

# Practice6.pdf



**TuxEnjoyer**



**Lenguajes, Tecnologías y Paradigmas de la Programación**



**2º Grado en Ingeniería Informática**



**Escuela Técnica Superior de Ingeniería Informática  
Universidad Politécnica de Valencia**



[Accede al documento original](#)



Escuela de  
Organización  
Industrial

Contigo que evolucionas.  
Contigo que lideras. Contigo que transformas.

**Esto es EOI.  
Mismo propósito,  
nueva energía.**



Descubre más aquí



**EOI** Escuela de  
Organización  
Industrial



## Practice 6

```
--Ej 3
instance (Show a) => Show (Queue a) where
  show (Queue [] []) = " <- "
  show (Queue [] (y:ys)) = (show (Queue [] ys)) ++ (show y) ++ " <- "
  show (Queue (x:xs) (ys)) = " <- " ++ (show x) ++ (show (Queue xs ys))
```

```
--Ej 4
-- instance Show ...
instance (Show a) => Show (Queue a) where
  show EmptyQueue = " <- "
  show (Item x y) = " <- " ++ (show x) ++ (show y)

instance (Eq a) => Eq (Queue a) where
  EmptyQueue == EmptyQueue = True
  EmptyQueue == Item _ _ = False
  Item _ _ == EmptyQueue = False
  Item a b == Item a2 b2 = (a == a2) && (b == b2)
```

```
--Ej 5, put this in the testEj5P6.hs
toList :: Queue a -> [a]
toList a = if (isEmpty a) then []
            else (first a) : (toList (dequeue a))

fromList :: [a] -> Queue a
fromList = foldl (\q x -> enqueue x q) empty
```

```
--Ej 6
type Side    = Float
type Apothem = Float
type Radius  = Float

data Shape = Pentagon Side Apothem |
            Circle Radius
            deriving (Eq, Show)

perimeter :: Shape -> Float
perimeter (Pentagon s a) = 5 * s
perimeter (Circle r)     = 2 * pi * r

area :: Shape -> Float
```



```

area (Pentagon s a) = (perimeter (Pentagon s a)) * a / 2
area (Circle r)     = pi * r * r

```

```

--Ej 7,8,9
type Side      = Float
type Apothem   = Float
type Radius    = Float

-- volumePrism
type Height = Float
type Volume = Float
volumePrism :: (Shape a) => a -> Height -> Volume
volumePrism base height = (area base) * height

data Pentagon = Pentagon Side Apothem
data Circle  = Circle Radius

class (Eq a, Show a) => Shape a where
    perimeter :: a -> Float
    area :: a -> Float

instance Shape Pentagon where
    perimeter (Pentagon s a) = 5 * s
    area (Pentagon s a)     = (perimeter (Pentagon s a)) * a / 2

instance Shape Circle where
    perimeter (Circle r) = 2 * pi * r
    area (Circle r)     = pi * r * r

--surfacePrism
type Surface = Float
surfacePrism :: (Shape a) => a -> Height -> Surface
surfacePrism base height = 2 * area base + (perimeter base) * height

instance Show Pentagon where
    show (Pentagon s a) = "Pentagon, side: " ++ show s ++ ", apothem: " ++ show a

instance Show Circle where
    show (Circle r) = "Circle, radius: " ++ show r

instance Eq Pentagon where
    (==) (Pentagon s a) (Pentagon s1 a1) = s == s1 && a == a1

instance Eq Circle where
    (==) (Circle r) (Circle r1) = r == r1

```