--CAPITULO 8----------Octavio Reyes Pinedo------------------------------

-------------------------------------------------------------------------------

type Set a = [a]

intersection :: Eq a => Set a -> Set a -> Set a

intersection xs = foldr intersect []

where intersect new acc = if new `elem` xs then new:acc else acc

difference :: Eq a => Set a -> Set a -> Set a

difference xs ys = foldr diff [] xs

where diff new acc = if new `elem` ys then acc else new:acc

-- Subset

subset :: (Eq a)=> [a] -> [a] -> Bool

subset xs ys = and [elem x ys | x <- xs]

setEq :: (Eq a) => [a] -> [a] -> Bool

setEq xs ys = subset xs ys && subset ys xs

-

---------------------------------------------------------

--Exercise 1

--A={1,2,3,4,5}

--B={2,4,6}

--(a) A ? (B n A)

-- {1,2,3,4,5} ? {2,4}

-- {1,2,3,4,5}

--(b) (A n B) ? B

-- {2,4} ? {2,4,6}

-- {2,4,6}

--(c) A - B

-- {1,2,3,4,5} - {2,4,6}

-- {1,3,5}

--(d) (B - A) n B

-- {2,4,6}-{1,2,3,4,5}

-- {6}

--(e) A ? (B - A)

-- {1,2,3,4,5} ? ({2,4,6}-{1,2,3,4,5})

-- {1,2,3,4,5} ? {6}

-- {1,2,3,4,5,6}

-----------Exercise 2----------------------------

--(a)

ex2a=[1,2,3] `union` [3]

--[1,2,3]

--(b)

ex2b=[4,2] `union` [2,4]

--[4,2]

--(c)

ex2c=[1,2,3] `intersection` [3]

--[3]

--(d)

ex2d = [] `intersection` [1,3,5]

--[]

--(e) Difference

ex2e = [1,2,3] `difference` [3]

--[1,2]

--(f)

ex2f=[2,3] `difference` [1,2,3]

--[]

--(g)

ex2g=[1,2,3] `intersection` [1,2]

--[1,2]

--(h)

ex2h=[1,2,3] `union` [4,5,6]

--[1,2,3,4,5,6]

--(i)

ex2i=([4,3] `difference` [5,4]) `intersection` [1,2]

--[]

--(j)

ex2j=([3,2,4] `union` [4,2]) `difference` [2,3]

--[4]

--(k)

ex2k=subset [3,4] [4,5,6]

--False

--(l)

ex2l=subset [1,3] [4,1,3,6]

--True

--(m)

ex2m=subset [] [1,2,3]

--True

--(n)

ex2n=setEq [1,2] [2,1]

--True

--(o)

ex2o= setEq [3,4,6] [2,3,5]

--False

--(p)

ex2p= [1,2,3] `difference` [1]

-- [2,3]

--(q)

ex2q = [] `difference` [1,2]

-- []

--------------------------------------------------------------

--Exercise 3. The function

--powerset :: (Eq a, Show a) => Set a -> Set (Set a)

--takes a set and returns its power set. Work out the values of the following

--expressions:

ex3a=powerset [3,2,4]

-- Result = [[],[4],[2],[2,4],[3],[3,4],[3,2],[3,2,4]]

ex3b=powerset [2]

-- Result = [[],[2]]

powerset :: Eq a => Set a -> [Set a]

powerset [] = [[]]

powerset (x:xs) = xss ++ map (x:) xss

where xss = powerset xs

--------------Exercise 4----------------------------------------

--Exercise 4. The cross product of two sets A and B is defined as

--A × B = {(a, b) | a *∈* A, b *∈* B}

--The function

--crossproduct :: (Eq a, Show a, Eq b, Show b) =>

--Set a -> Set b -> Set (a,b)

--takes two sets and returns their cross product. Evaluate these expressions:

--crossproduct [1,2,3] [’a’,’b’]

--crossproduct [1] [’a’,’b’]

crossProduct :: Set a -> Set b -> Set (a,b)

crossProduct xs ys= [(x,y) | x <- xs, y <- ys]

ex4a = crossProduct [1,2,3] ['a','b']

