

1.2)

Parámetros

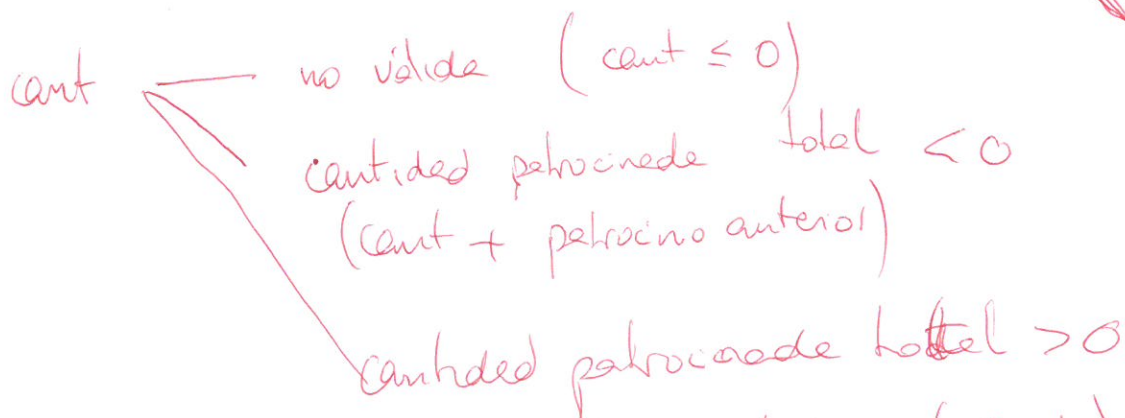
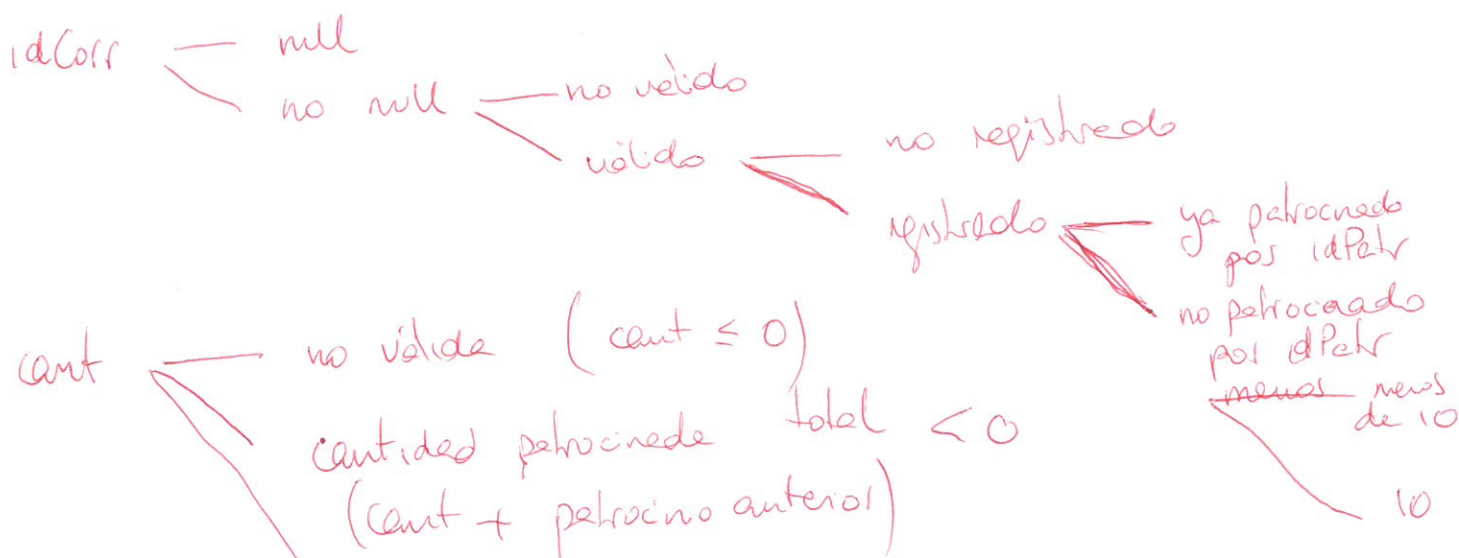
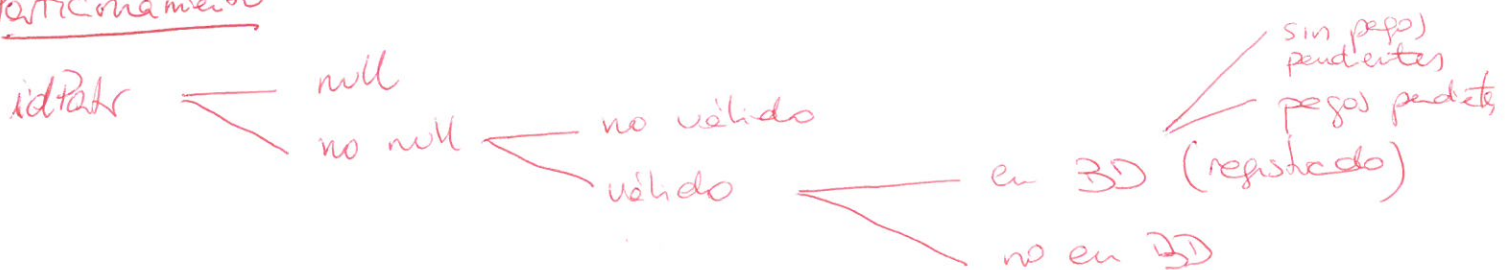
String idPatr } String de 4 caracteres alfanuméricos
String idCorr }
int cant } entero & con la cantidad a patrocinarse

