Programare Declarativa –Lab 11

Se dă tipul de date:
```
data Dom a =  Empty                 -- interval vid (multimea vida)
            | Full                  -- intervalul total (multimea totala)
            | Ran a a               -- interval inchis            [a,b]
            | (Dom a) :|: (Dom a)   -- reuniunea a 2 intervale    A U B
            | (Dom a) :&: (Dom a)   -- intersectia a 2 intervale  A ∩ B
           deriving Show
```

reprezentând un domeniu format din reuniuni și intersecții de intervale închise.

Exemplu:
```
exem:: Dom Int                              -- ([1,3] U [2,4]) ∩ ([3,5] ∩ {})
exem = (((Ran 1 3) :|: (Ran 2 4)) :&: ((Ran 3 5) :&: Empty))
```

Atentie:

- Pentru doua valori `x` si `y` de tip a vom presupune ca `Ran x y` si `Ran y x` reprezinta aceeasi valoare de tip `Dom a`.
- Anumite funcții necesită constrângeri de tipuri, domeniile fiind definite pe tipuri ordonate de date. Folositi constrangerea `Ord a` atunci cand este necesară. In acest caz se va presupune ca o valoare `Ran x y` este interpretat ca intervalul `[min{x,y}, max{x,y}]`.

1) Faceti `Dom a` instanta a clasei `Eq` tinand cont de precizarile de mai sus.

2) Scrieți funcția `exist` care primește o valoare de tip `a`, un domeniu de tip `a` și returnează o valoare booleană reprezentând apartenența valorii la domeniul respectiv

```
exist 3 ((Ran 1 3) :|: (Ran 6 10)) == True      -- 3 ∈ [1,3] U [6,10]
exist 5 ((Ran 1 3) :|: (Ran 6 10)) == False     -- 5 ∉ [1,3] U [6,10]
```

3) Scrieți funcția `overlap` care primește ca argumente două domenii de tip `a` și returnează `True` dacă cele intervalele se intersectează și `False` in caz contrar. Se va considera doar cazul de intersecție a două domenii de tip `Ran a a`, pentru celelalte cazuri returnați `False`.

```
overlap (Ran 1 4) (Ran 3 5) == True             -- [1,4] ∩ [3,5] = [3,4]
overlap (Ran 1 4) (Ran 5 6) == False            -- [1,4] ∩ [5,6] = {}
overlap Empty     (Ran 3 4) == False            -- Empty ∩ [3,4] = Empty
overlap Full      (Ran 3 4)  == True            -- Full ∩ [3,4] = [3,4]
```

4) Scrieți funcția `normalize` care primeste ca argument un domeniu de tip `a` și îl normalizează prin aplicarea proprietății de distributivitate: `(AUB) ∩ C = (A∩C) U (B∩C)`

```
normalize (((Ran 1 2) :|: (Ran 3 4)) :&: (Ran 2 3))
      == (Ran 1 2 :&: Ran 2 3) :|: (Ran 3 4 :&: Ran 2 3)
```

5) Faceți `Dom a` instanță a clasei monoid în doua moduri:

`(Dom a, :|:, Empty)` si `(Dom a, :&:, Full)`

Pentru fiecare modalitate definiti o un nou tip de date folosind `newtype`:

```
newtype SDom a = ....    -- pentru (Dom a, :|:, Empty)
newtype PDom a = ...     -- pentru (Dom a, :&:, Full)
```

6) Scrieți funcția `optimize` care primeste ca argument un domeniu normalizat de tip `a` și îl optimizează prin calculul intersecțiilor și reuniunilor, inclusiv eliminearea valorilor `Empty` și `Full` care sunt interpretate ca elemente neutre pentru `:|:` și `:&:`.

```
input: (([1,5] :&: [0,7]):|:([3,9] :&: [0,7]) ):|:( ([8,11] :&: [9,14]) :|: {})
output: [1,7] :&: [9,11]
```

7) Definim
```
data DomF a =  Empty                  -- interval vid (multimea vida)
             | Ran a a                -- interval inchis            [a,b]
             | (DomF a) :|: (DomF a)  -- reuniunea a 2 intervale    A U B
             deriving Show
```
Faceți `DomF` instanță a clasei `Foldable`.

8) Definiți tipul de date `Bin a` care are ca valori arborii binari cu frunze de tipul `a`.

(a) Faceți `Bin a` instanta a clasei `Foldable` definind funcția `fmap`.

(b) Folosind numai funcția fmap de la punctul (a) scrieți o functie `sum:: Bin SDom a -> SDom a` și o funcție `prod: Bin PDom a -> PDom a` care sa intoarca reuniunea, respectiv intersectia frunzelor arborelui (`SDom` și `PDom` sunt definiti la punctul 5).