--[(1,'a'),(1,'b'),(2,'a'),(2,'b'),(3,'a'),(3,'b')]

ex4b = crossProduct [1] ['a','b']

--[(1,'a'),(1,'b')]

--------------Exercise 5-------------------------------------------

--Exercise 5. In the following exercise, let u be [1,2,3,4,5,6,7,8,9,10], a

--be [2,3,4], b be [5,6,7] and c be [1,2]. Give the elements of each set:

--a) a +++ b

--b) u˜˜˜a \*\*\* (b +++ c)

--c) c ˜˜˜ b

--d) (a +++ b) +++ c

--e) u˜˜˜a

--f) u˜˜˜(b \*\*\* c)

ex5a=[2,3,4] `union` [5,6,7]

--[2,3,4,5,6,7]

ex5b= [1,2,3,4,5,6,7,8,9,10] `difference`[2,3,4] `intersection` ([5,6,7] `union` [1,2])

--[1,5,6,7,8,9,10] `intersection` [1,2,5,6,7]

--[1,5,6,7]

ex5c=[1,2] `difference` [5,6,7]

-- [1,2]

ex5d=([2,3,4] `union` [5,6,7]) `union` [1,2]

-- [2,3,4,5,6,7] `union` [1,2]

-- [1,2,3,4,5,6,7]

ex5e=[1,2,3,4,5,6,7,8,9,10]`difference`[2,3,4]

-- [1,5,6,7,8,9,10]

ex5f=[1,2,3,4,5,6,7,8,9,10]`difference`([5,6,7]`intersection`[1,2])

-- [1,2,3,4,5,6,7,8,9,10]`difference`[]

-- [1,2,3,4,5,6,7,8,9,10]

-----------Exercise 6-----------------------------------------------

--Exercise 6. What are the elements of the set {x+y | x *∈* {1, 2, 3} *∧* y *∈* {4, 5}}?

-- [( 1 + 4), ( 1 + 5), ( 2 + 4), ( 2 + 5), ( 3 + 4), ( 3 + 5)]

-- = [5,6,6,7,7,8]

ex6=[x+y | x <- [1,2,3], y<-[4,5]]

----------Exercise 7 -----------------------------------------------

--Exercise 7. Write and evaluate a list comprehension that expresses the set

--{x |x *∈* {1, 2, 3, 4, 5} *∧* x < 0}

ex7 = [x | x <- [1..5], x < 0]

-- []

---------Exercise 8---------------------------------------------------

--Exercise 8. Write and evaluate a list comprehension that expresses the set

--{x + y | x ∈ {1, 2, 3} ∧ y ∈ {4, 5}}

ex8 = [x + y | x <- [1,2,3], y <- [4,5]]

--[5,6,6,7,7,8]

--------Exercise 9 --------------------------------------------------

--Exercise 9. Write and evaluate a list comprehension that expresses the set

--{x |x ∈ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} ∧ even x}

ex9 = [x | x <- [1,2,3,4,5,6,7,8,9,10], even x]

-- [2,4,6,8,10]

--------Exercise 10--------------------------------------------------

--Exercise 10. What is the value of each of the following expressions?

--a) subset [1,3,4] [4,3]

--b) subset [] [2,3,4]

--c) setEq [2,3] [4,5,6]

--d) setEq [1,2] [1,2,3]

ex10a=subset [1,3,4] [4,3]

-- False

ex10b=subset [] [2,3,4]

-- True

ex10c=setEq [2,3] [4,5,6]

--False

ex10d=setEq [1,2] [1,2,3]

--False

-----------Exercise 11---------------------------------------------------

--Let *A*, *B*, and *C* be sets. Prove that if *A ⊂ B* and *B ⊂ C*, then *A ⊂ C*.