carretera activa (con info sobre su longitud y corredores inscritos a ella)

BD con info sobre

- patrocinadores (cuántos corredores ha patrocinado para cada carretera y en cuánto dinero)
- corredores (cantidad con la que ha sido patrocinado en cada carretera)

Particionamiento



carretera activa con una longitud válida (> 1)

BD con datos válidos (suponemos que el resto de ops sobre la BD la dejan en un estado consistente)

1.9) cont. 1

Restricciones

- Si cualquiera de los parámetros de tipo String es null o no válido no es necesario considerar el resto de parámetros
- Si patrocinador o corredor no registrados no es necesario considerar más casos
- Si patrocinador tiene pagos pendientes no es necesario considerar más casos

Casos resultantes de combinaciones

Carrera	BD	idPatr	idCorr	cant	Salida
"CA19"	BD	null	—	—	Error Parámetros no válidos
		—	null	—	
		—	—	≤ 0	
			3 0 2		
"CA19"	BD	registrado sin pagos pendientes "PAG3" con < 10 patrocin.	registrado no patrocinado anteriormente por "PAG3" "CO31"	cant $<$ máx para "CA19"	Patrocinio realizado (a)
"CA19"	BD	registrado con pagos pendientes	—	—	Error Pagos pendientes
"CA19"	BD	registrado sin pagos pendientes "PAG3"	registrado y ha sido anteriormente por "PAG3"	cant + patrocinio anterior $>$ max	Patrocinio no realizado. (b)

1.a/cont 2

Sea la siguiente BD (sistema)

- carrera activa "CA01" de 10 kms
- corredores "CO01", "CO02", ... "CO12" registrados en "CA01"
- patrocinadores "PA01" y "PA02" sin pagos pendientes
 - "PA01" patrocina a "CO01" ... "CO10" con 100 a cada uno
 - "PA02" patrocina a "CO01" con 1000
- patrocinador "PA03" tiene pagos pendientes

Casos de prueba

(a) addPatrocinio ("PA02", "CO02", 100)

Comprobamos que el patrocinio se ha realizado correctamente:

