

Especificacion de TADs

Ejercicio 1 Especificar en forma completa el TAD `NumeroRacional` que incluya al menos las operaciones aritmeticas basicas

```
TAD NumeroRacional {
  obs n:  $\mathbb{Z}$ 
  obs d:  $\mathbb{Z}$ 

  proc suma (in a,b: NumeroRacional) : NumeroRacional
    requiere {true}
    asegura {res =  $\frac{a.n*b.d+a.d*b.n}{b.d*a.d}$ }

  proc resta (in a,b: NumeroRacional) : NumeroRacional
    requiere {true}
    asegura {res =  $\frac{a.n*b.d-a.d*b.n}{b.d*a.d}$ }

  proc multiplicacion (in a,b: NumeroRacional) : NumeroRacional
    requiere {true}
    asegura {res =  $\frac{a.n*b.n}{b.d*a.d}$ }

  proc division (in a,b: NumeroRacional) : NumeroRacional
    requiere {b  $\neq$  0}
    asegura {res =  $\frac{a.n*b.d}{b.n*a.d}$ }
}
```

Ejercicio 2: Especificar TADs para las siguientes figuras geometricas. Tiene que contener las operaciones rotar, trasladar y escalar, y una mas propuesta por usted.

- Rectangulo (2D)
- Esfera (3D)

```
TAD Rectangulo {
  obs alto:  $\mathbb{R}$ 
  obs ancho:  $\mathbb{R}$ 
  obs posicion:  $\langle \mathbb{R} \times \mathbb{R} \rangle$ 
  obs angulo:  $\mathbb{R}$ 

  proc nuevoRectangulo (in h:  $\mathbb{R}$ , in w: float, in x: $\mathbb{R}$ ,in y: $\mathbb{R}$ ) : Rectangulo
    requiere {w  $\geq$  0  $\wedge$  h  $\geq$  0}
    asegura {res.ancho = w  $\wedge$  res.alto = h  $\wedge$  res.posicion = (x,y)  $\wedge$  r.angulo = 0}

  proc escalar (inout r: Rectangulo, mh: $\mathbb{R}$ , mw: $\mathbb{R}$ )
    requiere {r = R0}
    asegura {(mw  $\geq$  0  $\wedge$  mh  $\geq$  0)  $\wedge_L$  r.posicion = R0.posicion}
    asegura {r.alto = |R0.alto * mh|  $\wedge$  r.ancho = |R0.ancho * mw|}
    asegura {mw < 0  $\wedge_L$  r.posicion1 = R0.posicion1 + R0.ancho * mw}
    asegura {mh < 0  $\wedge_L$  r.posicion2 = R0.posicion2 + R0.alto * mh}

  proc rotar (inout r: Rectangulo, in rad:  $\mathbb{R}$ )
    requiere {r = R0}
    asegura {r.angulo = R0.angulo + rad}

  proc trasladar (inout r: Rectangulo, in pos:  $\langle x,y \rangle$ )
    requiere {r = R0}
    asegura {r.posicion = R0.posicion + pos}

  proc area (in r: Rectangulo) :  $\mathbb{R}$ 
    asegura {res = r.alto * r.ancho}
}
```

```

TAD Esfera {
  obs radio:  $\mathbb{R}$ 
  obs centro:  $\langle \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rangle$ 
  obs semieje:  $\langle \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rangle$ 

  proc nuevaEsfera (in r:  $\mathbb{R}$ , in pos:  $\langle \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rangle$ ) : Esfera
    asegura {res.radio =  $|r|$ }
    asegura {res.centro = pos}

  proc escalar (inout esfera: Esfera, in e:  $\mathbb{R}$ )
    requiere {esfera = esfera0}
    asegura {esfera.radio = ESFERA0.radio *  $|e|$ }

  proc trasladar (inout esfera: Esfera, in pos:  $\langle x, y, z \rangle$ )
    requiere {esfera = ESFERA0}
    asegura {esfera.centro = ESFERA0.centro + pos}

  proc rotar (inout esfera: Esfera, in angulosrad:  $\langle \alpha, \beta, \gamma \rangle$ )
    requiere {esfera = ESFERA0}
    asegura {esfera.semieje = ESFERA0.semieje + angulosrad}

}