1. *B ⊂ C {* Premise *}*

2. *x ∈ C → x*  *B {* Def. *⊂ }*

3. *A ⊂ B {* Premise *}*

4. *x*  *A → x ∈ B {* Def. *⊂ }*

5. *x*  *A → x ∈ C {* Hypothetical syllogism (chain rule), (2), (4) *}*

6. *∀x.* (*x*  *A → x ∈ C*) *{ ∀* introduction *}*

7. *A ⊂ C {* Def. *⊂ }*

-- Sea x un elemento en C que no está en B

-- x no puede estar en A y cada elemento de A esta en B

-- x está en C pero no está en A, por lo tanto A es un subconjunto de C

---------Exercise 12-------------------------------------------------

--Exercise 12. Consider the following two claims. For each one, if it is true

--give a proof, but if it is false give a counterexample.

**-- (a)** If *A ⊆ B* and *B ⊆ C*, then *A ⊂ C*.

**-- (b)** If *A ⊂ B* and *B ⊂ C*, then *A ⊆ C*.

------------12a es FALSO-------------------

--**(a)** If *A ⊆ B* and *B ⊆ C*, then *A ⊂ C*.

--Proof. Sea x un elemento

--1. A *⊆* B { Premise }

--2. *x ∈ A → x ∈ B* { Def. *⊆* }

--3. B *⊆*  C { Premise }

--4. *x ∈ B → x ∈ C* { Def. *⊆* }

--5. *x ∈ A → x ∈ C* { Hipotesis (2) (4) }

--6. *∀x.* (*x ∈ A → x ∈ C*) { *∀* introduction }

--7. A *⊆*  C { Def. *⊆* }

--Es falso en caso de que A,B y C tengan los mismos elementos.

--Ej. Si A=B y B=C entonces no puede ser que A *⊂*  C

--A=[1,2,3]

--B=[1,2,3]

--c=[1,2,3]

--A *⊆* B y B *⊆* C

--Entonces es Falso que A*⊂* C => [1,2,3] *⊂* [1,2,3]

----------12b es VERDADERO----------------

--**(b)** If *A ⊂ B* and *B ⊂ C*, then *A ⊆ C*.

--Proof. Sea x un elemento de A (x *∈* A)

--1. A *⊂* B { Premise }

--2. x *∈* B debido a que A *⊂* B { Def.  *⊂* }

--3. B *⊂* C { Premise }

--4. x *∈* C debido a que B *⊂* C { Def. *⊂* }

--5. *x ∈ A → x ∈ C* { Silogismo hipotetico (regla de cadena), (2), (4) }

--6. *∀x.* (*x ∈ A → x ∈ C*) { *∀* introduction }

--7. A *⊆* C { Def. *⊆* }

--Ej.

-- A=[1,2,3]

-- B=[1,2,3,4]

-- C=[1,2,3,4,5]

-- A *⊂* B [1,2,3] *⊂* [1,2,3,4]

-- B *⊂* C [1,2,3,4] *⊂*  [1,2,3,4,5]

-- A *⊆*  C [1,2,3] *⊆* [1,2,3,4,5]

-----------Exercise 13-----------------------

--Exercise 13. For the following questions, give a proof using set laws, or find

--a counterexample.

**--(a)** (*A’ ∪ B*) *’ ∩ C’* = *A ∩* (*B ∪ C*) *’*

**-- (b)** *A −* (*B ∪ C*) *’* = *A ∩* (*B ∪ C*)

**-- (c)** (*A ∩ B*) *∪* (*A ∩ B\_*) = *A*

**-- (d)** *A ∪* (*B − A*) = *A ∪ B*

**-- (e)** *A − B* = *B’ − A’*

**-- (f)** *A ∩* (*B − C*) = (*A ∩ B*) *−* (*A ∩ C*)

**-- (g)** *A −* (*B ∪ C*) = (*A − B*) *∩* (*A − C*)

**-- (h)** *A ∩* (*A’ ∪ B*) = *A ∩ B*

**-- (i)** (*A − B’*) *∪* (*A − C’*) = *A ∩* (*B ∩ C*)

--a) (*A’ ∪ B*) *’ ∩ C’* = *A ∩* (*B ∪ C*) *’*

--(A'' *∩* B') *∩* C' = *A ∩* (*B ∪ C*) *’* {Demorgan's laws}

--(A *∩* B') *∩* C' = *A ∩* (*B ∪ C*) *’* {Double Complement}

--A *∩* (B' *∩* C') = *A ∩* (*B ∪ C*) *’* {Associativity of n}

--A *∩* (B *∪* C)' = *A ∩* (*B ∪ C*) *’* {Demorgan}

--**b)** *A −* (*B ∪ C*) *’* = *A ∩* (*B ∪ C*)