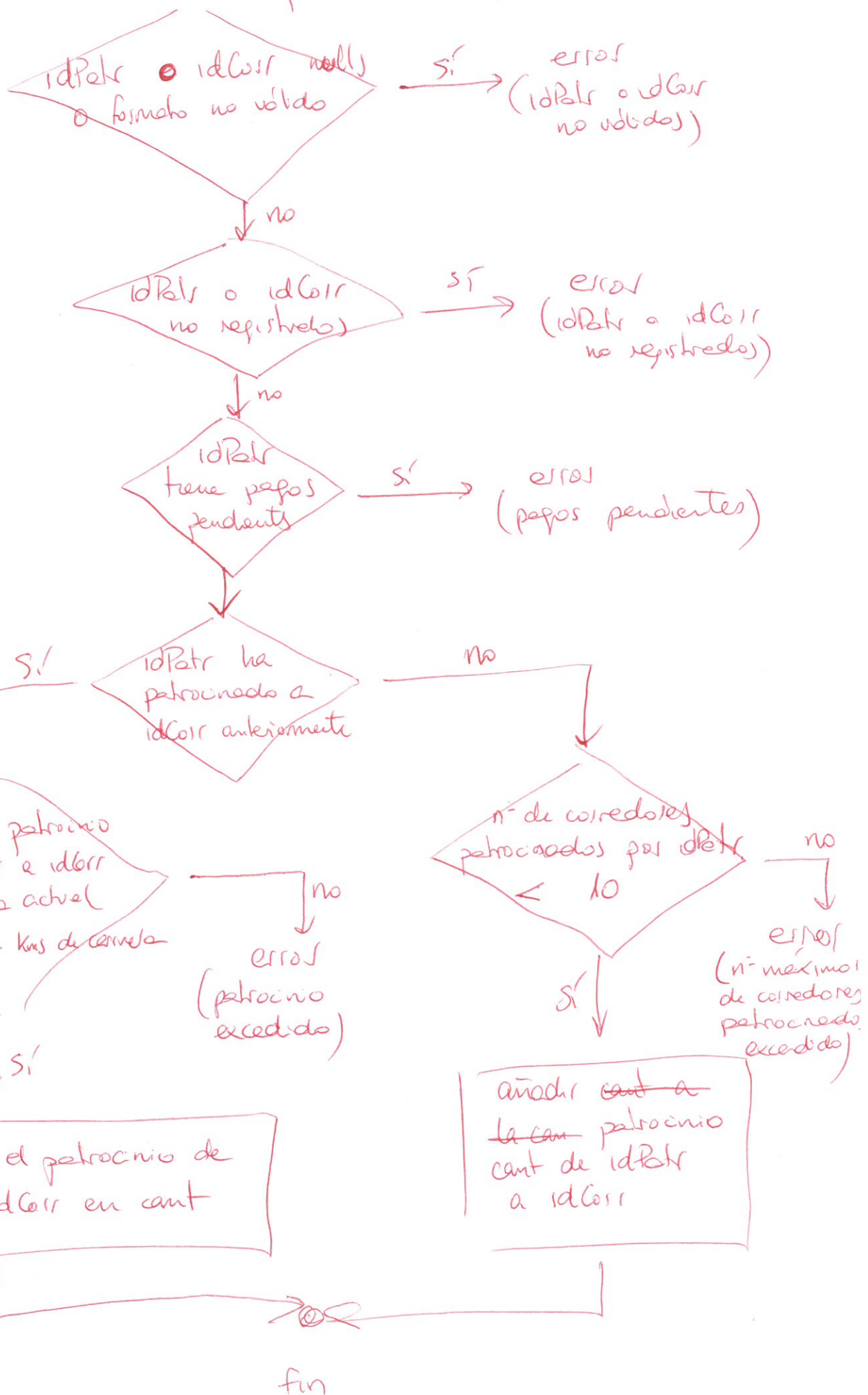
- "PA02" tiene pendiente el pago de 100
- "CO02" tiene el patrocinio de 100

(b) addPatrocinio ("PA02", "CO01", 6000)

~~Comprobamos que~~
~~se tiene~~ Patrocinio no realizado, según

1. b)

addPatrocinio (String idPatr, String idCorr, int cant)



1.b) Casos de Prueba

Dada la BD de 1.a)

ECP1: idPtr no válido

addPatrocinio (null, "COO1", 100)

Excepción no válida

ECP2 idPTr no registrado

addPatrocinio ("PA04", "COO1", 100)

Excepción no registrada

ECP3 idPTr con pagos pendientes

addPatrocinio ("PA03", "COO1", 100)

Excepción pago pendiente

ECP4 (a) de 1.a)

ECP5 Patrocinio excedido con patrocinio anterior

addPatrocinio ("PA02", "COO1", 6000)

Excepción patrocinio excedido

ECP6 Patrocinio correcto, no patrocinio anterior,
menos de 10 patrocinados

addPatrocinio ("PA02", "COO2", 100)

Comprobaciones como en (a) de 1.a)

ECP7 (b) de 1.a)

2.2) Condiciones simples

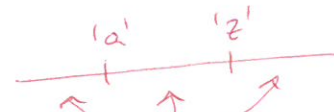
$ch \geq 'a'$

$ch \leq 'z'$

$ch == '\backslash'$

$word.trim().length() \leq 5$

$index < str.length$


Necesitamos caracteres de
cada uno de estos, menos,
y no '\'