```

Ejercicio 3. Especifique el TAD DobleCola $\langle T \rangle$, en el que los elementos pueden insertarse al principio o al final y se eliminan por el medio.

```

TAD DobleCola {
  obs elems: seq $\langle T \rangle$ 

  proc DobleColaVacía () : DobleCola
    asegura {res =  $\langle \rangle$ }

  proc encolarFinal (inout doblecola: DobleCola, in e: T)
    requiere {doblecola = DOBLECOLA0}
    asegura {doblecola.elems = concat(DOBLECOLA0.elems,  $\langle e \rangle$ )}

  proc encolarInicio (inout doblecola: DobleCola, in e: T)
    requiere {doblecola = DOBLECOLA0}
    asegura {doblecola.elems = concat( $\langle e \rangle$ , DOBLECOLA0.elems)}

  proc desencolar (inout doblecola: DobleCola) : T
    requiere {doblecola = DOBLECOLA0}
    requiere {cola.elems  $\neq \langle \rangle$ }
    asegura {res  $\notin$  doblecola.elems}

    asegura {res = DOBLECOLA0.elems  $\left[ \left\lceil \frac{|DOBLECOLA_0|}{2} \right\rceil \right]$  }

}

```

Ejercicio 4. Especifique el TAD `DiccionarioConHistoria`. El mismo guarda para cada clave, todos los valores que se asociaron con la misma a lo largo del tiempo (en orden)

```
TAD DiccionarioConHistoriaT,seq⟨K⟩ {
  obs data: dict<T,seq⟨K⟩>

  proc DiccionarioConHistoriaT,seq⟨K⟩ Vacio () : DiccionarioConHistoriaT,seq⟨K⟩
    asegura {res.data = {}}

  proc estaLaLlave (in dic:DiccionarioConHistoriaT,seq⟨K⟩ , in e: T) : Bool
    asegura {res = true  $\longleftrightarrow$  e  $\in$  dic.data}

  proc definir (inout dic: DiccionarioConHistoriaT,seq⟨K⟩ ,in k: T, in e: K)
    requiere {DIC0 = dic  $\wedge_L$  k  $\in$  RES0.data}
    asegura {dic.data[k][0] = setKey(DIC0.data,k,concat(DIC0.data[k],e))}

  proc consultarHistorial (in dic: DiccionarioConHistoriaT,seq⟨K⟩ , in k: T) : seq⟨K⟩
    asegura {res = res.data[k]}

  proc borrar (inout dic DiccionarioConHistoriaT,seq⟨K⟩ ,in k: T)
    requiere {dic = DIC0}
    requiere {k  $\in$  dic.data}
    asegura {dic.data = delKey(DIC0,k)}

  proc tamaño (in dic: DiccionarioConHistoriaT,seq⟨K⟩ ) :  $\mathbb{Z}$ 
    asegura {res = |dic.data|}
}
```

Ejercicio 5. Modifique el TAD `ColaDePrioridad` para que, si hay muchos valores iguales al maximo, la operacion `desapilarMax` los desapile a todos.

```
TAD ColaDePrioridad<T> {
  ---Toda la implementacion normal de ColaDePrioridad<T>---

  proc desapilarMax (inout cola:ColaDePrioridad<T>) : seq⟨T⟩
    requiere {cola = COLA0}
    asegura {  $\nexists$  e  $\in$  cola.data,  $\exists e' \in COLA_0.data$  : tienePriMax(COLA0.data,e')  $\wedge_L$  cola.data[e] = COLA0.data[e'] }
    asegura {  $\forall e \in COLA_0.data$ , (tienePriMax(COLA0.data,e)  $\wedge_L$  e  $\in$  res)  $\vee$  ( $\neg$ tienePriMax(COLA0.data,e)  $\wedge_L$  e  $\notin$  res) }
}
```