-- A *∩* (B *∪*  C)'' = *A ∩* (*B ∪ C*) {Teorema 69 . 5}

-- A *∩* (B *∪* C) = *A ∩* (*B ∪ C*) {Double complement}

--**(c)** (*A ∩ B*) *∪* (*A ∩ B’*) = *A*

-- A *∩* (B *∪*  B') = A {Teorema 70}

-- A *∩* U = A {Universe}

-- A =A {Identity}

-- **d)** A ∪ (B − A) = A ∪ B

-- A ∪ (B n A') = A ∪ B {Teorema 69}

-- (A ∪ B) n (A ∪ A') = A ∪ B {Teorema 71}

-- (A ∪ B) n U= A ∪ B {Universe}

-- A ∪ B = A ∪ B {Identity.2}

--e) A - B = B' - A'

-- A n B'= B' - A' {Teorema 69 .1}

-- B' n A = B' - A' {Teorema 69 .2}

-- B' n A''= B' - A' { Double complement}

-- B’ – A’= B' - A' {Teorema 69 .5}

-- **f)** *A ∩* (*B − C*) = (*A ∩ B*) *−* (*A ∩ C*)

-- A n (B - C) = (A n B) - (A n C)

-- A n (B - C) = (A n B) n (A n C)' {Teorema 69 .5}

-- A n (B - C) = (A n B) n (A' U C') {Teorema 72 .2}

-- A n (B - C) = ((A n B) n A') U ((A n B) n C') {Def. Set equality}

-- A n (B - C) = (A n B n A') U (A n B n C') {Associative}

-- A n (B - C) = Ø U (A n B n C') {Def. Ø }

-- A n (B - C) = A n B n C' {Identity}

-- A n (B - C) = A n (B - C) {Teorema 69 .5}

--g) A - (B U C) = (A - B) n (A - C)

-- A n (B U C)' = (A - B) n (A - C) {Teorema 69 .5}

-- A n (B' n C') = (A - B) n (A - C) {Teorema 72 .1}

-- A n A n B' n C'= (A - B) n (A - C) {Associative}

-- (A n B') n (A n C') = (A - B) n (A - C) { Associative}

-- (A - B) n (A - C) = (A - B) n (A - C) {Teorema 69 .5}

--h) A n (A' U B) = A n B

-- (A n A') U (A n B) = A n B {Distributive Laws}

-- Ø U (A n B) = A n B {Def. Ø }

-- A n B= A n B {Identity}

--i) (A - B') U (A - C') = A n (B n C)

-- (A n B'') U (A n C'') = A n (B n C) { Teorema 69 .5}

-- (A n B) U (A n C) = A n (B n C) {Double complement}

-- A n (B U C) = A n (B U C) {Distributive Laws}

---------------------------Exercise 14------------------------------------------------

--Exercise 14. The function

--smaller :: Ord a => a -> [a] -> Bool

--takes a value and a list of values and returns True if the value is smaller

--than the first element in the list. Using this function, write a function

--that takes a set and returns its powerset. Use foldr.

smaller :: Ord a => a -> [a] -> Bool

smaller x [] = True

smaller x (y:xs) = x < y

-- using this, write a function that takes a set and returns its powerset.

powerset2 :: (Eq a, Ord a) => [a] -> [[a]]

powerset2 = foldr cjoin [[]]

where cjoin new acc = [new:x | x <- acc, new `smaller` x, not (new `elem` x)] ++ acc

---------------------------Exercise 15------------------------------------------------

--Exercise 15. Prove that (*A ∪ B*)*’* = ((*A ∪ A’*) *∩ A’*) *∩* ((*B ∪ B’*) *∩ B’*).

