



Ejercicios para resolver en papel

1.1 Expresiones condicionales

Condicional simple

 **Ejercicio 1.** ¿Todas las condiciones dadas por estructuras if son posibles?

Analice en cada ítem la expresión condicional propuesta y complete la tabla asociada según corresponda. Para cada resultado en la tabla, debe completar qué valores deberían asumir las condiciones (<cond A>, <cond B>, etc.) para que la expresión reduzca al mismo. En el caso en que, independientemente de los valores de las condiciones, la expresión nunca pueda reducir al resultado debe marcar **X** en la columna 'Nunca'. Tenga en cuenta que en algunos casos el valor de una condición ha sido fijado y no puede ser modificado.

a) (if (and <cond A> <cond B>
 "resultado 1"
 (if (not <cond B>
 "resultado 2"
 "resultado 3"
)
)

		¿ #true, #false o indistinto?	
Reduce a	Nunca	<cond A>	<cond B>
"resultado 1"			
"resultado 2"			
"resultado 3"			

b) (if (and <cond A> <cond B>
 (if (not <cond B>
 "resultado 1"
 "resultado 2"
)

		¿ #true, #false o indistinto?	
Reduce a	Nunca	<cond A>	<cond B>
"resultado 1"			
"resultado 2"			
"resultado 3"			#true

```

c) (if (not (or <cond A> <cond B>))
      (if (and <cond B> <cond C>)
          "resultado 1"
          (if (not <cond C>)
              "resultado 2"
              (if <cond A>
                  "resultado 3"
                  "resultado 4"
              )
          )
      )
    )
    "resultado 5"
  )

```

		¿ #true, #false o indistinto?		
Reduce a	Nunca	<cond A>	<cond B>	<cond C>
"resultado 1"				
"resultado 2"				
"resultado 3"				
"resultado 4"				
"resultado 5"				#true

Ejercicio 2. ¿Son necesarias tantas estructuras if ?

Observe las siguientes expresiones condicionales. En cada caso debe indicar si es posible reescribir la expresión reduciendo la cantidad de estructuras `if`. En caso afirmativo, reescribala con la cantidad mínima de `if` necesarios.

Nota: en algunos casos es posible reescribir la expresión sin ningún `if`. Analice qué tienen de particular esos casos.

```

a) (if (or (= (modulo x 5) 0) (> x 15))
      #t
      #f )

```

```

b) (if (even? x)
      (if (> x 3) #t #f )
      #f )

```

```



c) (if (even? x)
      (if (> x 3)
          #t
          (if (< x 0) #t #f)
      )
      #f)

```

```
d) (if (< x 15)
      (if (odd? x)
          (* x 3)
          x)
      (if (even? x)
          x
          (* x 3)))
)
```

```
e) (if (> x 100)
      (if (= (modulo x 3) 0)
          x
          (if (= (modulo x 7) 0)
              x
              (+ x 1) ))
      (if (even? x)
          (+ x 1)
          (- x 1) ))
)
```

Condicional múltiple

 **Ejercicio 3.** Reescriba la expresión del  **Ejercicio. 2.e)** utilizando condición múltiple.

Ejercicio 4. ¿Importa el orden?

Las dos siguientes definiciones, solo se diferencian en el orden en el que se presentan las condiciones. Considere cada resultado y analice cómo cada función reduce al mismo. En la tabla indique si los valores de x para alcanzar dicho resultado coinciden en ambas funciones. En tal caso, especifique en la columna correspondiente cuáles son esos valores. En el caso en que no coincidan, indique para cada función cuáles son los valores de x que permiten obtener el resultado analizado.

```
;c1: Number -> String
(define (c1 x)
  (cond [(>= x 0) "resultado 1"]
        [(even? x) "resultado 2"]
        [else "resultado 3"])))
```

```
;c2: Number -> String
(define (c2 x)
  (cond [(even? x) "resultado 2"]
        [(>= x 0) "resultado 1"]
        [else "resultado 3"])))
```

Para obtener el resultado	c1 y c2 coinciden x es	c1 y c2 NO coinciden	
		x en c1 es	x en c2 es
"resultado 1"			
"resultado 2"			
"resultado 3"			

Ejercicio 5. Tres versiones para el Año Bisiesto

Año bisiesto es aquel que es divisible por 4, salvo que sea año secular (divisible por 100), en cuyo caso también debe ser divisible por 400.

Se requiere definir una función `bisiesto?` que reciba un año y determine si es o no bisiesto. Defina tres versiones de la función según se indique:

- a) La definición de la función sólo debe utilizar la sentencia `if`.
- b) La definición de la función sólo debe utilizar la sentencia `cond`.
- c) La definición de la función no debe utilizar ni `if`, ni `cond`.

Para resolver puede utilizar la función `divisible?` definida más abajo.

```
; divisible? : Number Number -> Boolean
; Si x es divisible por n, (divisible? x n) reducirá a #t; en caso contrario,
; reducirá a #f.
(define (divisible? x n)
  (= (modulo x n) 0))
```