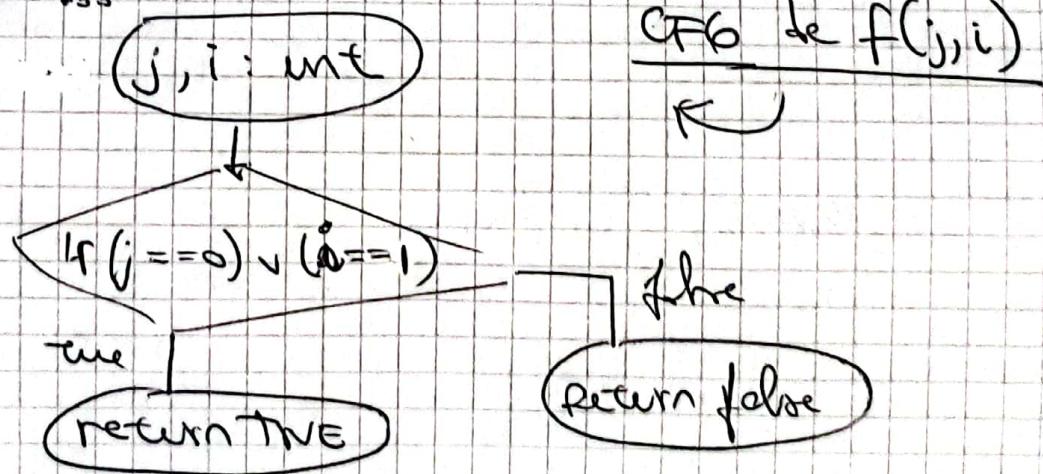
→ El caso contrario si vale, pues A es un caso particular de B. Entonces como  $P_2$  garantiza que  $\forall x \in B \rightarrow P_2$  y  $A \subseteq B$  cumplir la especificación de  $P_2$  implica cumplir la especificación de  $P_1$ .  
 (OBS: Tener más de uno Asegura C) B

---

ii) Es POSIBLE UN TEST SUITE CON 100% COBERTURA DE TODOS Y 0 FALLOS PERO CON 1 BUG?

OBS: Cubrir todos los nodos NO implica cubrir todas las DECISIONES BÁSICAS.

# por lo tanto, si es factible. A continuación 1 Ejemplo...



# TEST SUITE (todos los nodos): // Assert  $f(0, 3) == \text{true}$   
 // Assert  $f(1, 1) == \text{false}$  es  $\text{true}$

• Sobre la especificación Problema Algunos O ( $j, i : \mathbb{Z}$ ):  $\text{Bool} \{$   
 $\text{true}, \text{false} \} \dots$   
 $\{ \text{true} \} \cdot \{ \text{true} \} \Rightarrow (j=0 \vee i=0) \}$  mis propios

# Ej 4

Sebastián Andrés

1020/22  
Nº J.A. 6

Es decir, si se prueba:

El test suite cubre todos los nodos del programa que implementa la función  $f(j,i)$  y no falla.

Sin embargo  $f(x,0) \neq \text{true}$ , luego no cumple la especificación alguna  $\exists (x,y)$ .

Por lo tanto hay una falla pues el test suite pasado (cubrir todos los nodos) no encontró.

Programa en Python:

```
DEF f(j:int, i:int) → Bool
    IF (j==0) or (i==1):
        RETURN true
    ELSE:
        RETURN false
```

Test Suite (Nodos)

Assert  $f(0,3) == \text{true}$

Assert  $f(1,1) == \text{false}$

(Test Agregado)

Assert  $f(1,0) == \text{true}$

↳ falla

Especificación

Probar la función  $\exists (x,y) \rightarrow \text{Bool}$   
definida así

O sea  $\exists f \text{ es } \text{true} \Leftrightarrow (x=0 \vee y=0)$

{

Este es el ejemplo pero bien visto

B