--(A U B)' = A' n B'

--(A U B)' = (U - A) n (U - B)

--(A U B)' = ((A U A') - A) n ((B U B') - B)

--(A U B)' = ((A U A') n A') n ((B U B') n B')

---------------------------Exercise 16------------------------------------------------

--Exercise 16. Using a list comprehension, write a function that takes two sets

--and returns True if the first is a subset of the other.

isSubset :: Eq a => [a] -> [a] -> Bool

isSubset set1 set2 = null [e | e <- set1, not (elem e set2)]

---------------------------Exercise 17------------------------------------------------

---Exercise 17. What is wrong with this definition of diff, a function that takes

--two sets and returns their difference?

--diff :: Eq a => [a] -> [a] -> [a]

--diff set1 set2 = [e | e <- set2, not (elem e set1)]

----Los argumentos están en el orden incorrecto, debe ser e <-set1, not (elem e set2)

---------------------------Exercise 18------------------------------------------------

--Exercise 18. What is wrong with this definition of intersection, a function

--that takes two sets and returns their intersection?

--intersection :: [a] -> [a] -> [a]

--intersection set1 set2 = [e | e <- set1, e <- set2]

-- Al ejecutar la intersection con

-- intersection [3,4,5] [6,7,8]

-- el resultado es [6,7,8,6,7,8,6,7,8]

-- Se agrega a "e" el conjunto set1 pero también se le agrega a "e" el conjunto set2

-- en lugar de buscar los elementos e<-set1 en el conjunto de set2

-- Por lo que se sobreescriben los valores 6,7 y 8.

-- La definición correcta es:

-- intersection18 :: [a] -> [a] -> [a]

-- intersection18 set1 set2 = [e | e <- set1, e ‘elem‘ set2]

---------------------------Exercise 19------------------------------------------------

--Exercise 19. Write a function using a list comprehension that takes two sets

--and returns their union.

union :: Eq a => [a] -> [a] -> [a]

union set1 set2 = set1 ++ [e | e <- set2, not (elem e set1)]

---------------------------Exercise 20------------------------------------------------

--Exercise 20. Is it ever the case that *A ∪* (*B − C*) = *B*?

-- Solo cuando A *⊆* B and C *⊆*  A

-- A=[4,5,6]

-- B=[4,5,6,7]

-- C=[4,5]

-- [4,5,6] U ([4,5,6,7]-[4,5]) = [4,5,6,7]

-- [4,5,6] U [6,7] = [4,5,6,7]

-- [4,5,6,7] = [4,5,6,7]

---------------------------Exercise 21------------------------------------------------

--Exercise 21. Give an example in which (A U C) n (B U C) = Ø.

--Ambas uniones contienen C, entonces C debe ser vacío, además A y B deben ser diferentes

--Ej. A = {1, 2, 3}, B = {4, 5, 6}, C = Ø.

-- ({1, 2, 3} U Ø) n ({4, 5, 6} U Ø) = Ø

-- {1, 2, 3} n {4, 5, 6} = Ø

-- Ø = Ø

---------------------------Exercise 22------------------------------------------------

--Exercise 22. Prove the commutative law of set-intersection, A ∩ B = B ∩ A.

--A ∩ B= {x|x ∈ A ∧ x ∈ B} { defn ∩ }

--A ∩ B = {x|x ∈ B ∧ x ∈ A} { ∧ commutative }

--A ∩ B = B ∩ A { defn ∩ }

---------------------------Exercise 23------------------------------------------------

--Exercise 23. Express the commutative law of set-intersection in terms of the

--set operations and Boolean operations defined in the Stdm module.