Este condición se hará
false y esto por
cualquier cadena de
longitud < 5

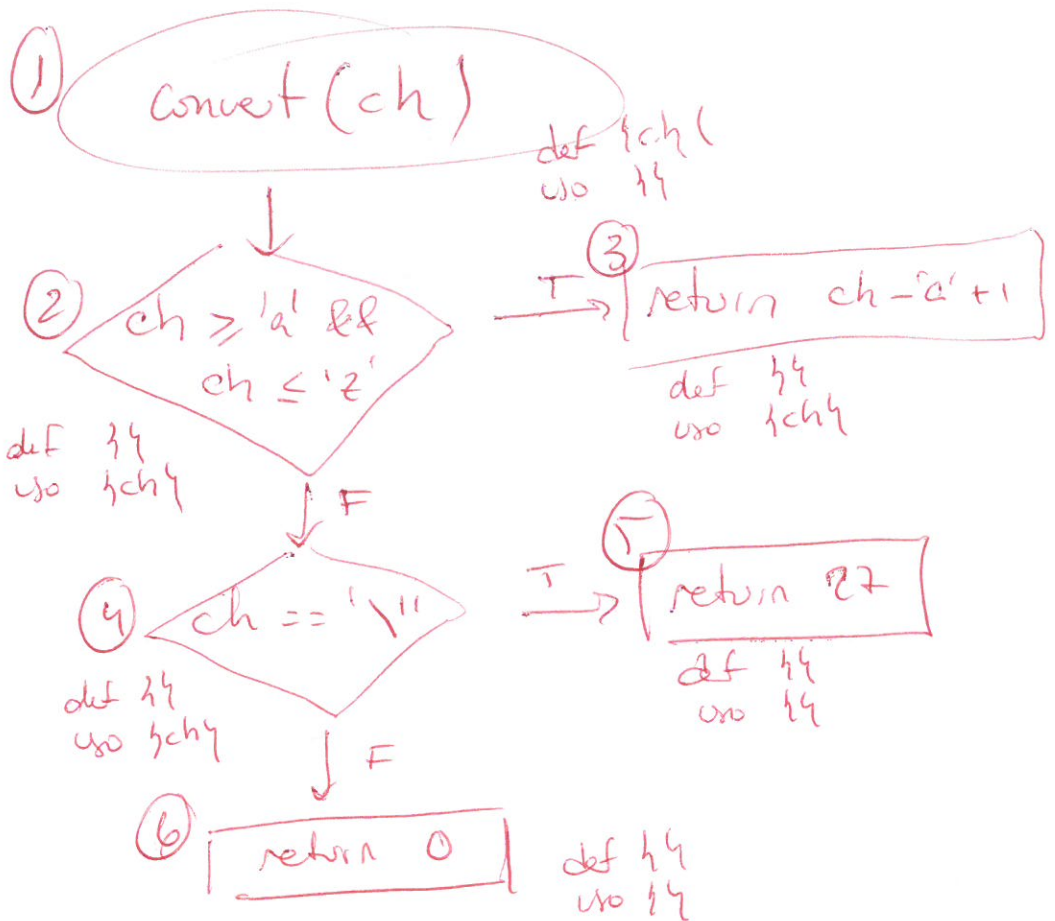
necesitamos una
cadena en longitud
menor que 5 }
otra en longitud
mayor que 5

2 casos de prueba:

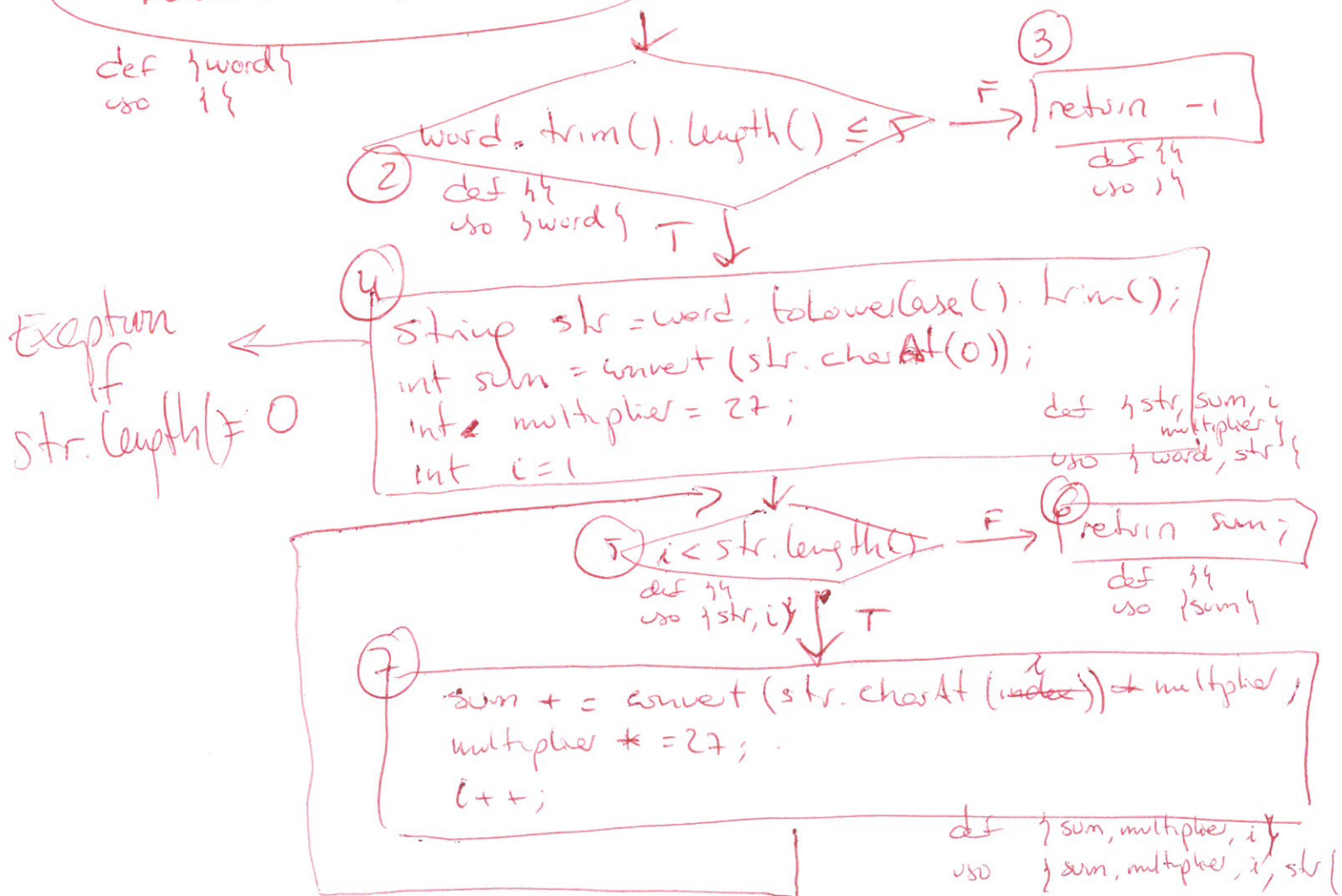
"Cadena" \longrightarrow salida -1

"a!á" \longrightarrow salida $0 + 27 \times 27 + 0$

2b)



① hashIndex(word)



2.6/ Cont 1

Pares DU / caminos DU

<u>convert</u>	ch	(1, 2)	[1, 2, 3] [1, 2, 4]
		(1, 3)	[1, 2, 3]
		(1, 4)	[1, 2, 4, 5] [1, 2, 4, 6]
<u>hashIndex</u>	word	(1, 2)	[1, 2, 3] [1, 2, 4]
		(1, 4)	[1, 2, 4]
	str	(4, 4)	[4]
		(4, 5)	[4, 5, 6] [4, 5, 7]
		(4, 7)	[4, 5, 7]
	sum	(4, 6)	[4, 5, 6]
		(4, 7)	[4, 5, 7]
		(7, 7)	[7, 5, 7]
		(7, 6)	[7, 5, 6]
	i	(4, 5)	[4, 5, 6] [4, 5, 7]
		(4, 7)	[4, 5, 7]
		(7, 5)	[7, 5, 6] [7, 5, 7]
		(7, 7)	[7, 5, 7]
	multiples	(4, 7)	[4, 5, 7]
		(7, 7)	[7, 5, 7]

2.b) Cont 2

Se pide 100% de casos DU

En Contet lo conseguimos llegando a nodos ③ y ④

En hashIndex necesitamos:

- No es necesario pasar por ③
- el ~~caso~~ par (4,6) obliga a considerar una cadena de longitud 1
- (7,7) obliga a una cadena de long 3

2 casos de prueba

"a" → salida 0

"3a3" → salida 27