-- Propiedad conmutativa

-- A intersection B = B intersection A

ex23a=[1,2,3] `intersection` [1,2,4]

--[1,2]

ex23b=[1,2,4] `intersection` [1,2,3]

--[1,2]

--- El resultado es [1,2] en ambas intersecciones

---------------------------Exercise 24------------------------------------------------

--Exercise 24. Prove the associative law of set-union, (A∪B)∪C = A∪(B∪C).

--(A ∪ B) ∪ C= {x |(x ∈ A ∨ x ∈ B) ∨ x ∈ C} { defn ∪ }

--(A ∪ B) ∪ C= {x|x ∈ A ∨ (x ∈ B ∨ x ∈ C)} { ∨ associative }

--(A ∪ B) ∪ C= A ∪ (B ∪ C) { def ∪ }

---------------------------Exercise 25------------------------------------------------

--Exercise 25. Prove that the difference between two sets is the intersection of

--one with the complement of the other, which can be written as

--A − B = A ∩ B'.

--A − B'' = A ∩ B' {double difference}

--A − B''= {x|x ∈ A ∧ x ∈ B''} { def. − }

--A − B''={x|x ∈ A∧ ¬(x ∈ U ∧ x ∈ B'')} { def. complement }

--A − B''={x|x ∈ A∧ ¬(x ∈ U ∧ ¬(x ∈ B'))} { def. ∈ }

--A − B''={x|x ∈ A∧ ¬(x ∈ U ∧ ¬(x ∈ U ∧ x ∈ B))} { def. complement }

--A − B''={x|x ∈ A∧ ¬(x ∈ U ∧ ¬(x ∈ U ∧ ¬(x ∈ B)))} { def. ∈ }

--A − B''={x|x ∈ A∧ ¬(x ∈ U ∧ (¬(x ∈ U) ∨ (x ∈ B)))} { DM, double negation }

--A − B''={x|x ∈ A ∧ (¬(x ∈ U)∨ ¬(¬(x ∈ U) ∨ (x ∈ B)))} { DM }

--A − B''={x|x ∈ A ∧ (x ∈ U ∨ (¬¬(x ∈ U)∧ ¬(x ∈ B)))} { DM }

--A − B''={x|x ∈ A ∧ (x ∈ U ∨ (x ∈ U ∧ x ∈ B))} { double negation, def. ∈ }

--A − B''={x|x ∈ A ∧ (x ∈ U ∧ x ∈ B)} { ∨ null, def. ∈ }

--A − B''={x|x ∈ A ∧ x ∈ U − B} { def.− }

--A − B''=A ∩ B' { def. complement,∩ }

---------------------------Exercise 26------------------------------------------------

--Exercise 26. Prove that union distributes over intersection,

--A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C).

--A ∪ (B ∩ C) = {x|x ∈ A ∨ (x ∈ B ∧ x ∈ C)} { def. ∩,∪ }

--A ∪ (B ∩ C) = {x|(x ∈ A ∨ x ∈ B) ∧ (x ∈ A ∨ x ∈ C)} { ∨ over ∧ }

--A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C) { def . ∩, ∪ }

---------------------------Exercise 27------------------------------------------------

--Exercise 27. Prove DeMorgan’s law for set intersection, (A ∩ B)' = A' ∪ B'

--(A ∩ B)'= {x|x ∈ U ∧ x ∈ (A ∩ B)} { def. complement }

--(A ∩ B)'= {x|x ∈ U ∧ ¬(x ∈ A ∧ x ∈ B)} { def. ∈,∩ }

--(A ∩ B)'= {x|x ∈ U ∧ (¬(x ∈ A)∨ ¬(x ∈ B))} { DM }

--(A ∩ B)'= {x|x ∈ U ∧ (x ∈ A ∨ x ∈ B)} { def. ∈ }

--(A ∩ B)'= {x|(x ∈ U ∧ x ∈ A) ∨ (x ∈ U ∧ x ∈ B)} { ∧ over ∨ }

--(A ∩ B)'= {x|(x ∈ U − A) ∨ (x ∈ U − B)} { def. − }

--(A ∩ B)'= A' ∪ B' { def. complement,∪ }